



小李的博客

www.6o06.com



前言

我们常用的操作系统是微软的 Windows 或是苹果的 OS X，因为它容易操作，所以使用者很多。其实还有一种操作系统，这个操作系统本身就是开源免费的，谁都可以免费使用和安装，它就是 linux。可是国内很少有用户使用 linux，主要是这个需要学习，不然很难操作。linux 系统入门学习教程，坚持“理论够用、侧重实用”的原则，用案例来讲解每个知识点，对 Linux 做了较为详尽的阐述，来帮助大家学习这个操作系统。 ，

| 适用人群

本教程是初级教程，旨在帮助需要使用 Linux 操作系统的程序开发者或多 Linux 系统感兴趣的技术爱好者。

| 学习前提

学习本教程需要你了解目前主流的操作系统，对 shell 脚本的编写有一定的了解。

目录

前言	1
第 1 章 关于 Linux 的历史	4
第 2 章 图形界面还是命令窗口	6
第 3 章 Linux 操作系统的安装	8
第 4 章 初步进入 linux 世界	30
第 5 章 Linux 系统的远程登录	46
第 6 章 Linux 文件与目录管理	54
第 7 章 linux 系统用户以及用户组管理	79
第 8 章 Linux 磁盘管理	87
第 9 章 文本编辑工具 vim	113
第 10 章 文档的压缩与打包	119
第 11 章 安装 RPM 包或者安装源码包	125
第 12 章 学习 shell 脚本之前的基础知识	140
第 13 章 正则表达式	158
第 14 章 shell 脚本	175
第 15 章 linux 系统日常管理	193
第 16 章 LAMP环境搭建	240
第 16 章 vim /usr/local/apache2/conf/httpd.conf	244
第 17 章 LNMP 环境搭建	249
第 18 章 学会使用简单的 MySQL 操作	258
第 19 章 NFS 服务配置	268

第 20 章	配置 FTP 服务	273
第 21 章	配置 squid 服务	277
第 22 章	配置 Tomcat	283
第 23 章	配置 samba 服务器	288
第 24 章	使用 Nagios 搭建监控服务器	294



关于 Linux 的历史



很多关于 linux 的书籍在前面章节中写了一大堆东西来介绍 linux，可惜读者看了好久也没有正式开始进入linux 的世界，这样反而导致了他们对 linux 失去了一些兴趣，而把厚厚的一本书丢掉。

Linux 的历史确实有必要让读者了解的，但是不了解也并不会影响你将来的 linux 技术水平。如果你感兴趣的话，那你去网上搜一下吧，一大堆呢足够让你看一天的。虽然我不太想啰嗦太多，但是关于linux最基本的认识，我还是想简单介绍一下的。也算是我对linux的创始人Linus Torvalds 先生的尊重。

在介绍 linux 的历史前，我想先针对大家如何对 linux 的发音说一下。我发现我身边的朋友对 linux 的发音大致有这么几种：“里那克斯”与“里你克斯”“里扭克斯”等。其实官方的标准发音为 [ˈliːn ə ks]，因为这个发音是创始人 Linus 的发音。如果你不认识这个音标，那么就读成“里那克斯”。而笔者习惯发音成“里你克斯”，当然你发音成什么，并没有人会说你，完全是一个人的习惯而已。

也许有的读者已经了解到，linux 和 unix 是非常像的。没错，linux 就是根据 unix 演变过来的。当年 linux 就是因为接触到了unix而后才自己想开发一个简易的系统内核的，他开发的简易系统内核其实就是linux。当时linux 把开发的这个系统内核丢到网上供大家下载，由于它的精致小巧，越来越多的爱好者去研究它。人们对这个内核添枝加叶，而后成为了一个系统。也许你听说过吧，linux 是免费的。其实这里的免费只是说linux的内核免费。在 linux 内核的基础上而产生了众多 linux 的版本。

Linux 的发行版说简单点就是将 Linux 内核与应用软件做一个打包。较知名的发行版有：Ubuntu、RedHat、CentOS、Debian、Fedora、SuSE、OpenSUSE、TurboLinux、BluePoint、RedFlag、Xterm、SlackWare 等

而笔者常用的就是 Redhat 和 CentOS，这里有必要说一下，其实 CentOS 是基于 Redhat 的，网上有人说，Centos 是 Redhat 企业版的克隆。笔者所在公司的服务器全部都是安装 CentOS 系统，并且相当稳定。CentOS 较之于 Redhat 可以免费使用 yum 下载安装所需要的软件包，这个是相当方便的。而Redhat要想使用 yum 必须要购买服务了。



T



2

图形界面还是命令窗口

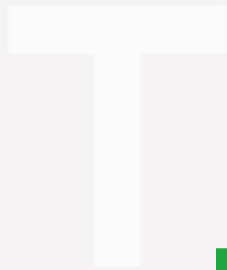


对于 linux 的应用，我想大多数都是用在服务器领域，对于服务器来讲真的没有必要跑一个图形界面。所以我们平时安装 linux 操作系统时往往是不安装图形界面的。说到这里也许你会有疑问，图形界面还能选择装或者不装？

是的，虽然 linux 和微软的 Windows 一样同位操作系统，但是它们有一个很大的区别就是 Windows 操作系统的图形界面是和内核一体的，俗称微内核，而 linux 操作系统图形界面就像一个软件一样，和内核并不是一体的。所以 linux 你可以选择不安装图形界面，这样不仅不影响服务器的正常使用还可以节省系统资源的开销，何乐而不为呢？

如果你对 linux 超级感兴趣，想使用 linux 就像使用 Windows 一样，那你可以安装图形界面，可以像 Windows 一样来体验 linux 也是蛮不错的。但是如果你想成为一个专业的 linux 系统工程师，那我建议你从第一天开始就不要去安装图形界面，从命令窗口开始熟悉它。

另外一点值得说的是，日常应用中，我们都是远程管理服务器的，不可能开着图形界面来让你去操作，虽然目前也有相应的工具支持远程图形连接服务器，可是那样太消耗网络带宽资源，所以从这方面来考虑还是建议你不要使用图形界面。



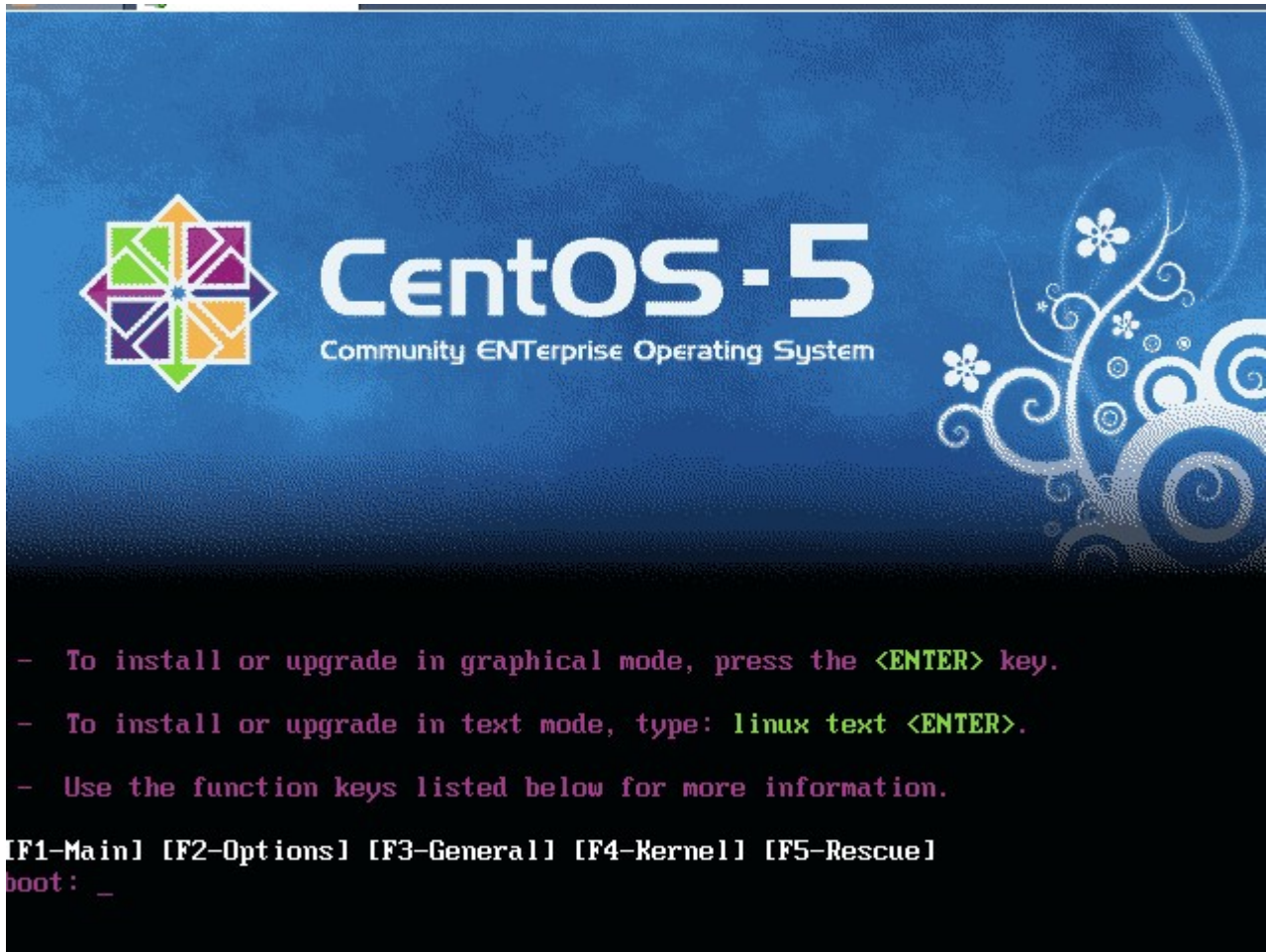
3

Linux 操作系统的安装



因为笔者一直都是使用 CentOS，所以这次安装系统也是基于CentOS的安装。把光盘插入光驱，设置bios光驱启动。进入光盘的欢迎界面。也可参考[Ubuntu 12.04 安装教程详细步骤 \(http://www.jb51.net/os/84475.html\)](http://www.jb51.net/os/84475.html)

。



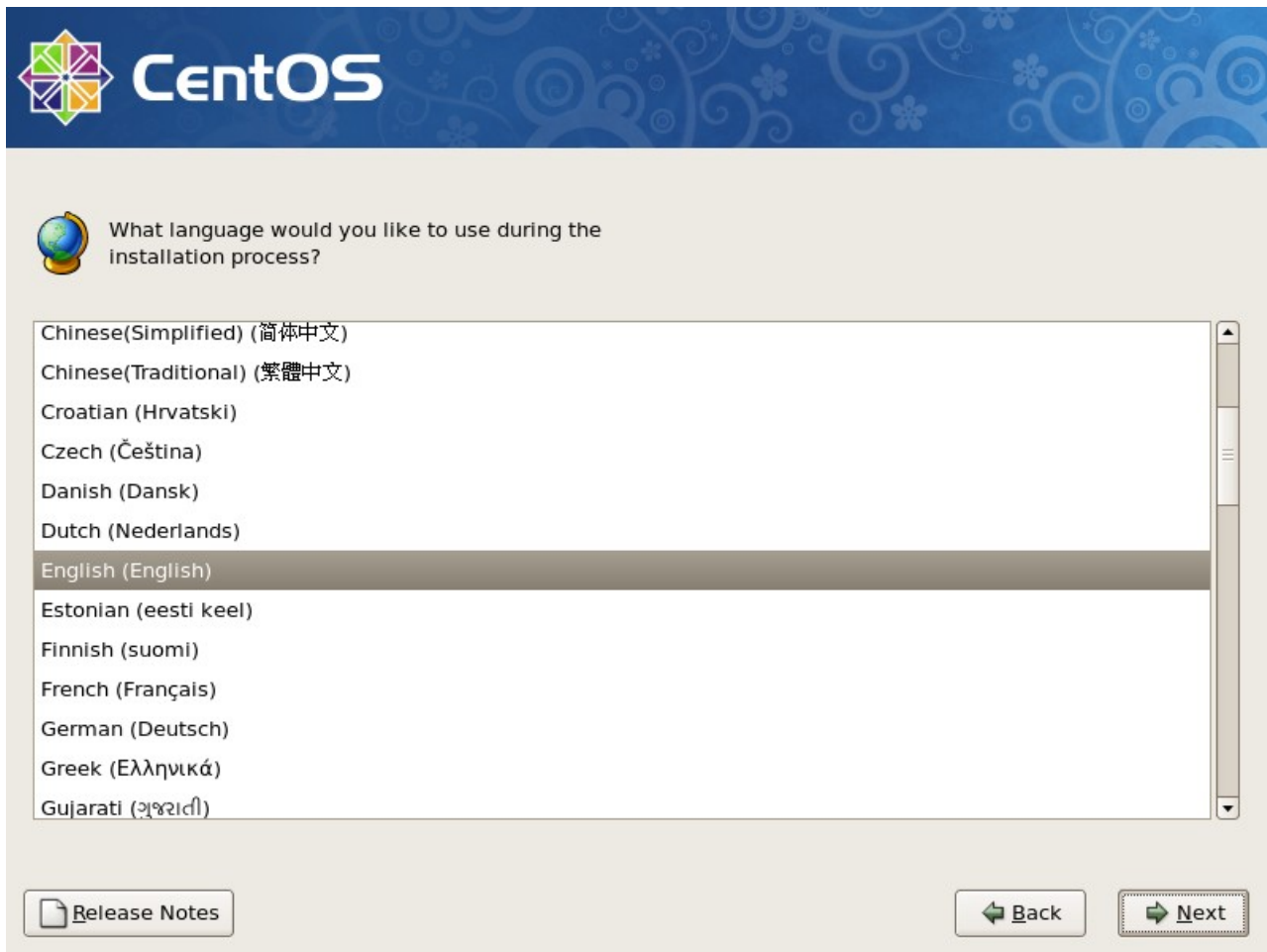
其中有两个选项，可以直接按回车，也可以在当前界面下输入 linux text 按回车。前者是图形下安装，可以动鼠标的，后者是纯文字形式的。建议初学者用前者安装。直接回车后，出现一下界面：



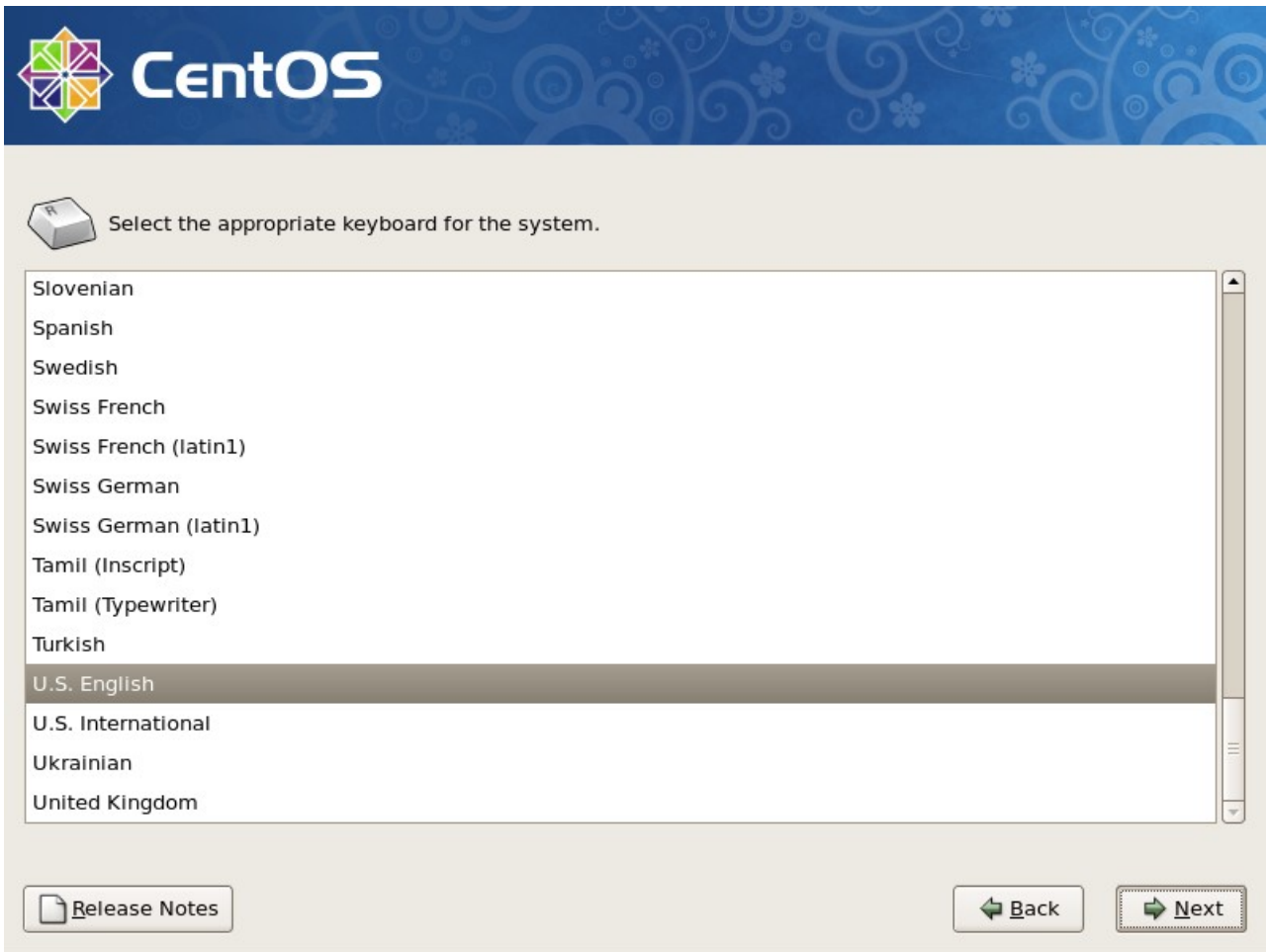
这一步是要提示你是否要校验光盘，目的是看看光盘中的安装包是否完整或者是否被人改动过，一般情况下，如果是正规的光盘不需要做这一步操作，因为太费时间。接下来是：



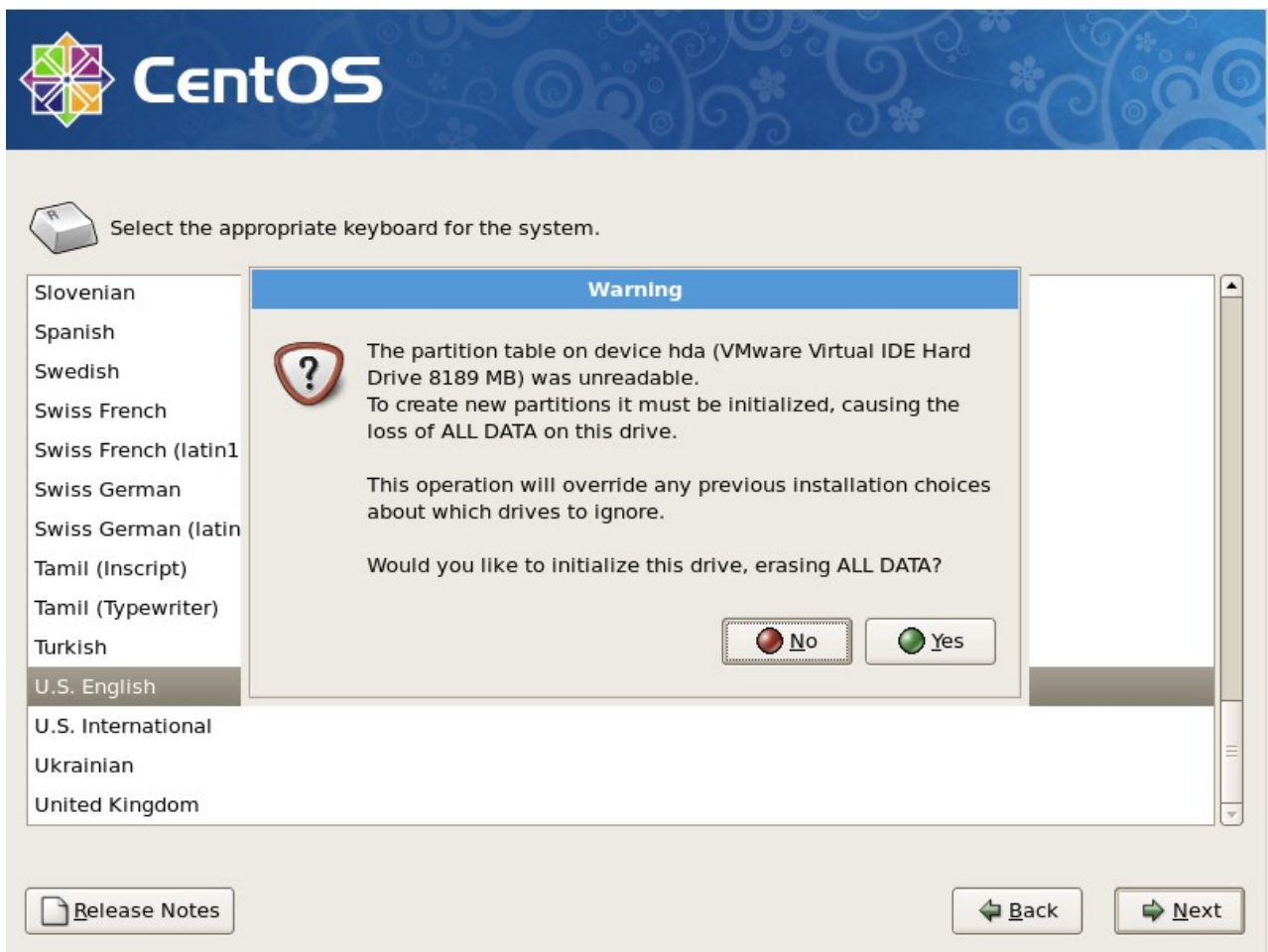
这一步没有什么可说明的，直接点“Next”：



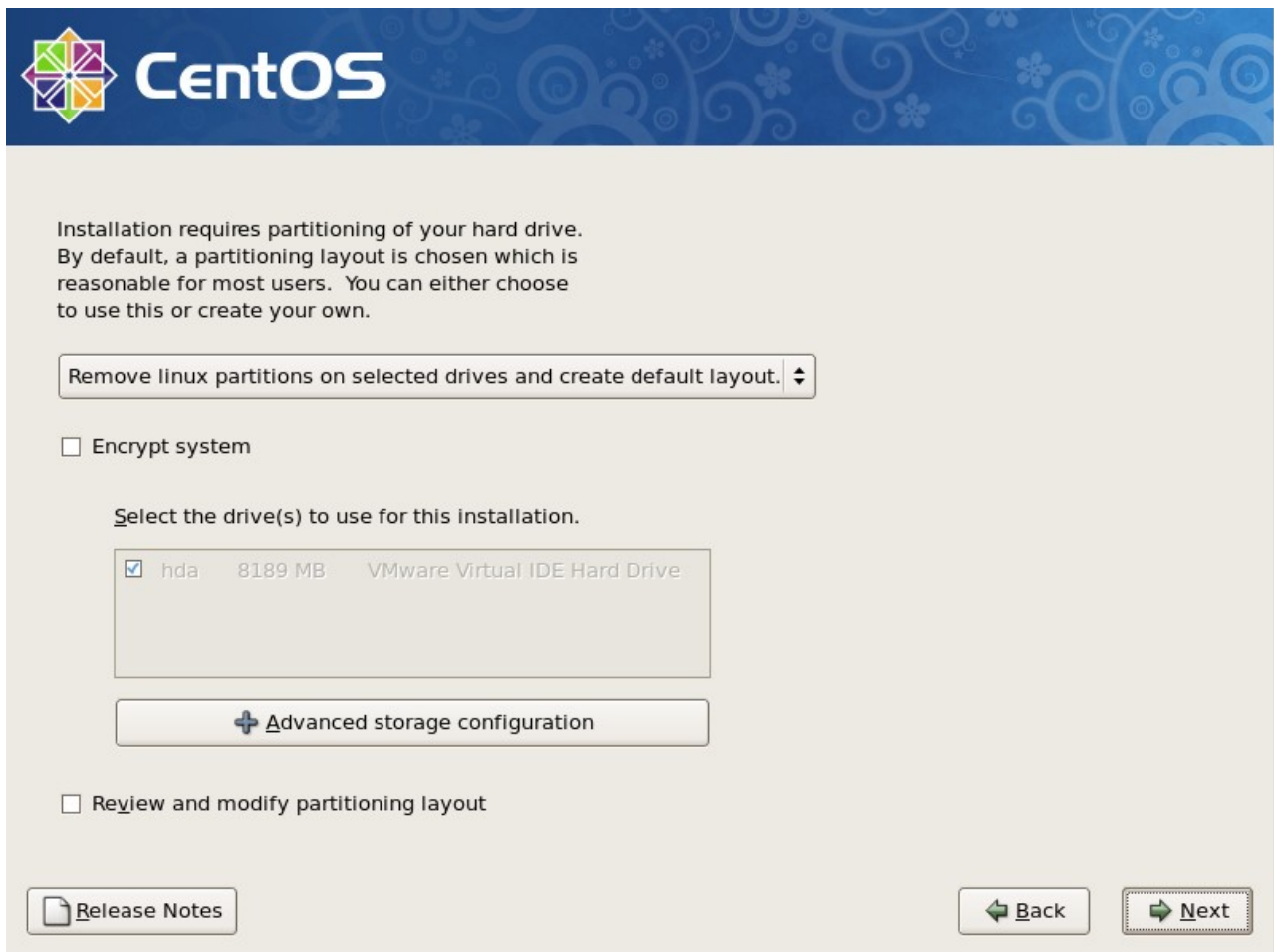
这里是选择安装系统时所用语言，笔者习惯用 English，当然你也可以选择 Chinese(Simplified)（简体中文），选择好后点“Next”：



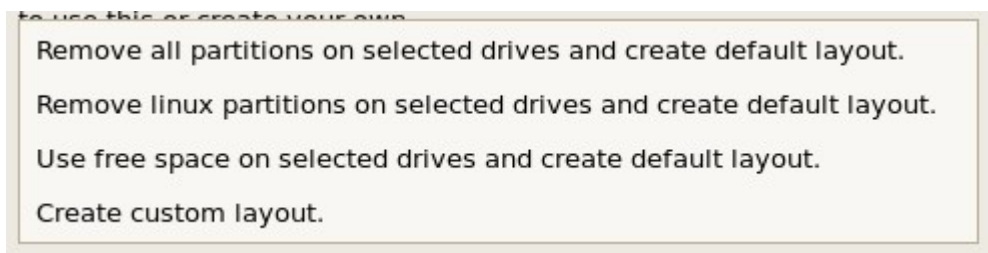
这里是选择合适的键盘，我们平时使用的都是英文键盘，所以这里不用动，默认即可，直接“Next”：



到这里就会提示你，下面会分区，会初始化磁盘，磁盘上的数据会丢失，问你是否要初始化设备并清除磁盘上的数据。因为是空盘，所以选择“**Yes**”：



到这一步，就该分区了。其中一共有四种方式可以供你选择：



第一种，在所选磁盘上把所有分区移除，然后按照默认的方式分区；

第二种，在所选磁盘上把所有 linux 分区移除（如果磁盘上有 Windows 格式的分区，并不会移除），然后按照默认方式分区；

第三种，在所选磁盘上只使用空闲部分，并且按照默认方式分区；

第四种，用户自定义。

这里我们选第四种。然后“Next”：



CentOS

Installation requires partitioning of your hard drive. By default, a partitioning layout is chosen which is reasonable for most users. You can either choose to use this or create your own.

Create custom layout.

Encrypt system

Select the drive(s) to use for this installation.

hda 8189 MB VMware Virtual IDE Hard Drive

+ Advanced storage configuration

Review and modify partitioning layout

Release Notes

Back

Next



接下来该分区了，分区的很灵活，但大体上按照这个规则来（这是服务器上这样分，如果你是虚拟机，请看后边部分）：

1./boot 分区 100M

Add Partition

Mount Point: /boot

File System Type: ext3

Allowable Drives:

<input checked="" type="checkbox"/>	hda	8189 MB	VMware Virtual IDE Hard Drive
-------------------------------------	-----	---------	-------------------------------

Size (MB): 100

Additional Size Options

Fixed size

Fill all space up to (MB): 1

Fill to maximum allowable size

Force to be a primary partition

Encrypt

2.Swap 分区 内存的2倍，如果大于等于 4G，则只需给 4G 即可：

Add Partition

Mount Point: <Not Applicable>

File System Type: swap

Allowable Drives:

<input checked="" type="checkbox"/>	hda	8189 MB	VMware Virtual IDE Hard Drive
-------------------------------------	-----	---------	-------------------------------

Size (MB): 4000

Additional Size Options

Fixed size

Fill all space up to (MB): 20000

Fill to maximum allowable size

Force to be a primary partition

Encrypt

3./ 分区给 20G

Add Partition

Mount Point: /

File System Type: ext3

Allowable Drives: hda 8189 MB VMware Virtual IDE Hard Drive

Size (MB): 20000

Additional Size Options

Fixed size

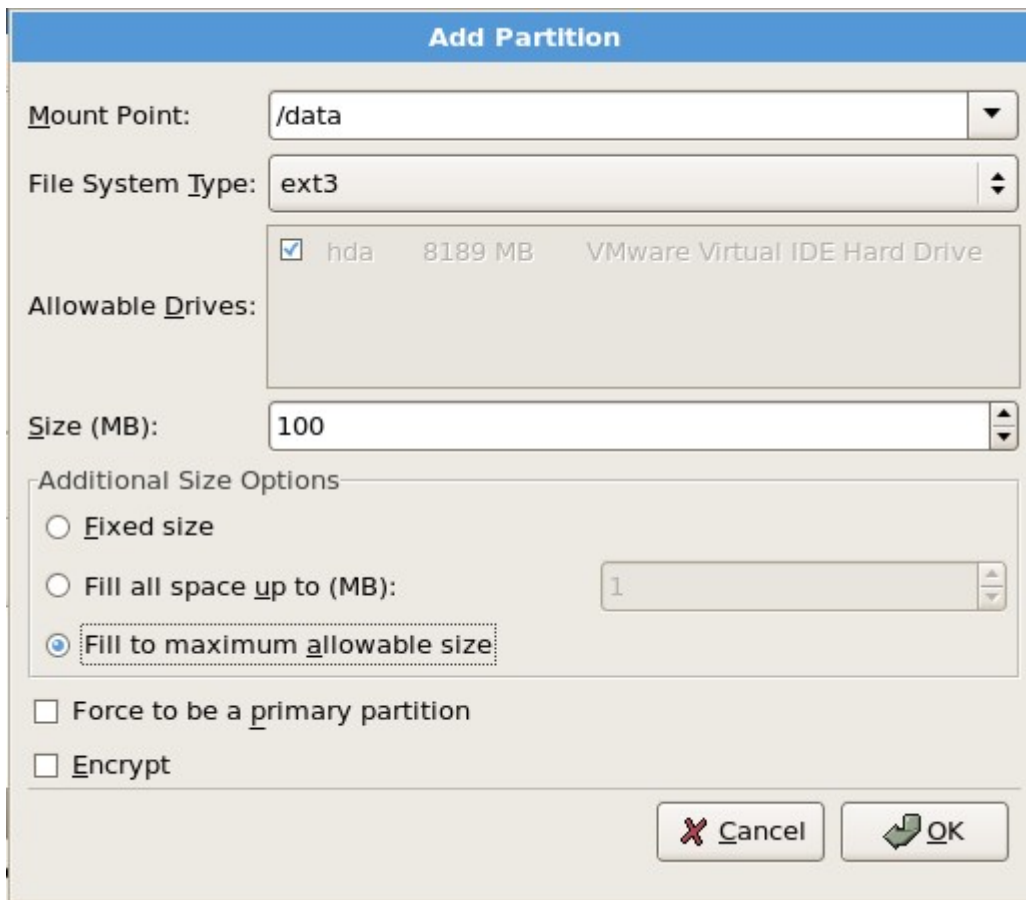
Fill all space up to (MB): 1

Fill to maximum allowable size

Force to be a primary partition

Encrypt

4. 剩余空间给/data



说明：/boot 分区是系统启动所需要的文件，就跟 Windows 的 C 盘中的 windows 目录类似，这个分区中的文件并不大，只需要 100M 足够。Swap 分区是交换分区，当内存不够时，系统会把这部分空间当内存使用。/ 分区，其实就是一个根目录，在以后的章节中会介绍到。现在不懂并没有关系，只要知道有这么一个东西即可。/data 这个分区是我们自定义的，就是专门放数据的分区。


如果你安装的是虚拟机，并且你只有 8G 的磁盘空间，那么我建议你这样分区：

1 /boot 100M

2 swap 内存的 2 倍

3 / 全部剩余空间

分区完后，点“Next”：



CentOS

The GRUB boot loader will be installed on /dev/hda.

No boot loader will be installed.

You can configure the boot loader to boot other operating systems. It will allow you to select an operating system to boot from the list. To add additional operating systems, which are not automatically detected, click 'Add.' To change the operating system booted by default, select 'Default' by the desired operating system.

Default	Label	Device
<input checked="" type="checkbox"/>	CentOS	/dev/hda3

[Add](#)
[Edit](#)
[Delete](#)

A boot loader password prevents users from changing options passed to the kernel. For greater system security, it is recommended that you set a password.

Use a boot loader password [Change password](#)

Configure advanced boot loader options

[Release Notes](#) [Back](#) [Next](#)

可以在 Use a boot loader password 前面打勾，这个选项的作用是，给 boot loader 加一个密码，为了防止有人通过光盘进入单用户模式修改 root 密码。

下面的选项同样可以打勾，笔者从来没有用过该功能，如果你有兴趣，可以研究一下。然后下一步。



The screenshot shows the CentOS network configuration interface. At the top is the CentOS logo and name. Below it is the 'Network Devices' section with a table of network interfaces. The 'Active on Boot' checkbox is checked for the 'eth0' interface, which is configured with 'DHCP' and 'Auto' settings. An 'Edit' button is next to the table. The 'Hostname' section offers two options: 'automatically via DHCP' (selected) and 'manually' with a text input field containing 'localhost.localdomain'. The 'Miscellaneous Settings' section has three empty input fields for 'Gateway', 'Primary DNS', and 'Secondary DNS'. At the bottom, there are buttons for 'Release Notes', 'Back', and 'Next'.

Active on Boot	Device	IPv4/Netmask	IPv6/Prefix
<input checked="" type="checkbox"/>	eth0	DHCP	Auto

Hostname
Set the hostname:
 automatically via DHCP
 manually (e.g., host.domain.com)

Miscellaneous Settings
Gateway:
Primary DNS:
Secondary DNS:

[Release Notes](#) [Back](#) [Next](#)

这一步是配置网卡信息，可以现在自定义网卡的 IP，和配置主机名，默认是通过 DHCP 获得，你也可以点 manually 自定义一个主机名，如 mail.example.com。如果这两种方式都没有配置，那么 linux 会给你配置一个万能的主机名，即 localhost.localdomain 剩下的几个就不用配置了，默认留空。

接着下一步，选择时区，在这里当然要选择我们所处的时区 Asia/Chongqing 如果没有 Chongqing 那就选择 Asia/Shanghai。



继续下一步：



CentOS

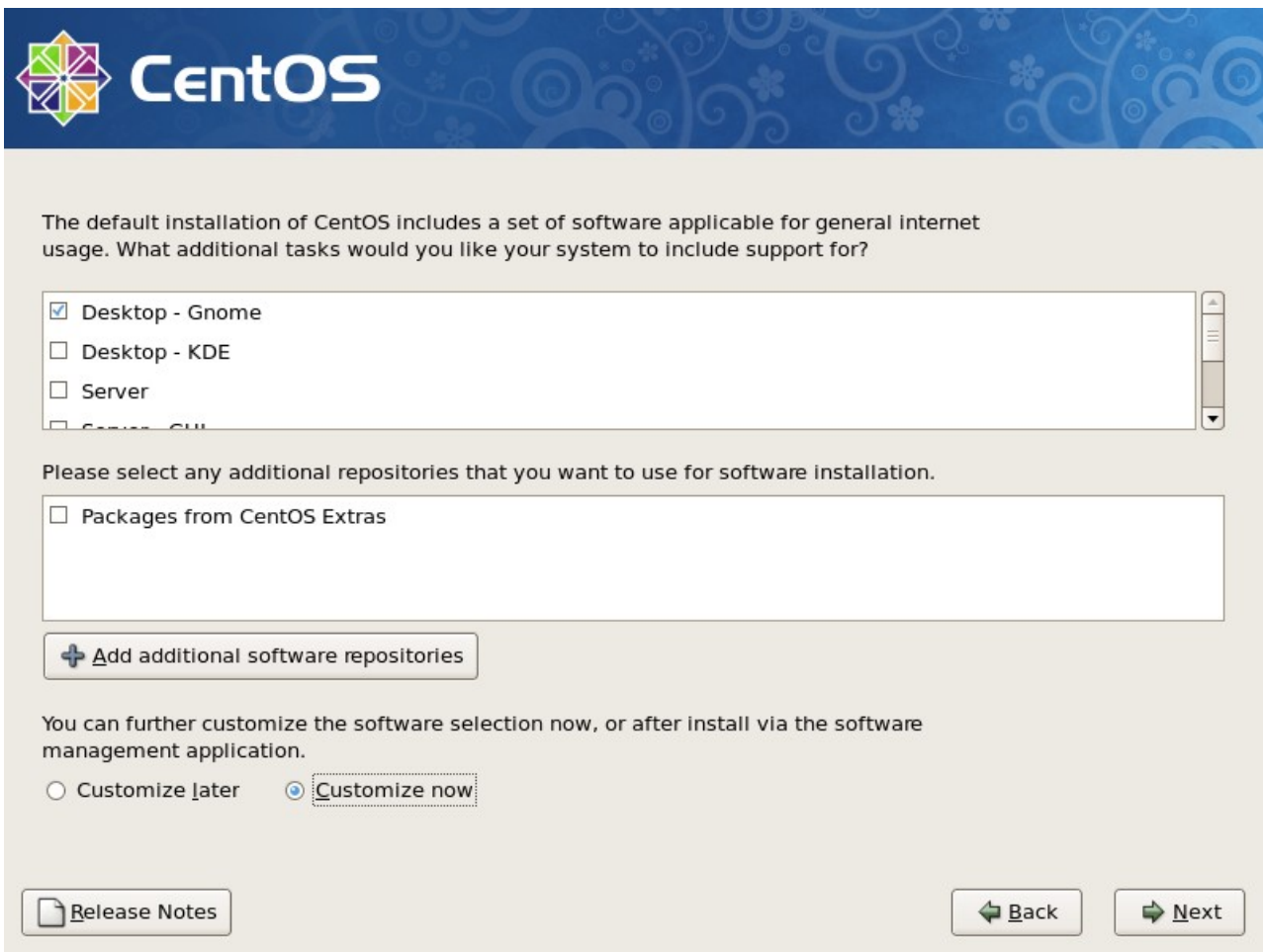
 The root account is used for administering the system. Enter a password for the root user.

Root Password:

Confirm:

[Release Notes](#) [Back](#) [Next](#)

在这里自己定义一个 root 的密码，继续下一步：

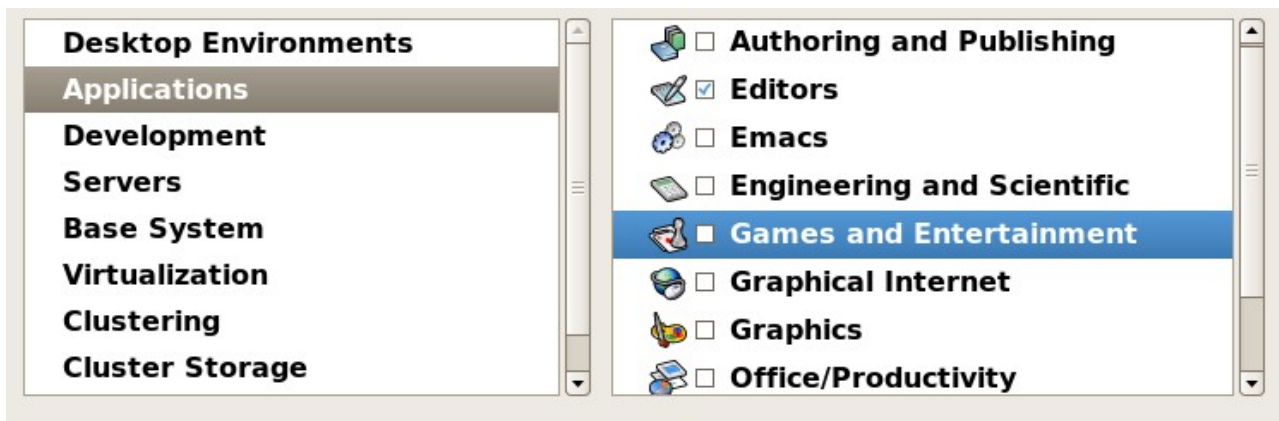


这里我们要选择要安装的包，笔者习惯自定义安装，需要点下面的“customize now”然后下一步

“Desktop Environments”看右侧，把 GNOME 前面的勾去掉，这个其实就是图形界面的安装包，如果不去掉这个勾，就会安装图形界面。



“Applications”除了 Editors 前面的勾去掉外，其他均不要



“Development” 全部都要勾上



“Servers” 以及以下所有项都不要勾任何，然后下一步



点 Next 后，系统就开始安装了。



等过会后，会出现



至此，linux 系统已经安装完成了。接下来点“Reboot”重启一下，进入 linux 系统看看吧。



T



4

初步进入 linux 世界



Linux 系统启动过程

Linux 的启动其实和 Windows 的启动过程很类似，不过 Windows 我们是无法看到启动信息的，而 linux 启动时我们会看到许多启动信息，例如某个服务是否启动。

Linux 系统的启动过程大体上可分为五部分：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。紧接着由启动设备上的 grub 程序开始引导 linux，当引导程序成功完成引导任务后，Linux 从它们手中接管了 CPU 的控制权，然后 CPU 就开始执行 Linux 的核心映像代码，开始了 Linux 启动过程。也就是所谓的内核引导开始了，在内核引导过程中其实是很复杂的，我们就当它是一个黑匣子，反正是 linux 内核做了一些列工作，最后内核调用加载了 init 程序，至此内核引导的工作就完成了。交给了下一个主角 init。

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。inittab 是一个不可执行的文本文件，它有若干行指令所组成。具体内容如下：（你可以在你的linux上执行命令 cat /etc/inittab 这样获得）

```
# inittab      This file describes how the INIT process should set up
#              the system in a certain run-level.
#
# Author:      Miquel van Smoorenburg,
#              Modified for RHS Linux by Marc Ewing and Donnie Barnes
#
# Default runlevel. The runlevels used by RHS are:
# 0 - halt (Do NOT set initdefault to this)
# 1 - Single user mode
# 2 - Multiuser, without NFS (The same as 3, if you do not havenetworking)
# 3 - Full multiuser mode
# 4 - unused
# 5 - X11
# 6 - reboot (Do NOT set initdefault to this)
#
###表示当前缺省运行级别为5(initdefault);
id:5:initdefault:
###启动时自动执行/etc/rc.d/rc.sysinit脚本(sysinit)
```



```

# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
###当运行级别为5时，以5为参数运行/etc/rc.d/rc脚本，init将等待其返回(wait)
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
###在启动过程中允许按CTRL-ALT-DELETE重启系统
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
# When our UPS tells us power has failed, assume we have a few minutes
# of power left. Schedule a shutdown for 2 minutes from now.
# This does, of course, assume you have powerd installed and your
# UPS connected and working correctly.
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"
# If power was restored before the shutdown kicked in, cancel it.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"
###在2、3、4、5级别上以ttyX为参数执行/sbin/mingetty程序，打开ttyX终端用于用户登录，
###如果进程退出则再次运行mingetty程序(respawn)
# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
###在5级别上运行xdm程序，提供xdm图形方式登录界面，并在退出时重新执行(respawn)
# Run xdm in runlevel 5
x:5:respawn:/etc/X11/prefdm -nodaemon

```

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

```
id:runlevel:action:process
```

对上面各项的详细解释如下：

1.id

id是指入口标识符，它是一个字符串，对于getty或mingetty等其他login程序项，要求id与tty的编号相同，否则getty程序将不能正常工作。

2.Runlevel

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

3.action

action 是描述其后的 process 的运行方式的。action 可取的值包括：initdefault、sysinit、boot、bootwait 等：initdefault 是一个特殊的 action 值，用于标识缺省的启动级别；当 init 由核心激活以后，它将读取 inittab 中的 initdefault 项，取得其中的 runlevel，并作为当前的运行级别。如果没有 inittab 文件，或者其中没有 initdefault 项，init 将在控制台上请求输入 runlevel。sysinit、boot、bootwait 等 action 将在系统启动时无条件运行，而忽略其中的 runlevel。其余的 action（不含 initdefault）都与某个 runlevel 相关。各个 action 的定义在 inittab 的 man 手册中有详细的描述。

4.process

process 为具体的执行程序。程序后面可以带参数。

Tips: 如果你看不懂这个文件，没有关系，随着你对 linux 的深入了解，你再回过头看这个文件你就会豁然开朗的。但是你现在必须要明白 runlevel 的各个级别的含义。

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。

rc.sysinit 约有 850 多行，但是每个单一的功能还是比较简单，而且带有注释，建议有兴趣的用户可以自行阅读自己机器上的该文件，以了解系统初始化所详细情况。由于此文件较长，所以不在本文中列出来，也不做具体的介绍。当 rc.sysinit 程序执行完毕后，将返回 init 继续下一步。通常接下来会执行到 /etc/rc.d/rc 程序。以运行级别 3 为例，init 将执行配置文件 inittab 中的以下这行：

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

这一行表示以 5 为参数运行 /etc/rc.d/rc，/etc/rc.d/rc 是一个 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使用法，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 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，以使用户登录系统。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

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。

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

login 程序成功后，会向对应的终端在输出最近一次登录的信息(在 /var/log/lastlog 中有记录)，并检查用户是否有新邮件(在 /usr/spool/mail/ 的对应用户名目录下)。然后开始设置各种环境变量：对于 bash 来说，系统首先寻找 /etc/profile 脚本文件，并执行它；然后如果用户的主目录中存在 .bash_profile 文件，就执行它，在这些文件中又可能调用了其它配置文件，所有的配置文件执行后后，各种环境变量也设好了，这时会出现大家熟悉的命令行提示符，到此整个启动过程就结束了。

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。

学会使用快捷键

Ctrl + C：这个是用来终止当前命令的快捷键，当然你也可以输入一大串字符，不想让它运行直接 Ctrl + C，光标就会跳入下一行。

Tab：这个键是最有用的键了，也是笔者敲击概率最高的一个键。因为当你打一个命令打一半时，它会帮你补全的。不光是命令，当你打一个目录时，同样可以补全，不信你试试。

Ctrl + D：退出当前终端，同样你也可以输入 exit。

Ctrl + Z：暂停当前进程，比如你正运行一个命令，突然觉得有点问题想暂停一下，就可以使用这个快捷键。暂停后，可以使用 fg 恢复它。

Ctrl + L：清屏，使光标移动到第一行。

学会查询帮助文档 — man

这个 man 通常是用来看一个命令的帮助文档的。例如：

```
[root@localhost ~]# man ls
LS(1)                                User Commands                                LS(1)

NAME
  ls - list directory contents

SYNOPSIS
  ls [OPTION]... [FILE]...

DESCRIPTION
  List information about the FILES (the current directory by default).  So
  t entries alphabetically if none of -cftuvSUX nor --sort.

  Mandatory arguments to long options are mandatory for short options too.

  -a, --all
        do not ignore entries starting with .

  -A, --almost-all
        do not list implied . and ..

  --author
        with -l, print the author of each file
```

输入 man ls 其实格式为 man + 命令

你就会看到相关的帮助文档了。从命令的介绍到命令的参数以及用法介绍的都非常详细的。不错吧。

Linux 系统目录结构

登录系统后，在当前命令窗口下输入 ls / 你会看到

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

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` 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 的预设置也是在这里。

如何正确关机

其实，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。

linux 和 Windows 不同，在 Linux 底下，由于每个程序（或者说是服务）都是在在背景下执行的，因此，在你看不到的屏幕背后其实可能有相当多人同时在你的主机上面工作，例如浏览网页啦、传送信件啦以 FTP 传送档案啦等等的，如果你直接按下电源开关来关机时，则其它人的数据可能就此中断！那就伤脑筋了！此外，最大的问题是，若不正常关机，则可能造成文件系统的毁损（因为来不及将数据回写到档案中，所以有些服务的档案会有问题！）。

如果你要关机，必须要保证当前系统中没有其他用户在线。可以下达 who 这个指令，而如果要查看网络的联机状态，可以下达 netstat -a 这个指令，而要看背景执行的程序可以执行 ps -aux 这个指令。使用这些指令可以让你稍微了解主机目前的使用状态！（这些命令在以后的章节中会提及，现在只要了解即可！）

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

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

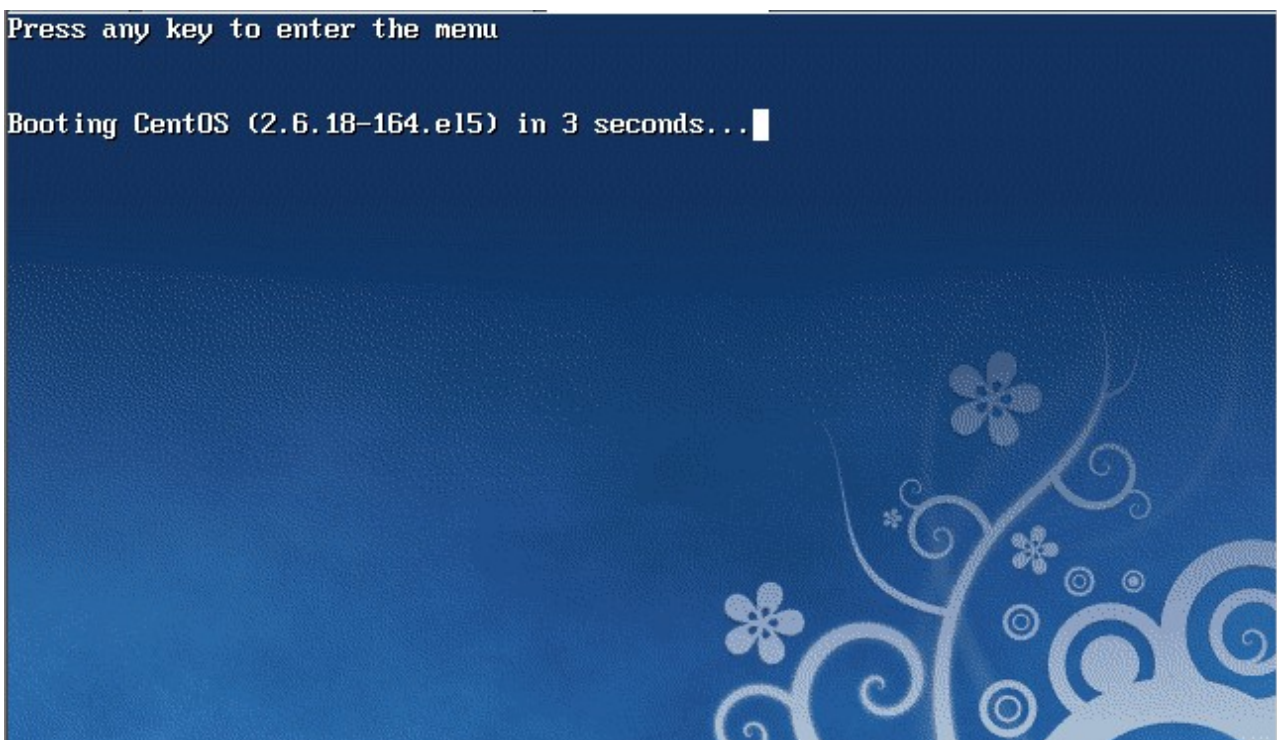
halt 关闭系统，等同于 shutdown -h now 和 poweroff

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。

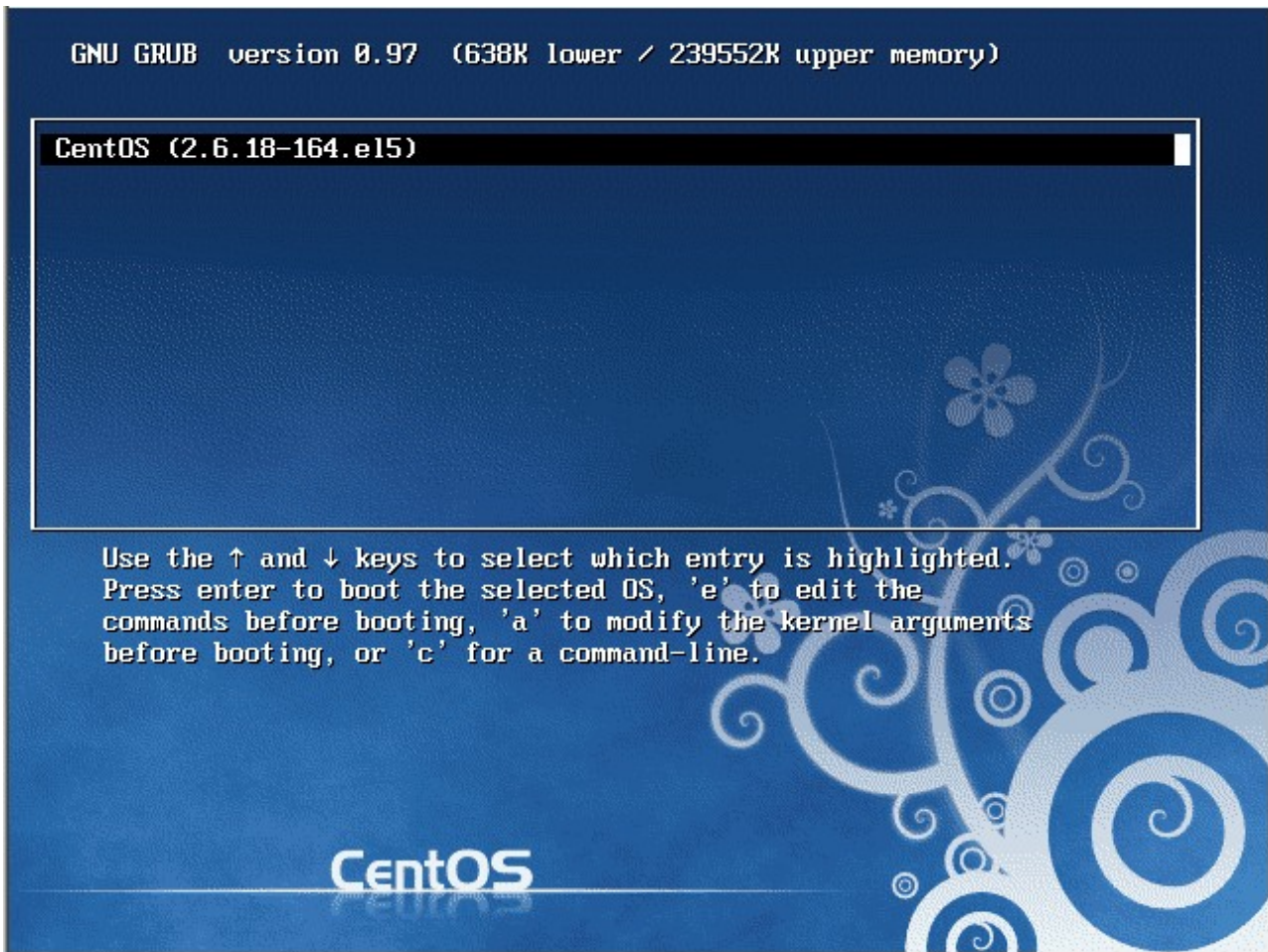
忘记 root 密码如何做

以前笔者忘记 Windows 的管理员密码，由于不会用光盘清除密码最后只能重新安装系统。现在想想那是多么愚笨的一件事情。同样 linux 系统你也会遇到忘记 root 密码的情况，如果遇到这样的情况怎么办呢？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。如何进入呢。

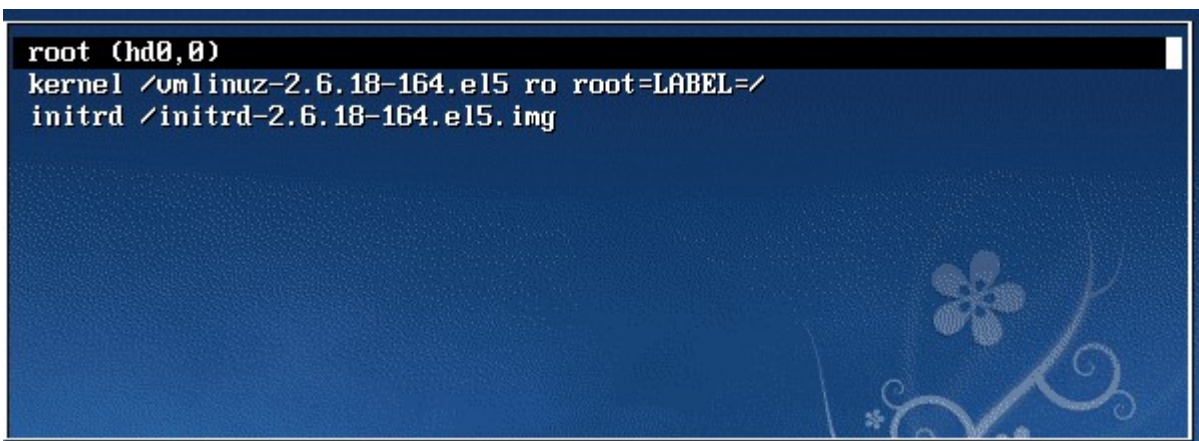
1 重启 linux 系统



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

```
[ Minimal BASH-like line editing is supported. For the first word, TAB
  lists possible command completions. Anywhere else TAB lists the possible
  completions of a device/filename. ESC at any time cancels. ENTER
  at any time accepts your changes.]

grub edit> kernel /vmlinuz-2.6.18-164.el5 ro root=LABEL=/ single
```

在后边加上 single 回车

```
root (hd0,0)
kernel /vmlinuz-2.6.18-164.el5 ro root=LABEL=/ single
initrd /initrd-2.6.18-164.el5.img
```

最后按“b”启动，启动后就进入了单用户模式了

```
no fstab.sys, mounting internal defaults
Switching to new root and running init.
unmounting old /dev
unmounting old /proc
unmounting old /sys
type=1404 audit(1303914022.636:2): enforcing=1 old_enforcing=0 auid=4294967295 s
ses=4294967295
type=1403 audit(1303914023.222:3): policy loaded auid=4294967295 ses=4294967295
INIT: version 2.86 booting
       Welcome to CentOS release 5.4 (Final)
       Press 'I' to enter interactive startup.
Setting clock (utc): Wed Apr 27 22:20:50 CST 2011      [ OK ]
Starting udev:                                       [ OK ]
Loading default keymap (us):                         [ OK ]
Setting hostname localhost.localdomain:              [ OK ]
No devices found
Setting up Logical Volume Management:                [ OK ]
Checking filesystems
/: clean, 118508/1969568 files, 713597/1967962 blocks
/boot: clean, 35/26104 files, 14714/104388 blocks
                                                    [ OK ]
Remounting root filesystem in read-write mode:      [ OK ]
Mounting local filesystems:                          [ OK ]
Enabling /etc/fstab swaps:                           [ OK ]
sh-3.2#
```

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

```
sh-3.2# passwd
Changing password for user root.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

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

```

- To install or upgrade in graphical mode, press the <ENTER> key.
- To install or upgrade in text mode, type: linux text <ENTER>.
- Use the function keys listed below for more information.

[F1-Main] [F2-Options] [F3-General] [F4-Kernel] [F5-Rescue]
boot: _

```

输入 linux rescue 回车

```

[F1-Main] [F2-Options] [F3-General] [F4-Kernel] [F5-Rescue]

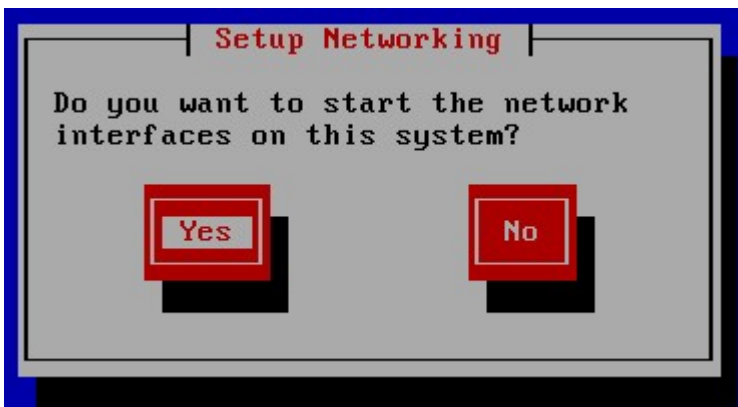
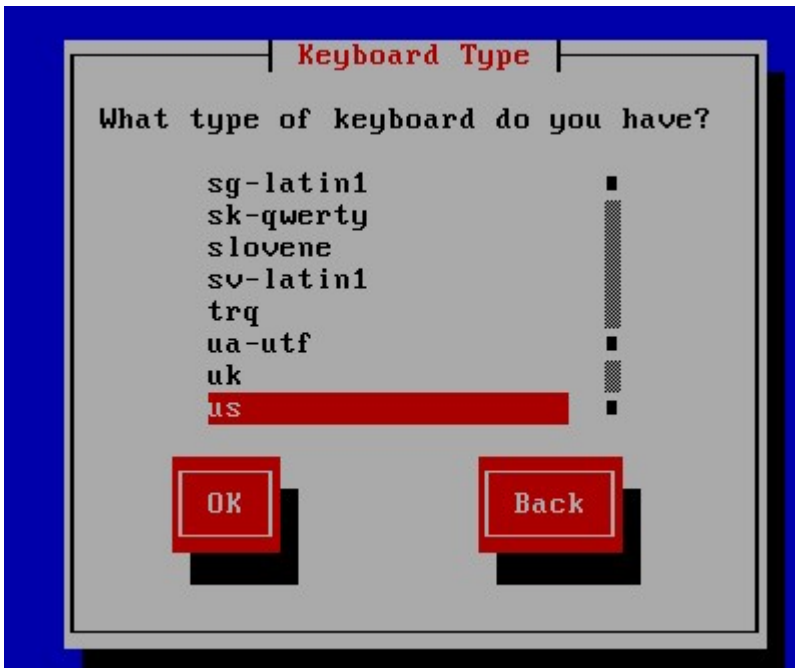
boot: linux rescue_

```

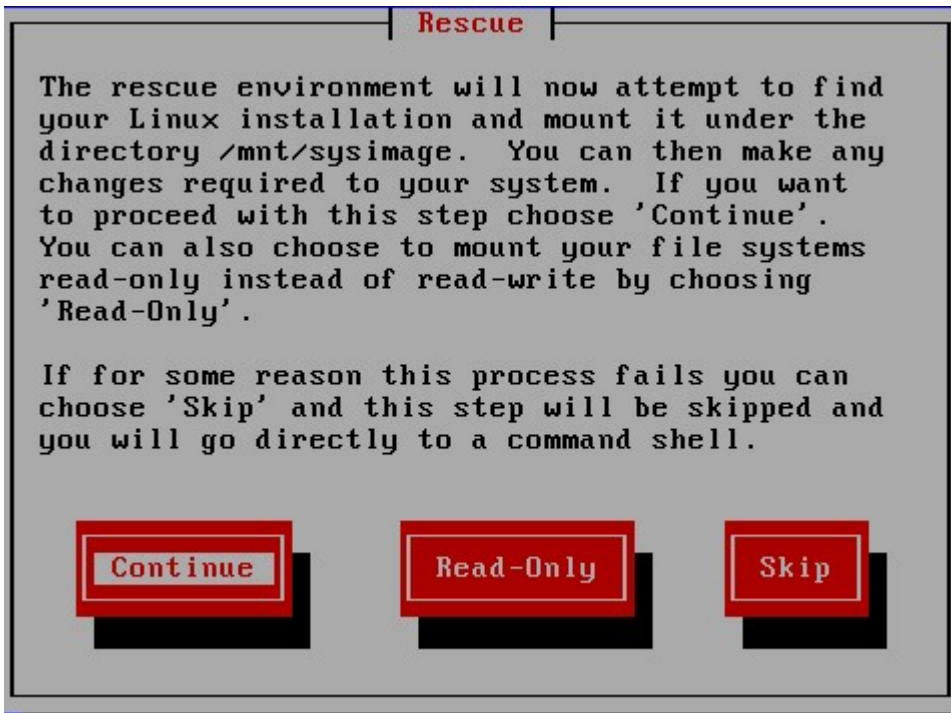
选择语言，笔者建议你选择英语



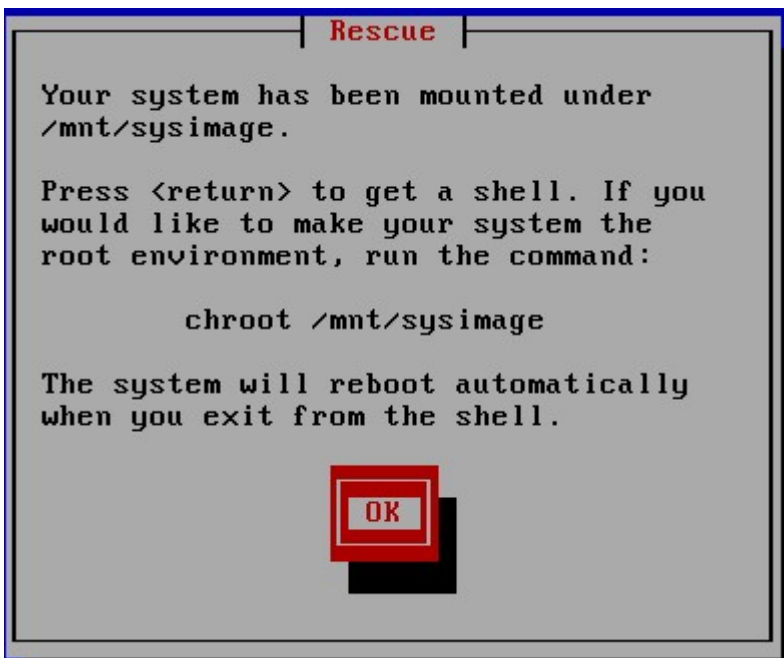
选择 us 键盘



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进一步损坏；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。

```
Your system is mounted under the /mnt/sysimage directory.  
When finished please exit from the shell and your system will reboot.  
  
sh-3.2# chroot /mnt/sysimage/  
sh-3.2# _
```

Tips: 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

```
sh-3.2# ls  
bin  etc  lib  modules  proc  sbin  sys  usr  
dev  init  mnt  oldtmp  root  selinux  tmp  var  
sh-3.2# ls /mnt/  
runtime  source  sysimage  
sh-3.2# _
```

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。



5

Linux 系统的远程登录

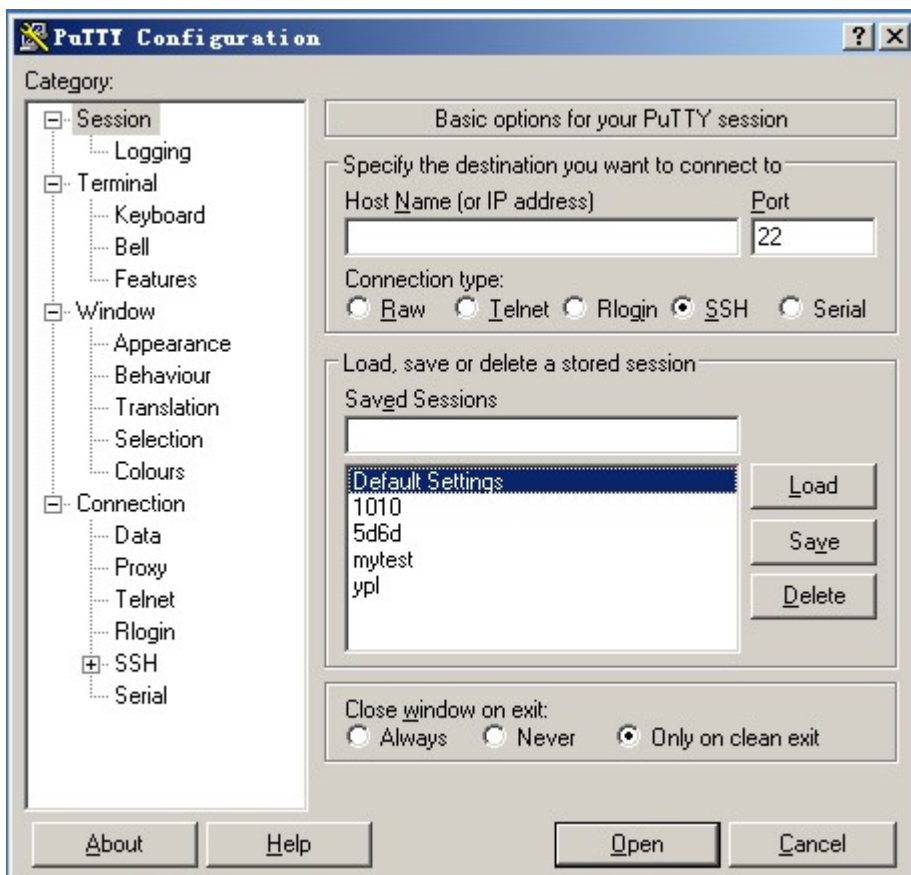


首先要说一下，该部分内容对于linux初学者来讲并不是特别重要的，可以先跳过该章节，先学下一章，等学完后再回来看这一章。

Linux 大多应用于服务器，而服务器不可能像 PC 一样放在办公室，它们是放在 IDC 机房的，所以我平时登录 linux 系统都是通过远程登录的。Linux 系统中是通过 ssh 服务实现的远程登录功能。默认 ssh 服务开启了 22 端口，而且当我们安装完系统时，这个服务已经安装，并且是开机启动的。所以不需要我们额外配置什么就能直接远程登录linux系统。ssh服务的配置文件为 /etc/ssh/sshd_config，你可以修改这个配置文件来实现你想要的 ssh 服务。比如你可以更改启动端口为 36000。

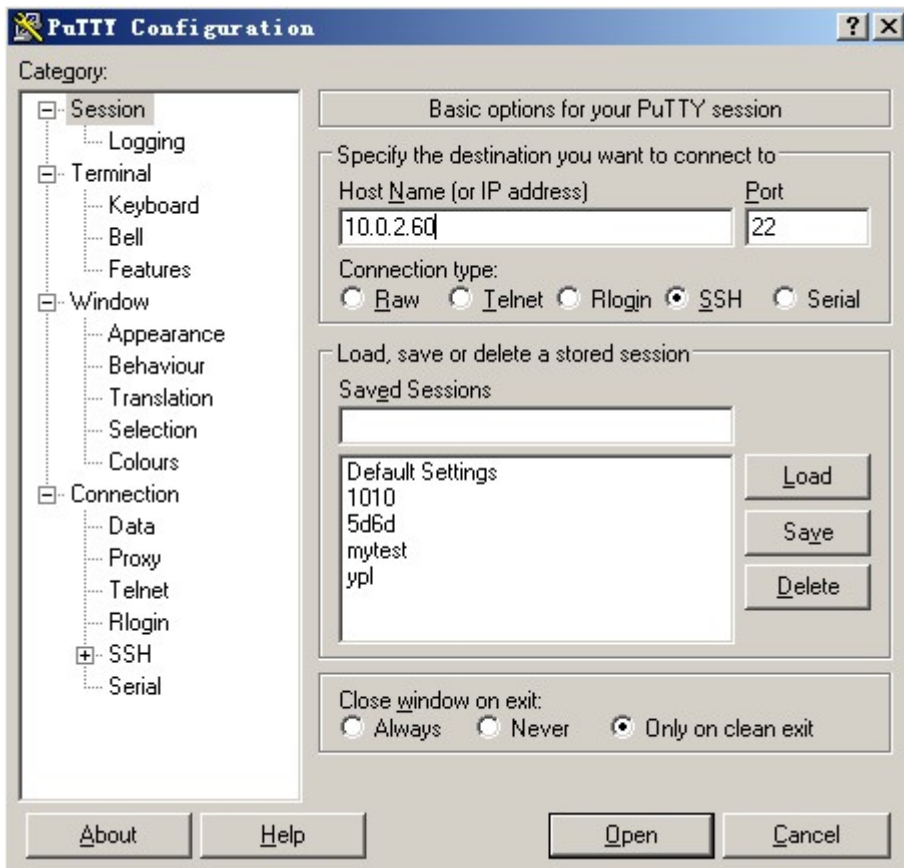
如果你是 Windows 的操作系统，则Linux远程登录需要在我们的机器上额外安装一个终端软件。目前比较常见的终端登录软件有SecureCRT, Putty, SSH Secure Shell等，很多朋友喜欢用SecureCRT因为它的功能是很强大的，而笔者喜欢用 Putty，只是因为它的小巧以及非常漂亮的颜色显示。不管你使用哪一个客户端软件，最终的目的只有一个，就是远程登录到 linux 服务器上。这些软件网上有很多免费版的，你可以下载一个试着玩玩。下面笔者介绍如何使用Putty登录远程linux服务器。

如果你下载了putty，请双击putty.exe 然后弹出如下的窗口。笔者所用putty为英文版的，如果你觉得英文的用着别扭，可以下载一个中文版的。



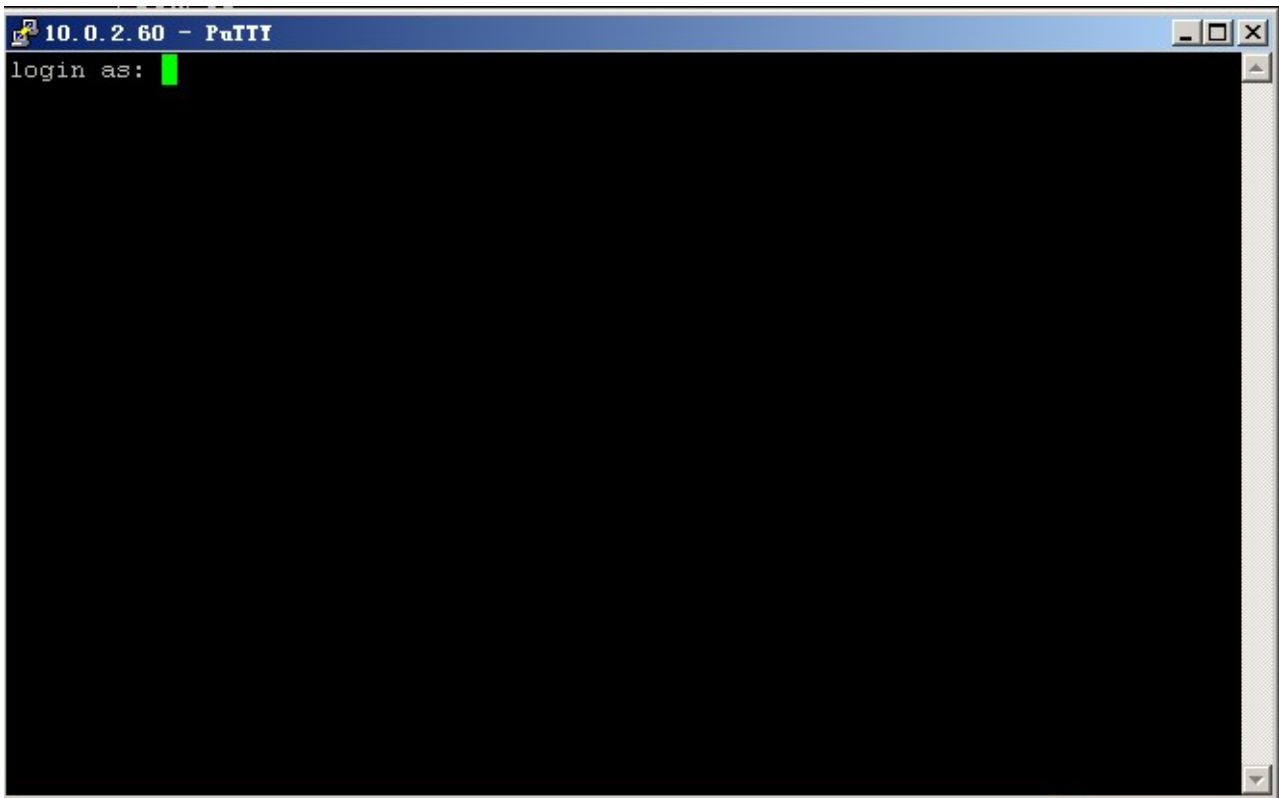
图片 5.15_1.png.jpg

因为是远程登录，所以你要登录的服务器一定会有一个IP或者主机名。请在Host Name(or IP address) 下面的框中输入你要登录的远程服务器IP(如果你的linux还没有IP，那么请自行设置一个IP，如何设置请到后续章节查找)，然后回车。



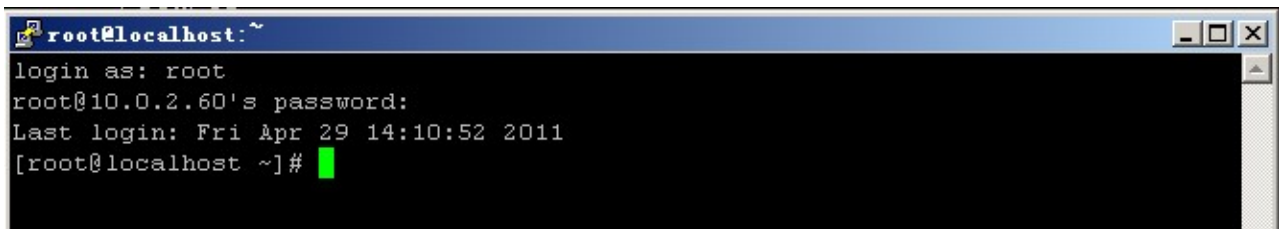
图片 5.2 5_12.png.jpg

此时，提示我们输入要登录的用户名。



图片 5.3 5_13.png.jpg

输入root 然后回车，再输入密码，就能登录到远程的linux系统了。

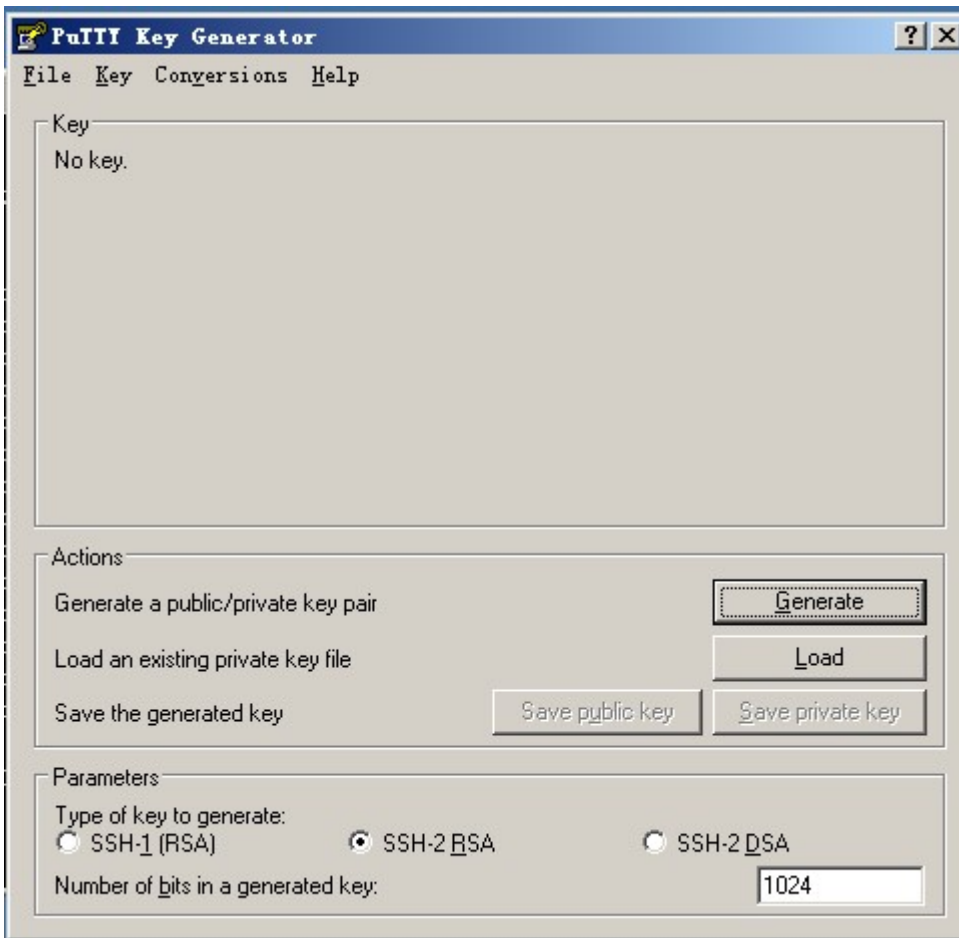


图片 5.4 5_14.png.jpg

使用密钥认证机制远程登录 linux

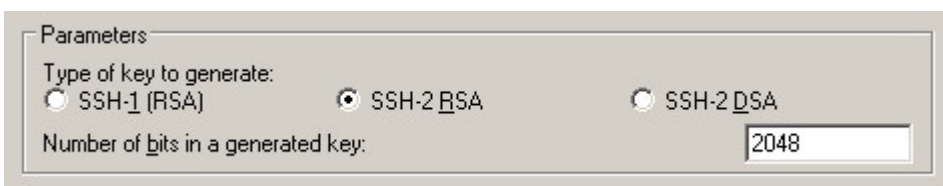
SSH服务支持一种安全认证机制，即密钥认证。所谓的密钥认证，实际上是使用一对加密字符串，一个称为公钥(public key)，任何人都可以看到其内容，用于加密；另一个称为密钥(private key)，只有拥有者才能看到，用于解密。通过公钥加密过的密文使用密钥可以轻松解密，但根据公钥来猜测密钥却十分困难。ssh 的密钥认证就是使用了这一特性。服务器和客户端都各自拥有自己的公钥和密钥。如何使用密钥认证登录linux服务器呢？

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：



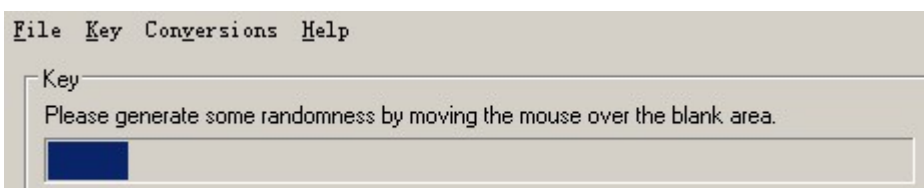
图片 5.5 5_15.png.jpg

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。



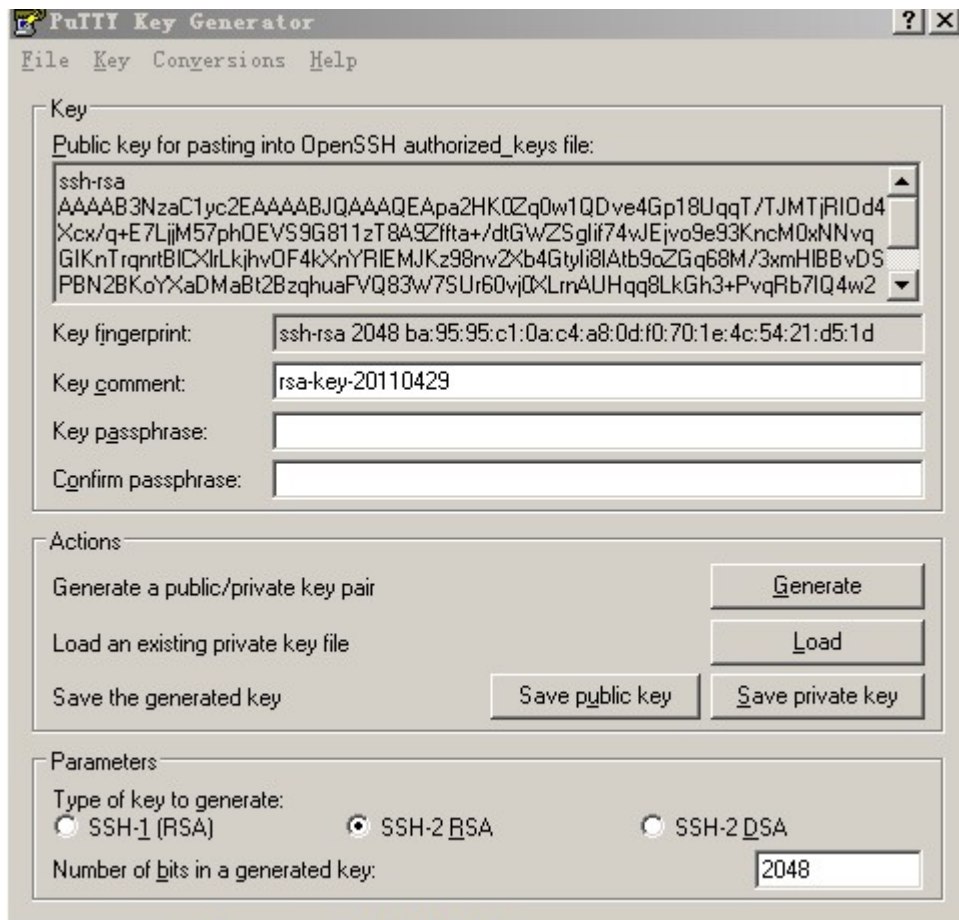
图片 5.6 5_16.png.jpg

然后单击Generate 开始生成密钥对：



图片 5.7 5_17.png.jpg

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。



图片 5.8 5_18.png.jpg

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。然后点击 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

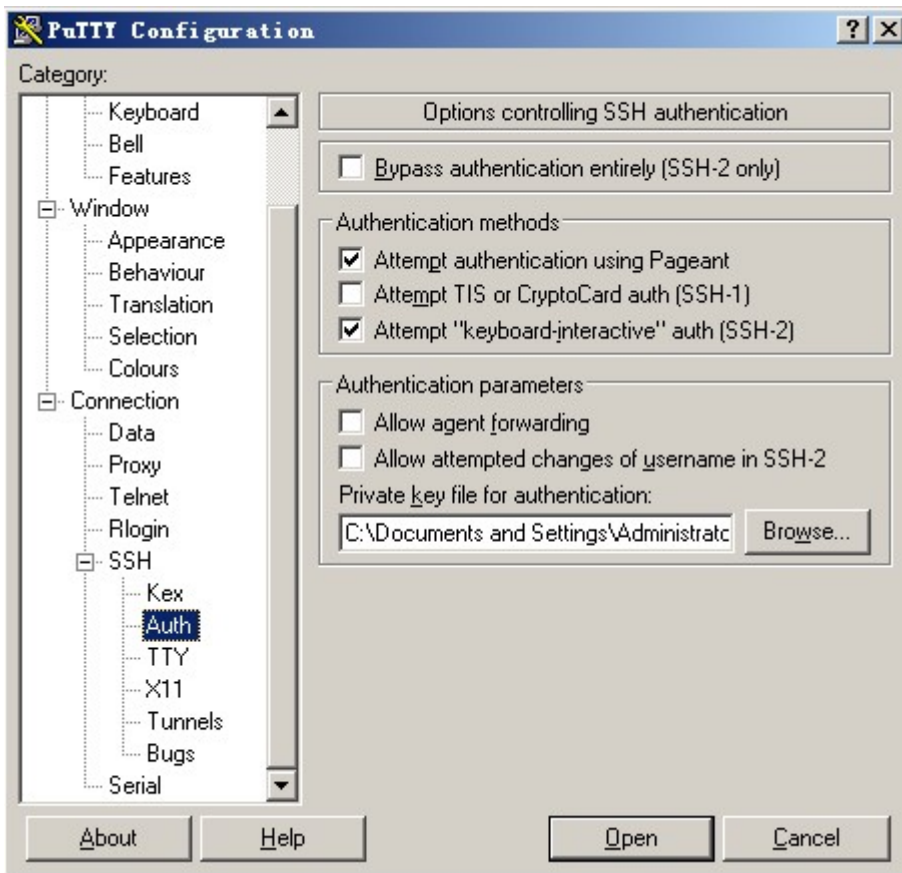
[root@localhost ~]# mkdir /root/.ssh mkdir 命令用来创建目录，以后会详细介绍，暂时只了解即可。

[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

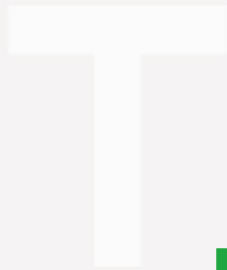
[root@localhost ~]# vim /root/.ssh/authorized_keys vim 命令是编辑一个文本文件的命令，同样在后续章节详细介绍。

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 “---- E ND 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



图片 5.10 5_20.png.jpg

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 文件与目录管理



在 linux 中什么是一个文件的路径呢，说白了就是这个文件存在的地方，例如在上一章提到的/root/.ssh/authorized_keys 这就是一个文件的路径。如果你告诉系统这个文件的路径，那么系统就可以找到这个文件。在 linux 的世界中，存在着绝对路径和相对路径。

绝对路径：路径的写法一定由根目录“/”写起，例如/usr/local/mysql 这就是绝对路径。

相对路径：路径的写法不是由根目录“/”写起，例如，首先用户进入到/然后再进入到 home，命令为 cd /home 然后 cd test 此时用户所在的路径为 /home/test。第一个cd命令后跟 /home 第二个 cd 命令后跟 test，并没有斜杠，这个test是相对于/home 目录来讲的，所以叫做相对路径。

pwd这个命令打印出当前所在目录



```

root@localhost: ~
[root@localhost ~]# pwd
/root
[root@localhost ~]#

```

图片 6.1 6_1.png.jpg

cd 进入到某一个目录



```

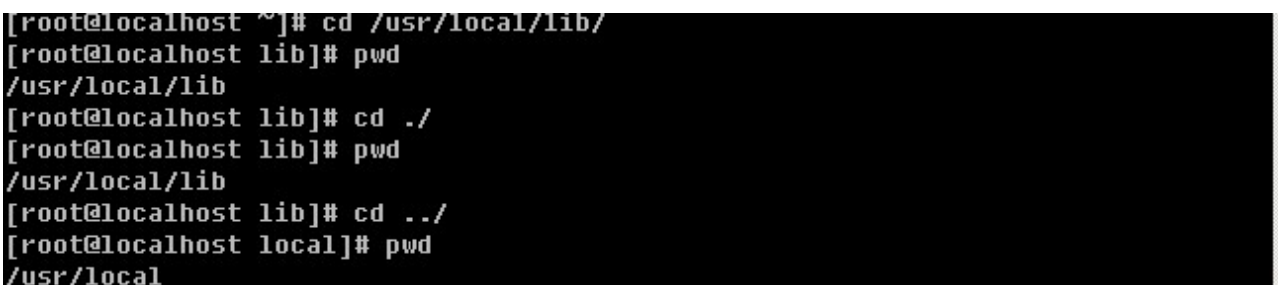
[root@localhost ~]# cd /usr/local/
[root@localhost local]# pwd
/usr/local

```

图片 6.2 6_12.png.jpg

./ 指的是当前目录

../ 指的是当前目录的上一级目录。



```

[root@localhost ~]# cd /usr/local/lib/
[root@localhost lib]# pwd
/usr/local/lib
[root@localhost lib]# cd ./
[root@localhost lib]# pwd
/usr/local/lib
[root@localhost lib]# cd ../
[root@localhost local]# pwd
/usr/local

```

图片 6.3 6_13.png.jpg

上图中，首先进入到/usr/local/lib/ 目录下，然后再进入 ./ 其实还是进入到当前目录下，用pwd查看当前目录，并没有发生变化，然后再进入../ 则是进入到了/usr/local/目录下，即/usr/local/lib目录的上一级目录。你看明白了吗？

mkdir创建一个目录，这个命令在上一章节中提及过。mkdir 其实就是make directory的缩写。其语法为 mkdir [-mp] [目录名称]，其中-m，-p 为其选项，-m：这个参数用来指定要创建目录的权限，该参数不常用，所以笔者不做重点解释。-p：这个参数很管用的，先来做个试验，你会一目了然的。


```
[root@localhost ~]# mkdir /tmp/test/123
mkdir: cannot create directory `/tmp/test/123': No such file or directory
[root@localhost ~]# ls /tmp/test
ls: /tmp/test: No such file or directory
```

图片 6.4 6_14.png.jpg

当我们想创建 /tmp/test/123 目录，可是提示不能创建，原因是/tmp/test目录不存在，你会说，这个linux怎么这样傻，/tmp/test目录不存在就自动创建不就OK了嘛，的确linux确实很傻，如果它发现要创建的目录的上一级目录不存在就会报错。然后linux也为我们想好了解决办法，即-p参数。

```
[root@localhost ~]# ls /tmp/test
ls: /tmp/test: No such file or directory
[root@localhost ~]# mkdir -p /tmp/test/123
[root@localhost ~]# ls /tmp/test
123
```

图片 6.5 6_15.png.jpg

你看到这里，是不是明白-p参数的作用了？没错，它的作用就是递归创建目录，即使上级目录不存在。还有一种情况就是如果你想要创建的目录存在的话，会提示报错，然后你加上-p参数后，就不会报错了。

```
[root@localhost ~]# ls /tmp/test
123
[root@localhost ~]# mkdir /tmp/test/123/
mkdir: cannot create directory `/tmp/test/123/': File exists
[root@localhost ~]# mkdir -p /tmp/test/123/
[root@localhost ~]#
```

图片 6.6 6_16.png.jpg

rmdir删除一个目录。

```
[root@localhost ~]# ls /tmp/test/
123
[root@localhost ~]# rmdir /tmp/test/123/
[root@localhost ~]# ls /tmp/test/
[root@localhost ~]#
```

图片 6.7 6_17.png.jpg

rmdir 其实是remove directory 缩写，其只有一个选项-p 类似与mkdir命令，这个参数的作用是将上级目录一起删除。举个例子吧，新建目录mkdir -p d1/d2/d3，rmdir -p d1/d2/d3相当于是删除了d1,d1/d2, d1/d2/d3。如果一个目录中还有目录，那么当你直接rmdir 该目录时，会提示该目录不为空，不能删除。如果你非要删除不为空的目录，那你用rm指令吧。

rm删除目录或者文件

rmdir 只能删除目录但不能删除文件，要想删除一个文件，则要用rm命令了。rm同样也有很多选项。你可以通过man rm 来获得详细帮助信息。在这里笔者只列举较常用的几个选项。

-f 强制的意思，如果不加这个选项，当删除一个不存在的文件时会报错。

```
[root@localhost ~]# ls /tmp/111
ls: /tmp/111: No such file or directory
[root@localhost ~]# /bin/rm /tmp/111
/bin/rm: cannot remove `/tmp/111': No such file or directory
[root@localhost ~]# /bin/rm -f /tmp/111
[root@localhost ~]#
```

图片 6.8 6_18.png.jpg

-i 这个选项的作用是，当用户删除一个文件时会提示用户是否真的删除。

```
[root@localhost ~]# /bin/rm -i /root/install.log
/bin/rm: remove regular file `/root/install.log'? n
[root@localhost ~]# ls /root/install.log
/root/install.log
```

图片 6.9 6_19.png.jpg

如果删除，输入y 否则输入 n

-r 当删除目录时，加该选项，如果不加这个选项会报错。rm是可以删除不为空的目录的。

```
[root@localhost ~]# ls /tmp/test/
111
[root@localhost ~]# /bin/rm /tmp/test
/bin/rm: cannot remove `/tmp/test': Is a directory
[root@localhost ~]# /bin/rm -r /tmp/test
[root@localhost ~]# ls /tmp/test
ls: /tmp/test: No such file or directory
```

图片 6.10 6_20.png.jpg

你会发现，笔者在列举的rm例子中使用的是绝对路径，而ls 则使用的相对路径。这是为什么呢？

```
[root@localhost ~]# which rm
alias rm='rm -i'
      /bin/rm
```

图片 6.11 6_21.png.jpg

which 用来查找一个命令的绝对路径，这个命令笔者不详细介绍，因为平时笔者只用来查找一个命令的绝对路径。

alias用来设置指令的别名。语法：alias[别名]=[指令名称]，例如 alias rm='rm -i'，即当我们使用rm命令时，实际上是使用的是rm -i，而用绝对路径的/bin/rm 则不会被alias，该命令在以后章节中会详细介绍。

环境变量 PATH

上边提到了alias，也提到了绝对路径的/bin/rm，然后你意识到没有，为什么我们输入很多命令时是直接打出了命令，而没有去使用这些命令的绝对路径？这是因为环境变量PATH在起作用了。请输入 echo \$PATH，这里的echo其实就是打印的意思，而PATH前面的\$表示后面接的是变量。

```
[root@localhost ~]# echo $PATH
/usr/lib/qt-3.3/bin:/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin
[root@localhost ~]# echo nihao
nihao
```

图片 6.12 6_65.png.jpg

因为/bin 在PATH的设定中，所以自然就可以找到ls了。如果你将ls 移动到 /root 底下的话，然后你自己本身也在 /root 底下，但是当你执行ls 的时候，他就是不理你？怎么办？这是因为 PATH 没有 /root 这个目录，而你又将ls 移动到 /root 底下了，自然系统就找不到可执行文件了，因此就会告诉你，command not found！那么该怎么克服这种问题呢？

有两个方法，一种方法是直接将 /root 的路径加入 PATH 当中！如何增加？可以使用：

```
PATH=" $PATH" :/root
```

另一种方式则是使用完整档名，亦即直接使用相对或绝对路径来执行，例如：

```
/root/ls
```

```
./ls
```

关于rm，笔者使用最多便是-rf两个选项合用了。不管删除文件还是目录都可以。但是方便的同时也要多注意，万一你的手太快后边跟了/那样就会把你的系统文件全部删除的，切记切记。

```
[root@localhost test]# mkdir -p d1/d2/d3
[root@localhost test]# rm -r d1
rm: descend into directory `d1'? y
rm: descend into directory `d1/d2'? y
rm: remove directory `d1/d2/d3'? y
rm: remove directory `d1/d2'? y
rm: remove directory `d1'? y
[root@localhost test]# mkdir -p d1/d2/d3
[root@localhost test]# rm -rf d1
[root@localhost test]# ls d1
ls: d1: No such file or directory
```

图片 6.13 6_66.png.jpg

ls 在前面的命令中多次用到它。现在你已经明白它的含义了吧。没有错，就是查看某个目录或者某个文件，是list的简写。ls 后可以跟一个目录，也可以跟一个文件。以下是ls的选项，在这里笔者并没有完全列出，只是列出了平时使用最多的选项。其他选项，你可以自行通过man ls 查询。

-a 全部的档案都列出，包括隐藏的。linux文件系统中同样也有隐藏文件。这些隐藏文件的文件名是以.开头的。例如.test, /root/.123, /root/.ssh 等等，隐藏文件可以是目录也可以是普通文件。

-l 详细列出文件的属性信息，包括大小、创建日期、所属主所属组等等。|| 这个命令等同于ls -l。

```
[root@localhost ~]# which ll
alias ll='ls -l --color=tty'
/bin/ls
```

图片 6.14 6_67.png.jpg

--color=never/always/auto never即不要显示颜色，always 即总显示颜色，auto 是由系统自行判断。在Red hat/CentOS 系统中，默认是带颜色的，因为我们平时用的ls已经alias成了ls - color=tty 所以目录的颜色是蓝色的，而可执行文件的颜色是绿色。这样有助于帮我们区分文件的格式。

```
[root@localhost ~]# which ls
alias ls='ls --color=tty'
/bin/ls
```

图片 6.15 6_68.png.jpg

-d 后边跟目录，如果不加这个选项则列出目录下的文件，加上后只列车目录本身。

```
[root@localhost ~]# ls -d /root/
/root/
[root@localhost ~]# ls /root
anaconda-ks.cfg  install.log  install.log.syslog  test
```

图片 6.16 6_69.png.jpg

cp copy的简写，即拷贝。格式为 cp [选项] [来源文件] [目的文件]，例如我想把test1 拷贝成test2，这样即可 cp test1 test2，以下介绍几个常用的选项

-d 这里涉及到一个“连接”的概念。连接分为软连接和硬连接。在以后的章节中会详细解释，现在你只要明白这里的软连接跟windows中的快捷方式类似即可。如果不加这个-d 则拷贝软连接时会把软连接的目标文件拷贝过去，而加上后，其实只是拷贝了一个连接文件（即快捷方式）。

```

[root@localhost test]# touch test
[root@localhost test]# ln -s test test1
[root@localhost test]# ls -l test test1
-rw-r--r-- 1 root root 0 May 12 03:36 test
lrwxrwxrwx 1 root root 4 May 12 03:36 test1 -> test
[root@localhost test]# cp test1 test2
[root@localhost test]# ls -l test test1 test2
-rw-r--r-- 1 root root 0 May 12 03:36 test
lrwxrwxrwx 1 root root 4 May 12 03:36 test1 -> test
-rw-r--r-- 1 root root 0 May 12 03:36 test2
[root@localhost test]# cp -d test1 test3
[root@localhost test]# ls -l test test1 test2 test3
-rw-r--r-- 1 root root 0 May 12 03:36 test
lrwxrwxrwx 1 root root 4 May 12 03:36 test1 -> test
-rw-r--r-- 1 root root 0 May 12 03:36 test2
lrwxrwxrwx 1 root root 4 May 12 03:36 test3 -> test

```

图片 6.17 6_70.png.jpg

上例中的ln 命令即为建立连接的，以后再做详细解释。

-r 如果你要拷贝一个目录，必须要加-r选项，否则你是拷贝不了目录的。

```

[root@localhost ~]# mkdir 123
[root@localhost ~]# ls
123 anaconda-ks.cfg install.log install.log.syslog
[root@localhost ~]# cp 123 1234
cp: omitting directory `123'
[root@localhost ~]# cp -r 123 1234
[root@localhost ~]# ls
123 1234 anaconda-ks.cfg install.log install.log.syslog

```

图片 6.18 6_71.png.jpg

-i 如果遇到一个存在的文件，会问是否覆盖。在Redhat/CentOS系统中，我们使用的cp其实是cp -i

```

[root@localhost ~]# which cp
alias cp='cp -i'
/bin/cp

```

图片 6.19 6_72.png.jpg

下面简单做一个小试验，很快你就会明白-i 选项的作用了。

```

[root@localhost 123]# ls
[root@localhost 123]# touch 111
[root@localhost 123]# touch 222
[root@localhost 123]# cp -i 111 222
cp: overwrite `222'? n
[root@localhost 123]# echo "abc" >111
[root@localhost 123]# echo "def" >222
[root@localhost 123]# cat 111
abc
[root@localhost 123]# cat 222
def
[root@localhost 123]# /bin/cp 111 222
[root@localhost 123]# cat 222
abc

```

图片 6.20 6_73.png.jpg

上例中，touch 命令，看字面意思就是摸一下，没错，如果有这个文件，则会改变文件的访问时间，如果没有这个文件就会创建这个文件。前面说过echo，其实就是打印，在这里所echo的内容“abc”和“def”并没有显示在屏幕上，而是分别写进了文件 111和222，其写入作用的就是这个大于号“>”在linux中这叫做重定向，即把前面产生的输出写入到后面的文件中。在以后的章节中会做详细介绍，这里你要明白它的含义即可。而cat 命令则是读一个文件，并把读出的内容打印到当前屏幕上。该命令也会在后续章节中详细介绍。

-u 该选项仅当目标文件存在时才会生效，如果源文件比目标文件新才会拷贝，否则不做任何动作。

mv 移动的意思，是move的简写。格式为 mv [选项] [源文件] [目标文件]，下面介绍几个常用的选项。

-i 和cp的-i 一样，当目标文件存在时会问用户是否要覆盖。在Redhat/CentOS系统中，我们使用的mv其实是 mv -i

-u 和上边cp 命令的-u选项一个作用，当目标文件存在时才会生效，如果源文件比目标文件新才会移动，否则不做任何动作。

该命令有集中情况，你注意到了吗？

- 1) 目标文件是目录，而且目标文件不存在；
- 2) 目标文件是目录，而且目标文件存在；
- 3) 目标文件不是目录不存在；
- 4) 目标文件不是目录存在；

目标文件是目录，存在和不存在，移动的结果是不一样的，如果存在，则会把源文件移动到目标文件目录中。不存在的话移动完后，目标文件是一个文件。这样说也许你会觉得有点不好理解，看例子吧。

```

[root@localhost ~]# mkdir aa bb
[root@localhost ~]# ls
aa anaconda-ks.cfg bb install.log install.log.syslog
[root@localhost ~]# mv aa cc
[root@localhost ~]# ls
anaconda-ks.cfg bb cc install.log install.log.syslog
[root@localhost ~]# mv cc bb
[root@localhost ~]# ls
anaconda-ks.cfg bb install.log install.log.syslog
[root@localhost ~]# ls bb
cc
[root@localhost ~]# touch dd
[root@localhost ~]# ls
anaconda-ks.cfg bb dd install.log install.log.syslog
[root@localhost ~]# mv dd ee
[root@localhost ~]# ls
anaconda-ks.cfg bb ee install.log install.log.syslog
[root@localhost ~]# mv ee bb
[root@localhost ~]# ls bb
cc ee

```

图片 6.21 6_74.png.jpg

windows下的重命名，在linux下用mv就可以搞定。

cat 比较常用的一个命令，即查看一个文件的内容并显示在屏幕上。

-n 查看文件时，把行号也显示到屏幕上。

```

[root@localhost ~]# echo "123123 " >bb/ee
[root@localhost ~]# echo "456456 " >>bb/ee
[root@localhost ~]# cat bb/ee
123123
456456
[root@localhost ~]# cat -n bb/ee
 1 123123
 2 456456

```

图片 6.22 6_75.png.jpg

上例中出现了一个”>>”，这个符号跟前面介绍的”>”的作用都是重定向，即把前面输出的东西输入到后边的文件中，只是”>>”是追加的意思，而用”>”，如果文件中有内容则会删除文件中内容，而”>>”则不会。

-A 显示所有东西出来，包括特殊字符

```

[root@localhost ~]# cat -A bb/ee
123123 $
456456 $

```

图片 6.23 6_76.png.jpg

tac其实是cat的反写，同样的功能也是反向打印文件的内容到屏幕上。

```
[root@localhost ~]# tac bb/ee
456456
123123
```

图片 6.24 6_77.png.jpg

more也是用来查看一个文件的内容。当文件内容太多，一屏幕不能占下，而你用cat肯定是看不前面的内容的，那么使用more就可以解决这个问题了。当看完一屏后按空格键继续看下一屏。但看完所有内容后就会退出。如果你想提前退出，只需按q键即可。

less作用跟more一样，但比more好在可以上翻，下翻。空格键同样可以翻页，而按”j”键可以向下移动（按一下就向下移动一行），按”k”键向上移动。在使用more和less查看某个文件时，你可以按一下”/”键，然后输入一个word回车，这样就可以查找这个word了。如果是多个该word可以按”n”键显示下一个。另外你也可以不按”/”而是按”?”后边同样跟word来搜索这个word，唯一不同的是，”/”是在当前行向下搜索，而”?”是在当前行向上搜索。

head head后直接跟文件名，则显示文件的前十行。如果加 -n 选项则显示文件前n行。

```
[root@localhost ~]# head /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
[root@localhost ~]# head -n 5 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

图片 6.25 6_78.png.jpg

tail 和head一样，后面直接跟文件名，则显示文件最后十行。如果加-n 选项则显示文件最后n行。

```
[root@localhost ~]# tail /etc/passwd
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
ucsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
oprofile:x:16:16:Special user account to be used by OProfile:/home/oprofile:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
exim:x:93:93:/:/var/spool/exim:/sbin/nologin
avahi:x:70:70:Avahi daemon:/:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/ssh:/sbin/nologin
xfs:x:43:43:X Font Server:/etc/X11/fs:/sbin/nologin
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
[root@localhost ~]# tail -n 2 /etc/passwd
xfs:x:43:43:X Font Server:/etc/X11/fs:/sbin/nologin
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
```


图片 6.26 6_79.png.jpg

-f 动态显示文件的最后十行，如果文件是不断增加的，则用 -f 选项。如：tail -f /var/log/messages

文件的所属主以及所属组

一个linux目录或者文件，都会有一个所属主和所属组。所属主，即文件的拥有者，而所属组，即该文件所属主所在的一个组。Linux这样设置文件属性的目的是为了文件的安全。例如，test文件的所属主是user0 而test1文件的所属主是user1，那么user1是不能查看test文件的，相应的user0也不能查看test1文件。然后有这样一个应用，我想创建一个文件同时让user0和user1来查看怎么办呢？

这时“所属组”就派上用场了。即，创建一个群组users，让user0和user1同属于users组，然后建立一个文件test2，且其所属组为users，那么user0和user1都可以访问test2文件。

Linux文件属性不仅规定了所属主和所属组，还规定了所属主（user）、所属组(group)以及其他用户（others）对该文件的权限。你可以通过ls -l 来查看这些属性。

```
[root@localhost ~]# ls -l
total 64
-rw----- 1 root root  904 Apr 27 00:44 anaconda-ks.cfg
drwxr-xr-x 2 root root 4096 May  3 13:17 bb
drwxr-xr-x 3 root root 4096 May  3 13:17 ff
-rw-r--r-- 1 root root 25685 Apr 27 00:44 install.log
-rw-r--r-- 1 root root  2575 Apr 27 00:37 install.log.syslog
```

图片 6.27 6_80.png.jpg

linux 文件属性

上例中，用ls -l 查看当前目录下的文件时，共显示了9列内容（用空格划分列），都代表了什么含义呢？

第1列，包含的东西有该文件类型和所属主、所属组以及其他用户对该文件的权限。第一列共10位。其中第一位用来描述该文件的类型。上例中，我们看到的类型有”d”，“-“，其实除了这两种外还有”l”，”b”，”c”，”s”等。

d 表示该文件为目录；

- 表示该文件为普通文件；

l 表示该文件为连接文件（linux file），上边提到的软连接即为该类型；

```
[root@localhost ~]# ls -l /etc/rc.local
lrwxrwxrwx 1 root root 13 Apr 27 00:31 /etc/rc.local -> rc.d/rc.local
```

图片 6.28 6_81.png.jpg

b 表示该文件为块设备文件，比如磁盘分区

```
[root@localhost ~]# ls -l /dev/hda*
brw-r----- 1 root disk 3, 0 May  3 11:48 /dev/hda
brw-r----- 1 root disk 3, 1 May  3 11:49 /dev/hda1
brw-r----- 1 root disk 3, 2 May  3 11:48 /dev/hda2
brw-r----- 1 root disk 3, 3 May  3 11:49 /dev/hda3
```

图片 6.29 6_82.png.jpg

c 表示该文件为串行端口设备，例如键盘、鼠标。

s 表示该文件为套接字文件（socket），用于进程间通信。

后边的9位，每三个为一组。均为rwx三个参数的组合。其中r代表可读，w代表可写，x代表可执行。前三位为所属主（user）的权限，中间三位为所属组（group）的权限，最后三位为其他非本群组（others）的权限。下面拿一个具体的例子来述说一下。

一个文件的属性为-rwxr-xr--，它代表的意思是，该文件为普通文件，文件拥有者可读可写可执行，文件所属组对其可读不可写可执行，其他用户对其只可读。

对于一个目录来讲，打开这个目录即为执行这个目录，所以任何一个目录必须要有x权限才能打开并查看该目录。例如一个目录的属性为drwxr--r-- 其所属主为root，那么除了root外的其他用户是不能打开这个目录的。

第2列，表示为连接占用的节点（inode），若为目录时，通常与该目录地下还有多少目录有关系，关于连接（link）在以后章节详细介绍。

第3列，表示该文件的所属主。

第4列，表示该文件的所属组。

第5列，表示该文件的大小。

第6列、第7列和第8列为该文件的创建日期或者最近的修改日期，分别为月份日期以及时间。

第9列，文件名。如果前面有一个.则表示该文件为隐藏文件。

更改文件的权限

更改文件的权限，也就是更改所属主、所属组以及他们对应的读写执行权限。

1) 更改所属组 chgrp

语法: chgrp [组名] [文件名]

```
[root@localhost ~]# groupadd testgroup
[root@localhost ~]# touch test1
[root@localhost ~]# ls -l test1
-rw-r--r-- 1 root root 0 May  3 16:19 test1
[root@localhost ~]# chgrp testgroup test1
[root@localhost ~]# ls -l test1
-rw-r--r-- 1 root testgroup 0 May  3 16:19 test1
```

图片 6.30 6_83.png.jpg

这里用到了groupadd 命令，其含义即增加一个用户组。该命令在以后章节中做详细介绍，你只要知道它是用来增加用户组的即可。

2) 更改文件的所属主 chown

语法: chown [-R] 账户名文件名

chown [-R] 账户名: 组名文件名

这里的-R选项只作用于目录，作用是级联更改，即不仅更改当前目录，连目录里的目录或者文件全部更改。

```
[root@localhost ~]# mkdir test
[root@localhost ~]# useradd user1
[root@localhost ~]# ls -ld test
drwxr-xr-x 2 root root 4096 May  4 08:10 test
[root@localhost ~]# touch test/test2
[root@localhost ~]# ls -l test
total 4
-rw-r--r-- 1 root root 0 May  4 08:10 test2
[root@localhost ~]# chown user1 test
[root@localhost ~]# ls -ld test
drwxr-xr-x 2 user1 root 4096 May  4 08:10 test
[root@localhost ~]# ls -l test
total 4
-rw-r--r-- 1 root root 0 May  4 08:10 test2
[root@localhost ~]# chown -R user1:testgroup test
[root@localhost ~]# ls -ld test
drwxr-xr-x 2 user1 testgroup 4096 May  4 08:10 test
[root@localhost ~]# ls -l test
total 4
-rw-r--r-- 1 user1 testgroup 0 May  4 08:10 test2
```

图片 6.31 6_84.png.jpg

useradd 是增加一个账户，以后会详细介绍。上例中，首先建立一个目录test，然后在test目录下创建一个普通文件test2，因为是以root的身份创建的目录和文件，所以所属主以及所属组都是root。chown user1 test 这使test的目录所属主由root变为了user1，然后test目录下的test2文件所属主以及所属组还是root。接着 chown -R user1:testgroup test 这样把test连同test目录下的test2 的所属主以及所属组都改变了。

3) 改变用户对文件的读写执行权限 chmod

在linux中为了方便更改这些权限，linux使用数字去代替rwx，具体规则为：r: 4 w:2 x:1 -:0 举个例子，-rwxrwx--用数字表示就是 770，具体是这样来的：

$rwx = 4+2+1=7$; $rwx = 4+2+1=7$; $--- = 0+0+0=0$

chmod 语法：chmod [-R] xyz 文件名（这里的xyz，表示数字）

-R 选项作用同chown，级联更改。

值得提一下的是，在linux系统中，默认一个目录的权限为 755，而一个文件的默认权限为644。

```
[root@localhost ~]# ls -ld test
drwxr-xr-x 2 user1 testgroup 4096 May  4 08:10 test
[root@localhost ~]# ls -l test
total 4
-rw-r--r-- 1 user1 testgroup 0 May  4 08:10 test2
[root@localhost ~]# chmod 750 test
[root@localhost ~]# ls -ld test
drwxr-x--- 2 user1 testgroup 4096 May  4 08:10 test
[root@localhost ~]# ls -l test/test2
-rw-r--r-- 1 user1 testgroup 0 May  4 08:10 test/test2
[root@localhost ~]# chmod -R 700 test
[root@localhost ~]# ls -ld test
drwx----- 2 user1 testgroup 4096 May  4 08:10 test
[root@localhost ~]# ls -l test
total 4
-rwx----- 1 user1 testgroup 0 May  4 08:10 test2
```

图片 6.32 6_85.png.jpg

如果你创建了一个目录，而该目录不想让其他人看到内容，则只需设置成 rwxr----- (740) 即可。

chmod 还支持使用rwx的方式来设置权限。！从之前的介绍中我们可以发现，基本上就九个属性分别是(1)user (2)group (3)others 三群啦！那么我们就可以藉由 u, g, o 来代表三群的属性！此外，a 则代表 all 亦即全部的三群！那么读写的属性就可以写成了 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	档案或目录
-------	------------------	-------------------------	-------------	-------

图片 6.33 6_86.png.jpg

现在我想把一个文件设置成这样的权限 rwxr-xr-x (755)，使用这种方式改变权限的命令为

```
[root@localhost ~]# chmod u=rwx,og=rx test/test2
[root@localhost ~]# ls -l test/test2
-rwxr-xr-x 1 user1 testgroup 0 May  4 08:10 test/test2
```

图片 6.34 6_87.png.jpg

另外还可以针对u, g, o, a增加或者减少某个权限（读，写，执行），例如

```
[root@localhost ~]# chmod u-x test/test2
[root@localhost ~]# ls -l test
total 4
-rw-r-xr-x 1 user1 testgroup 0 May  4 08:10 test2
[root@localhost ~]# chmod a-x test/test2
[root@localhost ~]# ls -l test
total 4
-rw-r--r-- 1 user1 testgroup 0 May  4 08:10 test2
[root@localhost ~]# chmod u+x test/test2
[root@localhost ~]# ls -l test
total 4
-rwxr--r-- 1 user1 testgroup 0 May  4 08:10 test2
```

图片 6.35 6_88.png.jpg

另外linux下还有两个比较特殊的权限s和t，请点击[linux下文件的特殊权限s和t \(\)](#)

umask

上边也提到了默认情况下，目录权限值为766，普通文件权限值为644。那么这个值是由谁规定呢？追究其原因就涉及到了umask。

umask语法：umask xxx（这里的xxx代表三个数字）

查看umask值只要输入umask然后回车。umask预设是0022，其代表什么含义？先看一下下面的规则：

- 1) 若用户建立为普通文件，则预设“没有可执行权限”，只有rw两个权限。最大为666（-rw-rw-rw-）
- 2) 若用户建立为目录，则预设所有权限均开放，即777（drwxrwxrwx）

umask数值代表的含义为，上边两条规则中的默认值（文件为666，目录为777）需要减掉的权限。所以目录的权限为(rwxrwxrwx) - (----w--w-) = (rwxr-xr-x)，普通文件的权限为(rw-rw-rw-) - (----w--w-) = (rw-r--r--)。umask的值是可以自定义的，比如设定umask为002，你再创建目录或者文件时，默认权限分别为(rwxrwxrwx) - (-----w-) = (rwxrwxr-x)和(rw-rw-rw-) - (-----w-) = (rw-rw-r--)。

```
[root@localhost ~]# umask
0022
[root@localhost ~]# umask 002
[root@localhost ~]# mkdir test3
[root@localhost ~]# ll -d test3
drwxrwxr-x 2 root root 4096 May  4 13:24 test3
[root@localhost ~]# touch test4
[root@localhost ~]# ll test4
-rw-rw-r-- 1 root root 0 May  4 13:24 test4
```

图片 6.36 6_89.png.jpg

umask 可以在/etc/bashrc里面更改，预设情况下，root的umask为022，而一般使用者则为002，因为可写的权限非常重要，因此预设会去掉写权限。

chattr 修改文件的特殊属性

语法：chattr [+ -=][ASaci [文件或者目录名]

+ -=：分别为增加、减少、设定

A：增加该属性后，文件或目录的atime将不可被修改；

S：增加该属性后，会将数据同步写入磁盘中；

a：增加该属性后，只能追加不能删除，非root用户不能设定该属性；

c：自动压缩该文件，读取时会自动解压；

i：增加后，使文件不能被删除、重命名、设定连接、写入、新增数据；

```
[root@localhost ~]# chattr +i test3
[root@localhost ~]# touch test3/test1
touch: cannot touch `test3/test1': Permission denied
[root@localhost ~]# chattr -i test3
[root@localhost ~]# touch test3/test1
[root@localhost ~]# ls test3
test1
```

图片 6.37 6_90.png.jpg

增加i属性后不能在该目录中建立文件。

```
[root@localhost ~]# touch test3/test2
[root@localhost ~]# ls test3
test1 test2
[root@localhost ~]# chattr +a test3
[root@localhost ~]# rm -f test3/test1
rm: cannot remove `test3/test1': Operation not permitted
[root@localhost ~]# touch test3/test4
[root@localhost ~]# ls test3
test1 test2 test4
[root@localhost ~]# chattr -a test3
[root@localhost ~]# rm -f test3/test1
[root@localhost ~]# ls test3
test2 test4
```

图片 6.38 6_91.png.jpg

增加a属性后，只能追加不能删除。

lsattr 列出文件/目录的特殊属性

语法: lsattr [-aR] [文件/目录名]

-a: 类似与ls 的-a 选项, 即连同隐藏文件一同列出;

-R: 连同子目录的数据一同列出

```
[root@localhost ~]# lsattr -aR test3
----- test3/..
----- test3/.
----- test3/test2
----a----- test3/test4
[root@localhost ~]# lsattr -R test3
----- test3/test2
----a----- test3/test4
[root@localhost ~]# lsattr test3
----- test3/test2
----a----- test3/test4
```

图片 6.39 6_92.png.jpg

在上例中, test4是在test3目录增加a属性后建立的, 所以test4也有a属性, 通过这个例子可以看出, chattr 的属性是级联生效的, 不仅对当前目录生效而且会对目录下的文件同样生效。

在 linux 下搜索一个文件

在windows下有一个搜索工具, 可以让我们很快的找到一个文件, 这是很有用的。然而在linux下搜索功能更加强大。

which用来查找可执行文件的绝对路径。

在前面章节中已经多次用到该命令, 需要注意的一点是, which只能用来查找PATH环境变量中出现的的路径下的可执行文件。这个命令用的也是蛮多的, 有时候我们不知道某个命令的绝对路径, which一下很容易就知道了。

```
[root@localhost ~]# which ls
alias ls='ls --color=tty'
      /bin/ls
[root@localhost ~]# which cd
/usr/bin/which: no cd in (/usr/lib/qt-3.3/bin:/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin)
```

图片 6.40 6_93.png.jpg

当查找的文件在PATH变量中并没有时, 就会报错。

whereis 通过预先生成的一个文件列表库去查找跟给出的文件名相关的文件。

语法: whereis [-bmsu] [文件名称]

-b: 只找binary 文件

-m: 只找在说明文件manual路径下的文件

-s: 只找source来源文件

-u: 没有说明档的文件

```
[root@localhost ~]# whereis passwd
passwd: /usr/bin/passwd /etc/passwd /usr/share/man/man1/passwd.1.gz
[root@localhost ~]# whereis -b passwd
passwd: /usr/bin/passwd /etc/passwd
[root@localhost ~]# whereis -m passwd
passwd: /usr/share/man/man1/passwd.1.gz
```

图片 6.41 6_94.png.jpg

说明: whereis 笔者几乎很少用到, 如果你感兴趣请深入研究。

locate类似于whereis, 也是通过查找预先生成的文件列表库来告诉用户要查找的文件在哪里。后边直接跟文件名。如果你的linux没有这个命令, 请安装软件包 mlocate, 这个软件包在你的系统安装盘里, 后缀名是RPM, 随后介绍的find命令会告诉你如何查找这个包。如果你装的CentOS你可以使用这个命令来安装 yum install -y mlocate。前提是你的CentOS能连互联网。至于yum这个命令如何使用, 到后续章节你自然会明白。如果你刚装上这个命令, 初次使用会报错。

```
[root@localhost ~]# locate passwd
locate: can not open `/var/lib/mlocate/mlocate.db': No such file or directory
```

图片 6.42 6_95.png.jpg

这是因为系统还没有生成那个文件列表库。你可以使用updatedb 命令立即生成(更新)这个库。如果你的服务器上正跑着重要的业务, 那么你最好不要去运行这个命令, 因为一旦运行, 服务器的压力会变大。这个数据库默认情况下每周更新一次。所以你用locate命令去搜索一个文件, 正好是在两次更新时间段内, 那你肯定是得不到结果的。你可以到/etc/updated.conf 去配置这个数据库生成(更新)的规则。locate命令笔者用的也并不多, 所以你只要明白有这么一个东西即可。你用到时再去深究其用法吧。

find 这个搜索工具是笔者用的最多的一个, 所以请你务必要熟悉它。

语法: find [路径] [参数] 下面介绍几个笔者经常用的参数

-atime +n : 访问或执行时间大于n天的文件

-ctime +n : 写入、更改inode属性(例如更改所有者、权限或者连接)时间大于n天的文件

-mtime +n : 写入时间大于n天的文件

看到这里, 你对这三个time是不是有些晕了, 那笔者就先给你介绍一下这三个time属性。

文件的 Access time, atime 是在读取文件或者执行文件时更改的。文件的 Modified time, mtime 是在写入文件时随文件内容的更改而更改的。文件的 Create time, ctime 是在写入文件、更改所有者、权限或链接设置时随 Inode 的内容更改而更改的。因此,更改文件的内容即会更改 mtime 和 ctime,但是文件的 ctime 可能会在 mtime 未发生任何变化时更改,例如,更改了文件的权限,但是文件内容没有变化。如何获得一个文件的 atime mtime 以及 ctime ?

ls -l 命令可用来列出文件的 atime、ctime 和 mtime。

ls -lc filename 列出文件的 ctime

ls -lu filename 列出文件的 atime

ls -l filename 列出文件的 mtime

atime 不一定在访问文件之后被修改,因为:使用 ext3 文件系统的时候,如果在 mount 的时候使用了 noatime 参数那么就不会更新 atime 的信息。而这是加了 noatime 取消了,不代表真实情况。反正,这三个 time stamp 都放在 inode 中。若 mtime, atime 修改 inode 就一定会改,既然 inode 改了,那 ctime 也就跟著要改了。

继续讲 find 常用的参数。

-name filename 直接查找该文件名的文件,这个使用最多了。

```
[root@localhost ~]# ls
123 456 anaconda-ks.cfg install.log install.log.syslog test3 test4 up.txt
[root@localhost ~]# find /root -name test3
/root/test3
```

图片 6.43 6_96.png.jpg

-type type : 通过文件类型查找。文件类型在前面部分已经简单介绍过,相信你已经大体上了解了。type 包含了 f, b, c, d, l, s 等等。后续的内容还会介绍文件类型的。

```
[root@localhost ~]# mkdir file1
[root@localhost ~]# mkdir file1/file2
[root@localhost ~]# touch file1/file3
[root@localhost ~]# touch file1/file2/file4
[root@localhost ~]# find ./file1 -type d
./file1
./file1/file2
[root@localhost ~]# find ./file1 -type f
./file1/file3
./file1/file2/file4
```

图片 6.44 6_97.png.jpg

linux 的文件系统

搞计算机的应该都知道windows的系统格式化硬盘时会指定格式，fat 或者 ntfs。而linux的文件系统格式为Ext 2，或者Ext3。早期的linux使用Ext2格式，目前的linux都使用了Ext3。Ext2文件系统虽然是高效稳定的。但是，随着Linux系统在关键业务中的应用，Linux文件系统的弱点也渐渐显露出来了，因为Ext2文件系统是非日志文件系统。这在关键行业的应用是一个致命的弱点。Ext3文件系统是直接由Ext2文件系统发展而来，Ext3文件系统带有日志功能，可以跟踪记录文件系统的变化，并将变化内容写入日志，写操作首先是对日志记录文件进行操作，若整个写操作由于某种原因（如系统掉电）而中断，系统重启时，会根据日志记录来恢复中断前的写操作，而且这个过程费时极短。目前Ext3文件系统已经非常稳定可靠。它完全兼容Ext2文件系统。用户可以平滑地过渡到一个日志功能健全的文件系统中来。这实际上也是ext3日志文件系统初始设计的初衷。

Linux文件系统在windows中是不能识别的，但是在linux系统中你可以挂载的windows的文件系统，linux目前支持MS-DOS，VFAT，FAT，BSD等格式。如果你使用的是Redhat或者CentOS，那么你不要妄图挂载NFS格式的文件到linux下，因为它不支持NFS。虽然有些版本的linux支持NFS，但不建议使用，因为目前的技术还不成熟。

Ext3文件系统为Redhat/CentOS默认使用的文件系统，除了Ext3文件系统外，有些linux发行版例如SuSE默认的文件系统为reiserFS，Ext3独特的优点就是易于转换，很容易在Ext2和Ext3之间相互转换，而具有良好的兼容性，其它优点ReiserFS都有，而且还比它做得更好。如高效的磁盘空间利用和独特的搜寻方式都是Ext3所不具备的，速度上它也不能和ReiserFS相媲美，在实际使用过程中，reiserFS也更加安全高效，据说反删除功能也不错。

ReiserFS的优势在于，它是基于B*Tree快速平衡树这种高效算法的文件系统，例如在处理小于1k的文件比Ext3快10倍。再一个就是ReiserFS空间浪费较少，它不会对一些小文件分配inode，而是打包存放在同一个磁盘块（簇）中，Ext2/Ext3是把它们单独存放在不同的簇上，如簇大小为4k，那么2个100字节的文件会占用2个簇，ReiserFS则只占用一个。当然ReiserFS也有缺点，就是每升级一个版本，都要将磁盘重新格式化一次。

linux 文件类型

在前面的内容中简单介绍了普通文件(-)，目录(d)等，在linux文件系统中，主要有以下几种类型的文件。

1) 正规文件 (regular file)：就是一般类型的文件，当用ls -l查看某个目录时，第一个属性为“-”的文件就是正规文件，或者叫普通文件。正规文件又可分成纯文字文件 (ascii) 和二进制文件 (binary)。纯文本文件是可以通过cat, more, less等工具直接查看内容的，而二进制文件并不能。例如我们用的命令/bin/ls 这就是一个二进制文件。

2) 目录 (directory) : 这个很容易理解, 就是目录, 跟windows下的文件夹一个意思, 只不过在linux中我们不叫文件夹, 而是叫做目录。ls -l 查看第一个属性为 "d" 。

3) 连接档 (link) : ls -l 查看第一个属性为 "l", 类似windows下的快捷方式。这种文件在linux中很常见, 而且笔者在日常的系统运维工作中用的很多, 所以你要特意留意一下这种类型的文件。在后续章节笔者会介绍。

4) 设备档 (device) : 与系统周边相关的一些档案, 通常都集中在 /dev 这个目录之下! 通常又分为两种: 区块 (block) 设备档: 就是一些储存数据, 以提供系统存取接口设备, 简单的说就是硬盘啦! 例如你的一号硬盘的代码是 /dev/hda1 等等的档案啦! 第一个属性为 "b"; 字符 (character) 设备档: 亦即是一些串行端口的接口设备, 例如键盘、鼠标等等! 第一个属性为 "c" 。

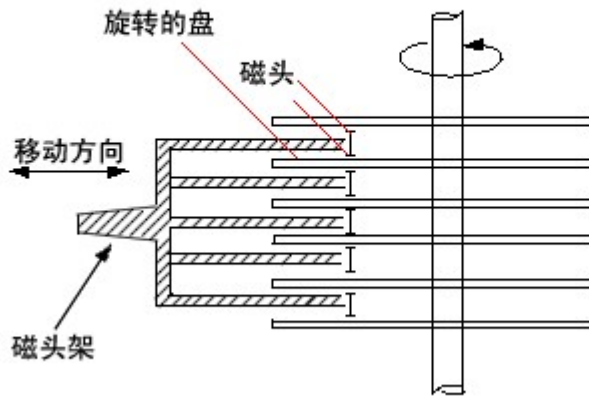
* linux 文件后缀名

对于后缀名这个概念, 相信你不陌生吧。在linux系统中, 文件的后缀名并没有具体意义, 也就是说, 你加或者不加, 都无所谓。但是为了容易区分, linux爱好者们都习惯给文件加一个后缀名, 这样当用户看到这个文件名时就会很快想到它到底是一个什么文件。例如1.sh, 2.tar.gz, my.cnf, test.zip等等, 如果你首次接触这些文件, 你也许会感到很晕, 没有关系, 随着学习的深入, 你就会逐渐的了解这些文件了。笔者所列举的几个文件名中1.sh代表它是一个shell script, 2.tar.gz代表它是一个压缩包, my.cnf代表它是一个配置文件, test.zip代表它是一个压缩文件。

另外需要你知道的是, 早期Unix系统文件名最多允许14个字符, 而新的Unix或者linux系统中, 文件名最长可以到达 256 个字符!

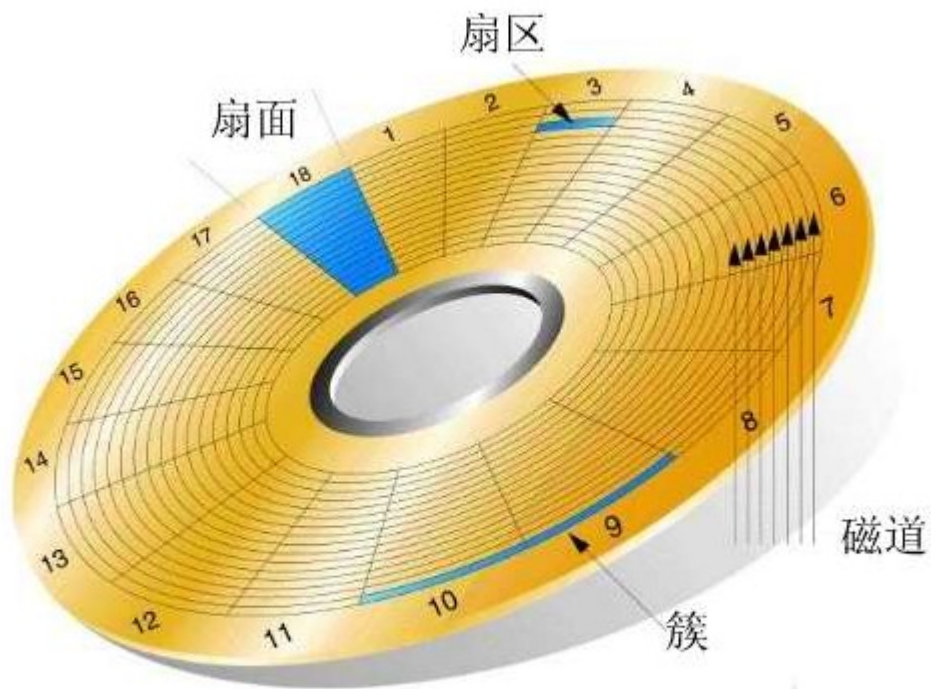
linux 中的连接档

在讲连接档之前, 需要你先理解inode的概念。什么是inode呢? 这就需要你知道磁盘的整体构造。磁盘是有多个盘片 (类似与光盘) 重叠在一起构成的, 而每个盘片上会有一个可以移动的磁头, 这个磁头的作用就是用来读写数据的。磁头并不是一直在动, 当磁头固定时, 盘片转一圈, 这一圈就是一个磁道了。很多个盘片同半径的那一圈的磁道总和称为磁柱。而由圆心向外画出直线, 可以得到一个个扇区, 如图二所示, 一个扇区的物理量大约是 512 bytes (约 0.5K)。



图片 6.45 6_98.png.jpg

图一



磁盘上的磁道、扇区和簇

图片 6.46 6_99.png.jpg

图二

知道了大体的硬盘构造之后，再来谈一谈怎么硬盘分割(partition)呢？我们在进行硬盘分割的时候，最小都是以磁柱为单位进行分割的，那么分割完成之后自然就是格式化(format)啰，在 Linux 里面我们在进行格式化的时候必须要考虑到 Block 与 inode 的信息，这个 block 还好理解，他是我们磁盘可以记录的最小单位，是由数个 sector 所组成的，所以他的大小通常为 $n \times 512 \text{ bytes}$ ，例如 4K。那么 inode 是什么？Block 是记录“档案内容数据”的地区，而 inode 则是记录“该档案的属性、及该档案放置在哪一个 Block 之内”的信息！所以，每个档案都会占用到至少一个 inode。而当我们 Linux 系统要找到这个档案时，他会先去搜寻 inode table 找到这个档案的属性及数据放置的地区，然后再到数据去找到数据存放的 Block 进而将数据取出利用。这个 inode 数目在一开始就会被设定好，他的设定方式通常是利用（硬盘大小 / 一个容量），这个容量至少应该比 Block 要大一些较佳，例如刚刚的 Block 订为 4K，那么 inode 可以订为 8K 左右。所以，一颗 1GB 的硬盘，如果以 8K 来规划他的 inode 数时，他的 inode 就会有 131072 个 inode 啦！而一个 inode 的大小为 128 bytes 这么大！这么一来的话，我们就可以清楚的知道了，那就是一个 partition 格式化为一个 filesystem 之后，基本上，他一定会有 inode table 与 data area 两个区块，一个用来记录档案的信息与该档案放置的 block 区块，一个用来记录档案的内容！

由于我们 Linux 在读取数据的时候，是先查询 inode table 以得到数据是放在那个 Block 里面，然后再去该 Block 里面读取真正的数据内容！然后，那个 block 是在我们格式化硬盘的时候规定出来的一个值，这个 block 是由 2 的 n 次方个 sector 所集结而成的！所以，他是 0.5K 的倍数！假设我们 block 规划为 4KBytes 好了，那么由于一个 inode 与一个 block 最多均只纪录一个档案，所以，如果你的一个档案有 0.1 K bytes 这么大时，你要晓得的是，由于你的 block 为 4K bytes，因此，你就会有 3.9 Kbytes 的空间浪费掉！所以，当你在格式化硬盘的时候，请千万注意到你的系统未来的使用方向。

In 建立连接档

前面提到过两次连接档的概念，现在终于该好好介绍下这部分内容了。连接档分为两种，硬连接（hard link）和软连接（symbolic link）。

Hard Links: 上面内容中说过，当系统要读取一个文件时，就会先去读inode table，然后再去根据inode中的信息到块区域去将数据取出来。而hard link 是直接再建立一个inode连接到文件放置的块区域。也就是说，进行hard link的时候实际上该文件内容没有任何变化，只是增加了一个指到这个文件的inode，不过这样一来就会有个问题，因为增加的inode会连接到块区域，而目录本身仅仅消耗inode而已，那么hard link就不能连接目录了。请记住，hard link 有两个限制：1 不能跨文件系统，因为不通的文件系统有不同的inode table；2 不能连接目录。

Symbolic Links: 跟hard link不同，这个是建立一个独立的文件，而这个文件的作用是在读取这个连接文件时，它会把读取的行为转发到该文件所link的文件上。这样讲，也许比较绕口，那么就来举一个例子。现在有文件 a，我们做了一个软连接文件 b（只是一个连接文件，非常小），b 指向了文件 a。当读取 b 时，那么 b 就会把读取的

动作转发到a上，这样就读取到了文件a。所以，当你删除文件a时，文件b并不会被删除，但是再读取b时，会提示无法打开文件。而，当你删除b时，a是不会有影响的。

看样子，似乎 hard link 比较安全，因为即使某一个 inode 被杀掉了，只要有任何一个 inode 存在，那么该文件就不会不见！不过，不幸的是，由于 Hard Link 的限制太多了，包括无法做目录的 link，所以在用途上面是比较受限的！反而是 Symbolic Link 的使用方向较广！那么如何建立软连接和硬连接呢？这就用到了ln 命令。

ln 语法：ln [-s] [来源文件] [目的文件]

ln 常用的选项就一个-s，如果不加就是建立硬连接，加上就建立软连接。

```
[root@localhost ~]# mkdir 123
[root@localhost ~]# cd 123
[root@localhost 123]# cp /etc/passwd ./
[root@localhost 123]# ll
total 8
-rw-r--r-- 1 root root 1126 May  4 11:12 passwd
[root@localhost 123]# du -sk
16
[root@localhost 123]# ln passwd passwd-hard
[root@localhost 123]# ll
total 16
-rw-r--r-- 2 root root 1126 May  4 11:12 passwd
-rw-r--r-- 2 root root 1126 May  4 11:12 passwd-hard
[root@localhost 123]# du -sk
16
```

图片 6.47 6_100.png.jpg

在建立硬连接前后，123目录所占空间大小并没有改变。

```
[root@localhost 123]# rm -f passwd
[root@localhost 123]# ll
total 8
-rw-r--r-- 1 root root 1126 May  4 11:12 passwd-hard
[root@localhost 123]# du -sk
16
```

图片 6.48 6_101.png.jpg

当把源文件删除后，空间仍旧没有变化。说明了删除一个文件其实只是删除了inode信息。

```
[root@localhost 123]# cd ..
[root@localhost ~]# ln 123 246
ln: `123': hard link not allowed for directory
```

图片 6.49 6_102.png.jpg

不能创建目录的硬连接。

```
[root@localhost ~]# mkdir 456
[root@localhost ~]# cd 456
[root@localhost 456]# cp /etc/passwd ./
[root@localhost 456]# du -sk
16
.
[root@localhost 456]# ln -s passwd passwd-soft
[root@localhost 456]# ll
total 12
-rw-r--r-- 1 root root 1126 May  4 11:17 passwd
lrwxrwxrwx 1 root root   6 May  4 11:17 passwd-soft -> passwd
[root@localhost 456]# du -sk
20
.
```

图片 6.50 6_103.png.jpg

建立软连接后，456目录增加了4k

```
[root@localhost 456]# head -n1 passwd-soft
root:x:0:0:root:/root:/bin/bash
[root@localhost 456]# rm -f passwd
[root@localhost 456]# head -n1 passwd-soft
head: cannot open `passwd-soft' for reading: No such file or directory
[root@localhost 456]# ll
total 4
lrwxrwxrwx 1 root root 6 May  4 11:17 passwd-soft -> passwd
```

图片 6.51 6_104.png.jpg

删除源文件后会提示“没有这个文件”的错误。

```
[root@localhost ~]# ln -s 456 789
[root@localhost ~]# ll 789
lrwxrwxrwx 1 root root 3 May  4 11:21 789 -> 456
[root@localhost ~]# ll 456/
total 4
lrwxrwxrwx 1 root root 6 May  4 11:17 passwd-soft -> passwd
[root@localhost ~]# ll 789/
total 4
lrwxrwxrwx 1 root root 6 May  4 11:17 passwd-soft -> passwd
```

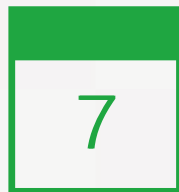
图片 6.52 6_105.png.jpg

目录是可以软连接的。

```
[root@localhost ~]# rm -f 789
[root@localhost ~]# ll 456
total 4
lrwxrwxrwx 1 root root 6 May  4 11:17 passwd-soft -> passwd
```

图片 6.53 6_106.png.jpg

删除软连接对源文件没有任何影响。



linux 系统用户以及用户组管理



关于这部分内容，笔者在日常的linux系统管理工作中用到的并不多，但这并不代表该内容不重要。毕竟linux系统是一个多用户的系统，每个账号都干什么用，你必须了如指掌。因为这涉及到一个安全的问题。

【认识/etc/passwd和/etc/shadow】

这两个文件可以说是linux系统中最重要的文件之一。如果没有这两个文件或者这两个文件出问题，则你是无法正常登录linux系统的。

```
[root@localhost ~]# cat /etc/passwd |head
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
```

图片 7.1 7_1.png.jpg

/etc/passwd由‘:’分割成7个字段，每个字段的具体含义是：

1) 用户名（如第一行中的root就是用户名），代表用户账号的字符串。用户名字符可以是大小写字母、数字、减号（不能出现在首位）、点以及下划线，其他字符不合法。虽然用户名中可以出现点，但不建议使用，尤其是首位为点时，另外减号也不建议使用，因为容易造成混淆。

2) 存放的就是该账号的口令，为什么是‘x’呢？早期的unix系统口令确实是存放在这里，但基于安全因素，后来就将其存放到/etc/shadow中了，在这里只用一个‘x’代替。

3) 这个数字代表用户标识号，也叫做uid。系统识别用户身份就是通过这个数字来的，0就是root，也就是说你可以修改test用户的uid为0，那么系统会认为root和test为同一个账户。通常uid的取值范围是0~65535，0是超级用户（root）的标识号，1~499由系统保留，作为管理账号，普通用户的标识号从500开始，如果我们自定义建立一个普通用户，你会看到该账户的标识号是大于或等于500的。

4) 表示组标识号，也叫做gid。这个字段对应着/etc/group中的一条记录，其实/etc/group和/etc/passwd基本上类似。

5) 注释说明，该字段没有实际意义，通常记录该用户的一些属性，例如姓名、电话、地址等等。不过，当你使用finger的功能时就会显示这些信息的（稍后做介绍）。

6) 用户的家目录，当用户登录时就处在这个目录下。root的家目录是/root，普通用户的家目录则为/home/user name，这个字段是可以自定义的，比如你建立一个普通用户test1，要想让test1的家目录在/data目录下，只要修改/etc/passwd文件中test1那行中的该字段为/data即可。

7) shell, 用户登录后要启动一个进程, 用来将用户下达的指令传给内核, 这就是shell。Linux的shell有很多种sh, csh, ksh, tcsh, bash等, 而Redhat/CentOS的shell就是bash。查看/etc/passwd文件, 该字段中除了/bin/bash外还有/sbin/nologin比较多, 它表示不允许该账号登录。如果你想建立一个账号不让他登录, 那么就可以把该字段改成/sbin/nologin, 默认是/bin/bash。

```
[root@localhost ~]# cat /etc/shadow
root:$1$8xBEuPuG$UT5bc0.mg50Kyqp9ncmpz/:15092:0:99999:7:::
bin:!:15090:0:99999:7:::
daemon:!:15090:0:99999:7:::
adm:!:15090:0:99999:7:::
```

图片 7.2 7_12.png.jpg

再来看看/etc/shadow这个文件, 和/etc/passwd类似, 用” : ”分割成9个字段。

1) 用户名, 跟/etc/passwd对应。

2) 用户密码, 这个才是该账号的真正的密码, 不过这个密码已经加密过了, 但是有些黑客还是能够解密的。所以为了安全, 该文件属性设置为600, 只允许root读写。

3) 上次更改密码的日期, 这个数字是这样计算得来的, 距离1970年1月1日到上次更改密码的日期, 例如上次更改密码的日期为2012年1月1日, 则这个值就是 $365 * (2012 - 1970) + 1 = 15331$ 。

4) 要过多少天才可以更改密码, 默认是0, 即不限制。

5) 密码多少天后到期。即在多少天内必须更改密码, 例如这里设置成30, 则30天内必须更改一次密码, 否则将不能登录系统, 默认是99999, 可以理解为永远不需要改。

6) 密码到期前的警告期限, 若这个值设置成7, 则表示当7天后密码过期时, 系统就发出警告告诉用户, 提醒用户他的密码将在7天后到期。

7) 账号失效期限。你可以这样理解, 如果设置这个值为3, 则表示: 密码已经到期, 然而用户并没有在到期前修改密码, 那么再过3天, 则这个账号就失效了, 即锁定了。

8) 账号的生命周期, 跟第三段一样, 是按距离1970年1月1日多少天算的。它表示的含义是, 账号在这个日期前可以使用, 到期后账号作废。

9) 作为保留用的, 没有什么意义。

【新增/删除用户和用户组】

a. 新增一个组 `groupadd [-g GID] groupname`

```
[root@localhost ~]# groupadd grptest1
[root@localhost ~]# tail -n 5 /etc/group
stapusr:x:103:
haldaemon:x:68:
slocate:x:21:
test:x:500:
grptest1:x:501:
```

图片 7.3 7_13.png.jpg

不加-g 则按照系统默认的gid创建组，跟用户一样，gid也是从500开始的

```
[root@localhost ~]# groupadd -g 505 grptest2
[root@localhost ~]# tail -n 5 /etc/group
haldaemon:x:68:
slocate:x:21:
test:x:500:
grptest1:x:501:
grptest2:x:505:
```

图片 7.4 7_14.png.jpg

-g选项可以自定义gid

b. 删除组 groupdel groupname

```
[root@localhost ~]# groupdel grptest1
[root@localhost ~]# groupdel grptest2
[root@localhost ~]# tail -n 5 /etc/group
stapdev:x:102:
stapusr:x:103:
haldaemon:x:68:
slocate:x:21:
test:x:500:
```

图片 7.5 7_15.png.jpg

没有特殊选项。

c. 增加用户 useradd [-u UID] [-g GID] [-d HOME] [-M] [-s]

-u 自定义UID

-g 使其属于已经存在的某个GID

-d 自定义用户的家目录

-M 不建立家目录

-s 自定义shell

```
[root@localhost ~]# useradd test10
[root@localhost ~]# useradd -u 510 -g 500 -M -s /sbin/nologin test11
[root@localhost ~]# tail /etc/passwd
dbus:x:81:81:System message bus:/:/sbin/nologin
exim:x:93:93::/var/spool/exim:/sbin/nologin
avahi:x:70:70:Avahi daemon:/:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
xfs:x:43:43:X Font Server:/etc/X11/fs:/sbin/nologin
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
test:x:500:500::/home/test:/bin/bash
test1:x:501:500::/home/test1:/bin/bash
test10:x:502:502::/home/test10:/bin/bash
test11:x:510:500::/home/test11:/sbin/nologin
```

图片 7.6 7_16.png.jpg

你会发现，创建test11时，加上了-M选项后，在/etc/passwd文件中test11那行的第六字段依然有/home/test11，可是ls查看该目录时，会提示该目录不存在。

```
[root@localhost ~]# ls /home/test11
ls: /home/test11: No such file or directory
```

图片 7.7 7_17.png.jpg

-M选项的作用就是不创建用户的家目录。

-d. 删除用户 userdel [-r] username

```
[root@localhost ~]# userdel test11
[root@localhost ~]# userdel -r test10
[root@localhost ~]# ls /etc/test10
ls: /etc/test10: No such file or directory
```

图片 7.8 7_18.png.jpg

-r 选项的作用是删除用户时，连同用户的家目录一起删除。

【 chfn 更改用户的finger（不常用）】

前面内容中提到了finger，即在/etc/passwd文件中的第5个字段中所显示的信息，那么如何去设定这个信息呢？

```
[root@localhost ~]# chfn test
Changing finger information for test.
Name []: test
Office []: test's Office
Office Phone []: 12345
Home Phone []: 67890

Finger information changed.
[root@localhost ~]# grep test /etc/passwd
test:x:500:500:test,test's Office,12345,67890:/home/test:/bin/bash
```

图片 7.9 7_19.png.jpg

就是chfn这个命令了。修改完后，就会在/etc/passwd文件中的test的那一行第五个字段中看到相关信息了，默认是空的。

【创建/修改一个用户的密码 “passwd [username]”】

等创建完账户后，默认是没有设置密码的，虽然没有密码，但该账户同样登录不了系统。只有设置好密码后方可登录系统。

为用户创建密码时，为了安全起见，请尽量设置复杂一些。你可以按照这样的规则来设置密码：a. 长度大于10个字符；b. 密码中包含大小写字母数字以及特殊字符（*&等）；c. 不规则性（不要出现root, happy, love, linux, 123456, 111111等等单词或者数字）；d. 不要带有自己名字、公司名字、自己电话、自己生日等。

```
[root@localhost ~]# passwd
Changing password for user root.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@localhost ~]# passwd test
Changing password for user test.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

图片 7.10 7_20.png.jpg

passwd 后面不跟用户名则是更改当前用户的密码，当前用户为root，所以此时修改的是root的密码，后面跟test则修改的是test的密码。

【用户身份切换】

Linux系统中，有时候普通用户有些事情是不能做的，除非是root用户才能做到。这时就需要临时切换到root身份来做事了。

```
login as: test
test@10.0.2.60's password:
[test@localhost ~]$ █
```

图片 7.11 7_21.png.jpg

用test账号登录linux系统，然后使用su - 就可以切换到root身份，前提是知道root的密码。

```
[test@localhost ~]$ su -
Password:
[root@localhost ~]# █
```

图片 7.12 7_29.png.jpg

你可以使用echo \$LOGNAME来查看当前登录的用户名

```
[test@localhost ~]$ echo $LOGNAME
test
[test@localhost ~]$ su -
Password:
[root@localhost ~]# echo $LOGNAME
root
```

图片 7.13 7_30.png.jpg

su 的语法为: su [-] username

后面可以跟“-”也可以不跟,普通用户su不加username时就是切换到root用户,当然root用户同样可以su到普通用户。

```
[root@localhost ~]# su - test
[test@localhost ~]$ pwd
/home/test
[test@localhost ~]$ logout
[root@localhost ~]# su test
[test@localhost root]$ pwd
/root
```

图片 7.14 7_31.png.jpg

加“-”后会连同用户的环境变量一起切换过来。su test 后虽然切换到了test用户,但是当前目录还是切换前的/root目录,然后当用su - test时切换用户后则到了test的家目录/home/test。当用root切换普通用户时,是不需要输入密码的。这也体现了root用户至高无上的权利。

用su是可以切换用户身份,如果每个普通用户都能切换到root身份,如果某个用户不小心泄漏了root的密码,那岂不是系统非常的不安全?没有错,为了改进这个问题,产生了sudo这个命令。使用sudo执行一个root才能执行的命令是可以办到的,但是需要输入密码,这个密码并不是root的密码而是用户自己的密码。默认只有root用户能使用sudo命令,普通用户想要使用sudo,是需要root预先设定的,即,使用visudo命令去编辑相关的配置文件/etc/sudoers。如果没有visudo这个命令,请使用“yum install -y sudo”安装。

```
## Allow root to run any commands anywhere
root    ALL=(ALL)    ALL
test    ALL=(ALL)    ALL
```

图片 7.15 7_32.png.jpg

默认root能够sudo是因为这个文件中有一行“root ALL=(ALL) ALL”在该行下面加入“test ALL=(ALL) ALL”就可以让test用户拥有了sudo的权利。如果每增加一用户就设置一行,这样太麻烦了。所以你可以这样设置。

```
## Allows people in group wheel to run all commands
# %wheel    ALL=(ALL)    ALL
```

图片 7.16 7_33.png.jpg

把这一行前面的”#”去掉，让这一行生效。它的意思是，wheel这个组的所有用户都拥有了sudo的权利。接下来就需要你把想让有sudo权利的所有用户加入到wheel这个组中即可。

```
[root@localhost ~]# su test
[test@localhost root]$ touch 1.txt
touch: cannot touch `1.txt': Permission denied
[test@localhost root]$ sudo touch 1.txt
[test@localhost root]$ ls -l 1.txt
ls: 1.txt: Permission denied
[test@localhost root]$ sudo ls 1.txt
1.txt
[test@localhost root]$ sudo ls -l 1.txt
-rw-r--r-- 1 root root 0 May 10 15:30 1.txt
```

图片 7.17 7_34.png.jpg



Linux 磁盘管理



【查看磁盘或者目录的容量 df 和 du】

df 查看已挂载磁盘的总容量、使用容量、剩余容量等，可以不加任何参数，默认是按k为单位显示的

```
[root@localhost ~]# df
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/hda3        7625092    2610472    4621028   37% /
/dev/hda1         101086      11412     84455   12% /boot
tmpfs             117564         0     117564    0% /dev/shm
```

图片 8.1 8_1.png.jpg

df常用参数有 -i -h -k -m等

-i 使用inodes 显示结果

```
[root@localhost ~]# df -i
Filesystem      Inodes    IUsed   IFree IUse% Mounted on
/dev/hda3      1969568  118574 1850994    7% /
/dev/hda1       26104      35    26069    1% /boot
tmpfs           29391      1    29390    1% /dev/shm
```

图片 8.2 8_12.png.jpg

-h 使用合适的单位显示，例如G

```
[root@localhost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/hda3        7.3G  2.5G  4.5G  37% /
/dev/hda1         99M   12M   83M  12% /boot
tmpfs            115M     0   115M   0% /dev/shm
```

图片 8.3 8_13.png.jpg

-k -m 分别为使用K，M为单位显示

```
[root@localhost ~]# df -m
Filesystem      1M-blocks      Used Available Use% Mounted on
/dev/hda3         7447         2550     4513   37% /
/dev/hda1          99           12        83   12% /boot
tmpfs             115           0        115    0% /dev/shm
[root@localhost ~]# df -k
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/hda3        7625092    2610472    4621028   37% /
/dev/hda1         101086      11412     84455   12% /boot
tmpfs             117564         0     117564    0% /dev/shm
```

图片 8.4 8_14.png.jpg

简单介绍一下，你看到的相关数据。Filesystem 表示扇区，也就是你划分磁盘时所分的区；1K-blocks/1M-blocks表示以1K/1M为单位；Used 和 Available 分别是已使用和剩余；Use% 就是已经使用的百分比，如果这个值大于90% 那么你就应该注意了，磁盘很有可能马上就会变满的；Mounted on 则表示该分区（扇区）所挂载的地方。

du 用来查看某个目录所占空间大小

语法: du [-abckmsh] [文件或者目录名] 常用的参数有:

-a: 全部文件与目录大小都列出来。如果不加任何选项和参数只列出目录 (包含子目录) 大小。

```
[root@localhost ~]# du file1
12    file1/file2
24    file1
[root@localhost ~]# du -a file1
4     file1/file3
4     file1/file2/file4
12    file1/file2
24    file1
```

图片 8.5 8_15.png.jpg

-b: 列出的值以bytes为单位输出, 默认是以Kbytes

```
[root@localhost ~]# du -b file1
4096  file1/file2
8192  file1
```

图片 8.6 8_16.png.jpg

-c: 最后加总

```
[root@localhost ~]# du -c file1
12    file1/file2
24    file1
24    total
```

图片 8.7 8_17.png.jpg

-k: 以KB为单位输出

-m: 以MB为单位输出

-s: 只列出总和

-h: 系统自动调节单位, 例如文件太小可能就几K, 那么就以K为单位显示, 如果大到几G, 则就以G为单位显示。笔者习惯用 du -sh filename 这样的形式。

```
[root@localhost ~]# du -sh file1
24K    file1
[root@localhost ~]# du -sh /usr/
2.6G   /usr/
```

图片 8.8 8_18.png.jpg

【磁盘的分区和格式化】

笔者经常做的事情就是拿一个全新的磁盘来分区并格式化。这也说明了作为一个linux系统管理员，对于磁盘的操作必须要熟练。所以请你认真学习该部分内容。

fdisk linux下的硬盘分区工具

语法：fdisk [-l] [设备名称]

-l：后边不跟设备名会直接列出系统中所有的磁盘设备以及分区表，加上设备名会列出该设备的分区表。

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

Disk /dev/hda: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1          13        104391   83   Linux
/dev/hda2             14          64       409657+   82   Linux swap / Solaris
/dev/hda3             65         1044       7871850   83   Linux
```

图片 8.9 8_19.png.jpg

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

Disk /dev/hda: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1          13        104391   83   Linux
/dev/hda2             14          64       409657+   82   Linux swap / Solaris
/dev/hda3             65         1044       7871850   83   Linux
```

图片 8.10 8_20.png.jpg

如果不加-l 则进入另一个模式，在该模式下，可以对磁盘进行分区操作。

```
[root@localhost ~]# fdisk /dev/hda

The number of cylinders for this disk is set to 1044.
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): m
Command action
  a toggle a bootable flag
  b edit bsd disklabel
  c toggle the dos compatibility flag
  d delete a partition
  l list known partition types
  m print this menu
  n add a new partition
  o create a new empty DOS partition table
  p print the partition table
  q quit without saving changes
  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
  x extra functionality (experts only)
```

图片 8.11 8_21.png.jpg

刚进入该模式下，会有一个提示Command (m for help): 此时按m则会打印出帮助列表，如果你英文好，我想你不难理解这些字母的功能。笔者常用的有p, n,d, w, q.

P: 打印当前磁盘的分区情况。

```
Command (m for help): p

Disk /dev/hda: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1          13        104391   83  Linux
/dev/hda2             14          64       409657+   82  Linux swap / Solaris
/dev/hda3             65         1044       7871850   83  Linux
```

图片 8.12 8_29.png.jpg

n: 重新建立一个新的分区。

w: 保存操作。

q: 退出。

d: 删除一个分区

因为笔者的linux系统是安装在虚拟机上的，所以我可以增加一块新的磁盘。然后笔者会把新的磁盘分成多个分区。

```
[root@localhost ~]# fdisk -l
Disk /dev/hda: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1    *           1          13        104391   83   Linux
/dev/hda2                14          64        409657+   82   Linux swap / Solaris
/dev/hda3                65         1044       7871850   83   Linux

Disk /dev/hdb: 1073 MB, 1073741824 bytes
16 heads, 63 sectors/track, 2080 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

Disk /dev/hdb doesn't contain a valid partition table
```

图片 8.13 8_30.png.jpg

当再次fdisk -l 查看时发现多了一个/dev/hdb 设备，并提示该设备没有可用的分区表。那么下面就来分一下这个/dev/hdb.

```
Command (m for help): p
Disk /dev/hdb: 1073 MB, 1073741824 bytes
16 heads, 63 sectors/track, 2080 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot      Start         End      Blocks   Id  System
```

图片 8.14 8_31.png.jpg

首先用p查看一下，并没有任何分区信息。

```

Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-2080, default 1): 1
Last cylinder or +size or +sizeM or +sizeK (1-2080, default 2080): +100M

Command (m for help): p

Disk /dev/hdb: 1073 MB, 1073741824 bytes
16 heads, 63 sectors/track, 2080 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1            1          195       98248+   83  Linux

```

图片 8.15 8_32.png.jpg

用n创建一个新的分区，会提示要建立e（extended 扩展分区）或者p（primary partition主分区），这里笔者选择主分区，所以按了p回车后，又让输入First cylinder 你或者直接回车或者输入一个数字，因为这块磁盘是新的并没有任何分区，所以直接回车其实就是从1开始了。你也可以自定义输入，但不要超过2080，笔者这里输入1回车。此时会提示要分多大，可以写一个数值（2-2080），也可以输入+sizeK或者+sizeM，后者比较直观容易理解，所以笔者在这里输入+100M，即我分了一个100M的主分区。再用p查看时，果真多出来一个分区。然后笔者继续重复前面的操作，建立了4个主分区。当笔者再次输入n创建分区时，结果提示错了。

```

Command (m for help): p

Disk /dev/hdb: 1073 MB, 1073741824 bytes
16 heads, 63 sectors/track, 2080 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1            1          195       98248+   83  Linux
/dev/hdb2           196          390       98280    83  Linux
/dev/hdb3           391          585       98280    83  Linux
/dev/hdb4           586          780       98280    83  Linux

Command (m for help): n
You must delete some partition and add an extended partition first

```

图片 8.16 8_33.png.jpg

由此你会发现，在linux中最多只能创建4个主分区，那如果你想多创建几个分区如何做？很容易，在创建完第三个分区后，创建第四个分区时选择扩展分区。

```

Command (m for help): d
Partition number (1-4): 4

Command (m for help): p

Disk /dev/hdb: 1073 MB, 1073741824 bytes
16 heads, 63 sectors/track, 2080 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1            1          195       98248+   83  Linux
/dev/hdb2           196          390       98280    83  Linux
/dev/hdb3           391          585       98280    83  Linux

```

图片 8.17 8_34.png.jpg

先删除第四个主分区，然后建立一个扩展分区

```

Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
e
Selected partition 4
First cylinder (586-2080, default 586):
Using default value 586
Last cylinder or +size or +sizeM or +sizeK (586-2080, default 2080):
Using default value 2080

Command (m for help): p

Disk /dev/hdb: 1073 MB, 1073741824 bytes
16 heads, 63 sectors/track, 2080 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1            1          195       98248+   83  Linux
/dev/hdb2           196          390       98280    83  Linux
/dev/hdb3           391          585       98280    83  Linux
/dev/hdb4           586         2080      753480    5  Extended

```

图片 8.18 8_35.png.jpg

在建立扩展分区时，会问你要分多少给这个扩展分区，笔者直接回车，即把所有空间都分给了这个扩展分区。这个扩展分区/dev/hdb4并不能往里写数据，它只是一个空壳子，需要我们继续在这个空壳中继续创建分区。

```

Command (m for help): n
First cylinder (586-2080, default 586):
Using default value 586
Last cylinder or +size or +sizeM or +sizeK (586-2080, default 2080): +100M

Command (m for help): p

Disk /dev/hdb: 1073 MB, 1073741824 bytes
16 heads, 63 sectors/track, 2080 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1             1          195       98248+   83  Linux
/dev/hdb2           196          390       98280    83  Linux
/dev/hdb3           391          585       98280    83  Linux
/dev/hdb4           586         2080      753480    5  Extended
/dev/hdb5           586          780       98248+   83  Linux

```

图片 8.19 8_83.png.jpg

当建立完扩展分区，然后按n创建新分区时你会发现不再提示是要建立p还是e了，因为我们已经不能再创建p了。在这里需要你明白的是，hdb5 其实只是 hdb4 中的一个子分区，到目前为止可用的分区也才4个，那笔者就再创建第5个分区出来。

```

Command (m for help): n
First cylinder (781-2080, default 781):
Using default value 781
Last cylinder or +size or +sizeM or +sizeK (781-2080, default 2080): +100M

Command (m for help): p

Disk /dev/hdb: 1073 MB, 1073741824 bytes
16 heads, 63 sectors/track, 2080 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1             1          195       98248+   83  Linux
/dev/hdb2           196          390       98280    83  Linux
/dev/hdb3           391          585       98280    83  Linux
/dev/hdb4           586         2080      753480    5  Extended
/dev/hdb5           586          780       98248+   83  Linux
/dev/hdb6           781          975       98248+   83  Linux

```

图片 8.20 8_84.png.jpg

然后按w保存，该模式自动退出，如果你不想保存分区信息直接按q即可退出。

```

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.

```

图片 8.21 8_85.png.jpg

下面我们把刚分好的分区删除，重新建立分区。如何删除你还记得吧，对了就是直接按d然后选择合适的数字。删除完所有分区后，这块磁盘就恢复如初了。

```
Command (m for help): p

Disk /dev/hdb: 1073 MB, 1073741824 bytes
16 heads, 63 sectors/track, 2080 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot      Start         End      Blocks   Id  System

```

图片 8.22 8_86.png.jpg

```
Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
e
Partition number (1-4): 1
First cylinder (1-2080, default 1): 1
Last cylinder or +size or +sizeM or +sizeK (1-2080, default 2080): +200M

Command (m for help): p

Disk /dev/hdb: 1073 MB, 1073741824 bytes
16 heads, 63 sectors/track, 2080 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1          1           389     196024+    5  Extended

```

图片 8.23 8_87.png.jpg

第一个分区，我们就建立成扩展分区。并且分给它200M。

```
Command (m for help): n
Command action
  l   logical (5 or over)
  p   primary partition (1-4)

```

图片 8.24 8_88.png.jpg

当再次新建分区时，发生了变化，不再是p或者e了，而是p或者l（逻辑分区），这是为什么呢？在上面也提到了，一个扩展分区只是一个空壳，在扩展分区下才可以继续划分小的分区，这个小的分区其实就是逻辑分区了。

```

Command (m for help): n
Command action
  l   logical (5 or over)
  p   primary partition (1-4)
l
First cylinder (1-389, default 1): 1
Last cylinder or +size or +sizeM or +sizeK (1-389, default 389): +100M

Command (m for help): p

Disk /dev/hdb: 1073 MB, 1073741824 bytes
16 heads, 63 sectors/track, 2080 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1            1         389     196024+    5  Extended
/dev/hdb5            1         195      98217    83  Linux

```

图片 8.25 8_89.png.jpg

而且这个逻辑分区默认都是从数字5开始的，因为前面的数字要么给主分区留着，要么给扩展分区留着。由此我们也可以得到，在linux中最多可以创建4个主分区，一旦创建4个主分区后就不能增加任何分区了。另外最多也只能创建一个扩展分区。扩展分区下的逻辑分区最多可以创建多少个呢？IDE的硬盘（类似于hda, hdb, hdc等）最多可以创建10个（hdb5-hdb15），这是笔者试验出来的结果。有的资料说linux下的逻辑分区是没有限制的，也有的说最大可以到64，至于对不对，需要你去进一步考察了，我们没有必要多么深入的研究这个问题，也没有什么意义。

通过以上操作，相信你也学会了用fdisk来分区了吧。值得提出的是，不要闲着没事分区玩儿，这操作的危险性是很高的，一不留神就把你服务器上的数据全部给分没有了。如果有分区操作，那么请保持百分之百的细心，切记切记！

mkfs.ext2 / mkfs.ext3 / mke2fs 格式化linux硬盘分区

当用man查询这三个命令的帮助文档时，你会发现我们看到了同一个帮助文档，这说明三个命令是一样的。常用的选项有：

-b：分区时设定每个数据区块占用空间大小，目前支持1024, 2048 以及4096 bytes每个块。

-i：设定inode大小

-N：设定inode数量，有时使用默认的inode数不够用，所以要自定义设定inode数量。

-c：在格式化前先检测一下磁盘是否有问题，加上这个选项后会非常慢

-L：预设该分区的标签label

-j：建立ext3格式的分区，如果使用mkfs.ext3就不用加这个选项了

```
[root@localhost ~]# mkfs.ext3 /dev/hdb1
mke2fs 1.39 (29-May-2006)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
24576 inodes, 98248 blocks
4912 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=67371008
12 block groups
8192 blocks per group, 8192 fragments per group
2048 inodes per group
Superblock backups stored on blocks:
    8193, 24577, 40961, 57345, 73729

Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 21 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

图片 8.26 8_90.png.jpg

不加任何选项，直接格式化/dev/hdb1

```
[root@localhost ~]# mkfs.ext3 -b 4096 -i 4096 /dev/hdb2
mke2fs 1.39 (29-May-2006)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
24576 inodes, 24570 blocks
1228 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=25165824
1 block group
32768 blocks per group, 32768 fragments per group
24576 inodes per group

Writing inode tables: done
Creating journal (1024 blocks): done
Writing superblocks and filesystem accounting information: done
```

图片 8.27 8_91.png.jpg

上例中更改了block size为4096 默认是1024，而inode大小设定为4096。

下面的例子分区时自定义分区的label（标签）名。

```
[root@localhost ~]# mkfs.ext3 -L label2 /dev/hdb2
mke2fs 1.39 (29-May-2006)
Filesystem label=label2
```

图片 8.28 8_92.png.jpg

e2label 用来查看或者修改分区的标签 (label)

这个命令很简单，后边直接跟分区编号，即可查看该分区的label，当想要修改标签名时，分区编号后边跟想要的标签名即可。

```
[root@localhost ~]# e2label /dev/hdb2
label2
[root@localhost ~]# e2label /dev/hdb1

[root@localhost ~]# e2label /dev/hdb1 label1
[root@localhost ~]# e2label /dev/hdb1
label1
```

图片 8.29 8_93.png.jpg

fsck 检查硬盘有没有坏道

语法：fsck [-Aar] [分区]

-A：加该参数时，后不需要跟分区名作为参数。它会自动检查/etc/fstab 文件下的所有分区（开机过程中就会执行一次该操作）；

-a：自动修复检查到有问题的分区；

-r：当检查到有坏道的分区时会让用户决定是否修复。

```
[root@localhost ~]# fsck -A
fsck 1.39 (29-May-2006)
e2fsck 1.39 (29-May-2006)
/dev/hda3 is mounted.

WARNING!!! Running e2fsck on a mounted filesystem may cause
SEVERE filesystem damage.

Do you really want to continue (y/n)? yes

/: recovering journal
/: clean, 118571/1969568 files, 713920/1967962 blocks
e2fsck 1.39 (29-May-2006)
/dev/hda1 is mounted. e2fsck: Cannot continue, aborting.
```

图片 8.30 8_94.png.jpg

当你使用fsck检查磁盘有无坏道时，会提示用户“跑这个任务可能会导致某些挂载的文件系统损坏”，所以这个命令不要轻易运行。否则真的遇到问题，系统甚至都不能启动了。

```
[root@localhost ~]# fsck -a /dev/hda1
fsck 1.39 (29-May-2006)
/dev/hda1 is mounted.

WARNING!!! Running e2fsck on a mounted filesystem may cause
SEVERE filesystem damage.

Do you really want to continue (y/n)? yes

/boot: recovering journal
/boot: clean, 35/26104 files, 14714/104388 blocks
```

图片 8.31 8_95.png.jpg

【挂载/卸载磁盘】

在上面的内容中讲到了磁盘的分区和格式化，那么格式化完了后，如何去用它呢？这就涉及到了挂载这块磁盘。格式化后的磁盘其实是一个块设备文件，类型为b，也许你会想，既然这个块文件就是那个分区，那么直接在那个文件中写数据不就写到了那个分区中么？当然不行。

在挂载某个分区前需要先建立一个挂载点，这个挂载点是以目录的形式出现的。一旦把某一个分区挂载到了这个挂载点（目录）下，那么再往这个目录写数据使，则都会写到该分区中。这就需要你注意一下，在挂载该分区前，挂载点（目录）下必须是个空目录。其实目录不为空并不影响所挂载分区的使用，但是一旦挂载上了，那么该目录下以前的东西就不能看到了。只有卸载掉该分区后才能看到。

mount 挂载设备

```
[root@localhost ~]# mkdir /test1 /test2
[root@localhost ~]# touch /test1/1.txt
[root@localhost ~]# ls /test1
1.txt
```

图片 8.32 8_96.png.jpg

先建立/test1 /test2 目录，然后在/test1目录下建立一个1.txt文件。

```
[root@localhost ~]# mount /dev/hdb1 /test1
[root@localhost ~]# ls /test1
lost+found
```

图片 8.33 8_97.png.jpg

把/dev/hdb1分区挂载到/test1目录，然后再查看/test1目录发下，1.txt不存在了。此时往/test1目录下写数据，则会写到/dev/hdb1分区中。在讲mount的-a选项时，我们有必要先了解一下这个文件 /etc/fstab

```
[root@localhost ~]# cat /etc/fstab
LABEL=/          /              ext3            defaults        1 1
LABEL=/boot     /boot          ext3            defaults        1 2
tmpfs           /dev/shm      tmpfs           defaults        0 0
devpts          /dev/pts      devpts          gid=5,mode=620 0 0
sysfs           /sys          sysfs           defaults        0 0
proc            /proc         proc            defaults        0 0
LABEL=SWAP-hda2 swap           swap            defaults        0 0
```

图片 8.34 8_98.png.jpg

这个文件是系统启动时，需要挂载的各个分区。第一列就是分区的label；第二列是挂载点；第三列是分区的格式；第四列则是mount的一些挂载参数，等下会详细介绍一下有哪些参数，一般情况下，直接写defaults即可；第五列的数字表示是否被dump备份，是的话这里就是1，否则就是0；第六列是开机时是否自检磁盘，就是刚才讲过的那个fsck检测。1，2都表示检测，0表示不检测，在Redhat中，这个1，2还有个说法，/分区必须设为1，而且整个fstab中只允许出现一个1，这里有一个优先级的说法。1比2优先级高，所以先检测1，然后再检测2，如果有多个分区需要开机检测那么都设置成2吧，1检测完了后会同时去检测2。下面该说说第四列中常用到的参数了。

async/sync：async表示和磁盘和内存不同步，系统每隔一段时间把内存数据写入磁盘中，而sync则会时时同步内存和磁盘中数据；

auto/noauto：开机自动挂载/不自动挂载；

default：按照大多数永久文件系统的缺省值设置挂载定义，它包含了rw, suid, dev, exec, auto, nouser, async；

ro：按只读权限挂载；

rw：按可读可写权限挂载；

exec/noexec：允许/不许可执行文件执行，但千万不要把根分区挂载为noexec，那就无法使用系统了，连mount命令都无法使用了，这时只有重新做系统了；

user/nouser：允许/不允许root外的其他用户挂载分区，为了安全考虑，请用nouser；

suid/nosuid：允许/不允许分区有suid属性，一般设置nosuid；

usrquota：启动使用者磁盘配额模式，磁盘配额相关内容在后续章节会做介绍；

grquota：启动群组磁盘配额模式；

学完这个/etc/fstab后，我们就可以自己修改这个文件，增加一行来挂载新增分区。例如，笔者增加了这样一行

```
/dev/hdb1 /test1 ext3 defaults 0 0
```

那么系统再重启时就会挂载这个分区了。

讲完了/etc/fstab 我们继续回来讲这个mount, `mount -a` 如果运行了这个命令, 则会把/etc/fstab中出现的所有磁盘分区挂载上。所以当你/etc/fstab文件中增加一行后, 你完全可以直接运行`mount -a` 来挂载你增加的那一行, 这样就不用重启啦。

你可以使用`mount -o` 选项来重新挂载一个分区, 并同时指定你想要的选项(即上边介绍fstab第六列中那些)

```
[root@localhost ~]# mount -o remount,ro,sync,noauto /dev/hdb1 /test1
[root@localhost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/hda3       7.3G  2.5G  4.5G  37% /
/dev/hda1        99M   12M   83M  12% /boot
tmpfs           115M    0  115M   0% /dev/shm
/dev/hdb1        93M   5.6M   83M   7% /test1
[root@localhost ~]# touch /test1/2.txt
touch: cannot touch `/test1/2.txt': Read-only file system
```

图片 8.35 8_99.png.jpg

看到了吧, 使用了ro选项, 则不能新建文件了。

```
[root@localhost ~]# mount -o remount /dev/hdb1 /test1
[root@localhost ~]# touch /test1/2.txt
[root@localhost ~]# ls /test1/
2.txt  lost+found
```

图片 8.36 8_100.png.jpg

再重新挂载一次就恢复正常了, 如果不加任何其他选项, 则就是defaults。

笔者在日常的运维工作中遇到过这样的情况, 一台服务器上新装了亮块磁盘, 磁盘a(在服务器上显示为sdc)和磁盘b(在服务器上显示为sdd), 有一次把这两块磁盘都拔掉了, 然后再重新插上, 重启机器, 结果磁盘编号调换了, a变成了sdd, b变成了sdc(这是因为把磁盘插错了插槽), 问题来了。通过上边的学习, 你挂载磁盘是通过/dev/hdb1这样的分区名字来挂载的, 如果先前加入到了/etc/fstab中, 结果系统启动后则会挂载错分区。那么怎么样避免这样的情况发生?

`blkid` 这个命令是用来显示磁盘分区uuid的, uuid其实就是一大串字符, 在linux系统中每一个分区都会有唯一的一个uuid。说到这, 聪明的你想到了吧, 没有错, 我们就用这唯一的uuid来挂载磁盘分区。

```
[root@localhost ~]# blkid
/dev/hdc: LABEL="CentOS_5.4_Final" TYPE="iso9660"
/dev/hda3: LABEL="/" UUID="98f3b220-bdf4-447c-9a72-9f30a68dd43f" SEC_TYPE="ext2" TYPE="ext3"
/dev/hda2: LABEL="SWAP-hda2" TYPE="swap"
/dev/hda1: LABEL="/boot" UUID="84c6733a-0303-438c-8caf-206532021b5a" SEC_TYPE="ext2" TYPE="ext3"
/dev/hdb1: LABEL="label1" UUID="a2fe40c2-859d-4538-801d-d38dd41a9baa" TYPE="ext3"
/dev/hdb2: LABEL="label2" UUID="8c6e67c8-b3fb-4b9c-babb-76d2cc168110" SEC_TYPE="ext2" TYPE="ext3"
```

图片 8.37 8_101.png.jpg

这个命令笔者只是用来显示uuid，没有其他用途所以不做详细介绍，当然你也可以在命令后边跟某一个分区，只显示该分区的uuid。

```
[root@localhost ~]# blkid /dev/hdb2
/dev/hdb2: LABEL="label2" UUID="8c6e67c8-b3fb-4b9c-babb-76d2cc168110" SEC_TYPE="ext2" TYPE="ext3"
[root@localhost ~]# mount UUID="8c6e67c8-b3fb-4b9c-babb-76d2cc168110" /test2
[root@localhost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/hda3       7.3G  2.5G  4.5G  37% /
/dev/hda1        99M   12M   83M  12% /boot
tmpfs           115M    0   115M   0% /dev/shm
/dev/hdb1        93M   5.6M   83M   7% /test1
/dev/hdb2        93M   5.6M   83M   7% /test2
```

图片 8.38 8_102.png.jpg

看到了吧，其实是很好用的。那么怎么让它也开机启动？很简单，把刚才敲的mount 磁盘的命令直接写到 /etc/rc.d/rc.local 文件即可。对了，笔者到现在还没有给你讲过这个rc.local文件的作用。简单点说，系统启动完后会执行这个文件中的命令。所以只要你想开机后运行什么命令统统写入到这个文件下面吧。

```
[root@localhost ~]# cat /etc/rc.d/rc.local
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local
```

图片 8.39 8_103.png.jpg

其实这个文件就是一个shell 脚本，以后笔者会单独用一章来介绍它。行开头的“#”是注释的意思，代表这行在这个脚本中不生效。你想让系统开机后运行什么命令，就把什么命令写到这里面来。就比如刚才笔者挂载的那条命令。你可以这样实现：

```
[root@localhost ~]# echo 'mount UUID="8c6e67c8-b3fb-4b9c-babb-76d2cc168110" /test2' >> /etc/rc.d/rc.local
[root@localhost ~]# cat /etc/rc.d/rc.local
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local
mount UUID="8c6e67c8-b3fb-4b9c-babb-76d2cc168110" /test2
```

图片 8.40 8_104.png.jpg

mount 还有一个比较常用的选项就是-t，后边指定文件系统的类型，比如挂载软盘时就需要指定 vfat，而挂载光盘时就需要指定iso9660，但在笔者多年来的经验，目前的系统都是智能识别所要挂载分区的系统格式类别的。也就是说，用不着你去指定，它会自动判断的。

```
[root@localhost ~]# mount -o remount -t ext3 /dev/hdb1 /test1
```

图片 8.41 8_105.png.jpg

umount 卸载设备

现在你学会了如何挂载一个设备，那么如何去卸载一个设备呢，这就要用到umount了，这个命令也简单的很，后边可以跟挂载点，也可以跟分区名(/dev/hdb1)

```
[root@localhost ~]# umount /dev/hdb1
[root@localhost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/hda3       7.3G  2.5G  4.5G  37% /
/dev/hda1       99M   12M   83M  12% /boot
tmpfs           115M   0    115M   0% /dev/shm
/dev/hdb2       93M   5.6M   83M   7% /test2
[root@localhost ~]# umount /test2
[root@localhost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/hda3       7.3G  2.5G  4.5G  37% /
/dev/hda1       99M   12M   83M  12% /boot
tmpfs           115M   0    115M   0% /dev/shm
```

图片 8.42 8_106.png.jpg

有时也许你会遇到比较难卸载的设备，就像在windows下无法删除U盘一样，教你一个特管用的方法就是 `umount -l /dev/hdb1`，这个-l选项有强制卸载的意思，你一定要记住哦，非常有用的。

【建立一个swap文件】

从装系统时就接触过这个swap了，前面也说过它类似与windows的虚拟内存，分区的时候一般大小为内存的2倍，如果你的内存超过4G，那么你分8G似乎是没有必要了。分4G足够日常交换了。然而，还会有虚拟内存不够用的情况发生。如果真遇到了，莫非还要重新分一下磁盘？当然不能！那我们就增加一个虚拟的磁盘出来。

基本的思路就是：建立swapfile ?格式化为swap格式?启用该虚拟磁盘

```
[root@localhost ~]# dd if=/dev/zero of=/tmp/newdisk bs=4k count=102400
102400+0 records in
102400+0 records out
419430400 bytes (419 MB) copied, 3.51114 seconds, 119 MB/s
```

图片 8.43 8_107.png.jpg

利用dd 来创建一个419M的文件/tmp/newdisk出来，其中if代表从哪个文件读，/dev/zero是linux下特有的一个0生成器，of表示输出到哪个文件，bs即块大小，count则定义有多少个块。

```
[root@localhost ~]# mkswap /tmp/newdisk
Setting up swapspace version 1, size = 419426 kB
```

图片 8.44 8_108.png.jpg

mkswap 这个命令是专门格式化swap格式的分区的，这个命令用的时候一定要看清楚了，否则把其他分区给格式化错了就只有哭了。

```
[root@localhost ~]# free -m
              total        used         free       shared    buffers     cached
Mem:           229          224           5            0           5          198
-/+ buffers/cache:           20          209
Swap:          400           0          399
[root@localhost ~]# swapon /tmp/newdisk
[root@localhost ~]# free -m
              total        used         free       shared    buffers     cached
Mem:           229          224           4            0           5          198
-/+ buffers/cache:           20          209
Swap:          800           0          799
```

图片 8.45 8_109.png.jpg

free 是用来查看系统内存以及虚拟内存使用情况的，-m选项是以M的形式查看。可以看到当前系统的。而swapon 是启用我们新建的swap文件，启用后再用free查看发现多了400M。

```
[root@localhost ~]# swapoff /tmp/newdisk
[root@localhost ~]# free -m
              total        used         free       shared    buffers     cached
Mem:           229          224           4            0           5          198
-/+ buffers/cache:           20          209
Swap:          400           0          399
```

图片 8.46 8_110.png.jpg

我们还可以用swapoff 关闭启用的swap文件。

【磁盘配额】

磁盘配合其实就是给每个用户分配一定的磁盘额度，只允许他使用这个额度范围内的磁盘空间。在linux系统中，是多用户多任务的环境，所以会有很多人共用一个磁盘的情况。针对每个用户去限定一定量的磁盘空间是有必要的，这样才显得公平。

在linux中，用来管理磁盘配额的东西就是quota了。如果你的linux上没有quota，则需要你安装这个软件包 quota-3.13-5.el5.RPM（其实版本是多少无所谓了，关键是这个软件包）。quota在实际应用中是针对整个分区进行限制的。如果你的/dev/hda3 是挂载在/home 目录下的，那么/home 所有目录都会受到限制。

quota 这个模块主要分为quota quotacheck quotaoff quotaon quotastats edquota setquota warnquota repquota这几个命令，下面就分别介绍这些命令。

quota 用来显示某个组或者某个使用者的限额。

语法: quota [-guvs] [user,group]

-g：显示某个组的限额

-u：显示某个用户的限额

-v : 显示的意思

-s : 选择inod或硬盘空间来显示

quotacheck 用来扫描某一个磁盘的quota空间。

语法: quotacheck [-auvg] /path

-a : 扫描所有已经mount的具有quota支持的磁盘

-u : 扫描某个使用者的文件以及目录

-g : 扫描某个组的文件以及目录

-v : 显示扫描过程

-m : 强制进行扫描

edquota 用来编辑某个用户或者组的quota值。

语法: edquota [-u user] [-g group] [-t]

edquota -p user -u user

-u : 编辑某个用户的quota

-g : 编辑某个组的quota

-t : 编辑宽限时间

-p : 拷贝某个用户或组的quta到另一个用户或组

当运行edquota -u user 时, 系统会打开一个文件, 你会看到这个文件中有7列, 它们分别代表的含义是:

Filesystem : 磁盘分区, 如/dev/hda3

blocks : 当前用户在当前的Filesystem中所占用的磁盘容量, 单位是Kb。该值请不要修改。

soft/hard : 当前用户在该Filesystem内的quota值, soft指的是最低限额, 可以超过这个值, 但必须要在宽限时间内将磁盘容量降低到这个值以下。hard指的是最高限额, 即不能超过这个值。当用户的磁盘使用量高于soft值时, 系统会警告用户, 提示其要在宽限时间内把使用空间降低到soft值之下。

inodes：目前使用掉的inode的状态，不用修改。

quotaon 启动quota，在编辑好quota后，需要启动才能是quta生效

语法：quotaon [-a] [-uv g directory]

-a：全部设定的quota启动

-u：启动某个用户的quota

-g：启动某个组的quota

-s：显示相关信息

quotaoff 关闭quota

该命令常用只有一种情况 quotaoff -a 关闭全部的quota

以上讲了很多quota的相关命令，那么接下来笔者教你如何在实践应用中去这个磁盘配额。整个执行过程如下：

首先先确认一下，你的/home目录是不是单独的挂载在一个分区下，用df 查看即可。如果不是则需要你跟我一起做。否则这一步即可省略。

```
[root@localhost ~]# df
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/hda3        7625092    3022328    4209172  42% /
/dev/hda1        101086      11412      84455   12% /boot
tmpfs            117564         0      117564   0% /dev/shm
```

图片 8.47 8_111.png.jpg

笔者的linux系统中，/home并没有单独占用一个分区。所以需要把/home目录挂载在一个单独的分区下，因为quota是针对分区来限额的。

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

Disk /dev/hda: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1           13       104391   83  Linux
/dev/hda2             14           64       409657+   82  Linux swap / S
/dev/hda3             65          1044       7871850   83  Linux

Disk /dev/hdb: 1073 MB, 1073741824 bytes
16 heads, 63 sectors/track, 2080 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1             1           195       98248+   83  Linux
/dev/hdb2            196           390       98280   83  Linux
/dev/hdb3            391           585       98280   83  Linux
/dev/hdb4            586          2080       753480    5  Extended
/dev/hdb5            586           780       98248+   83  Linux
/dev/hdb6            781           975       98248+   83  Linux
```

图片 8.48 8_112.png.jpg

笔者用fdisk -l 查看目前/dev/hdb 磁盘有5个可用分区，所以笔者打算把/dev/hdb1挂载在/home 目录下

```
[root@localhost ~]# mount /dev/hdb1 /home
[root@localhost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/hda3       7.3G  2.9G  4.1G  42% /
/dev/hda1       99M   12M   83M  12% /boot
tmpfs           115M   0    115M   0% /dev/shm
/dev/hdb1       93M   5.6M   83M   7% /home
```

图片 8.49 8_113.png.jpg

看到了吧，目前笔者的/home目录已经是一个单独的分区了。

1) 建立测试用户

```
[root@localhost ~]# useradd test
[root@localhost ~]# grep test /etc/passwd
test:x:500:500:~/home/test:/bin/bash
[root@localhost ~]# useradd -m -g 500 test1
[root@localhost ~]# grep test /etc/passwd
test:x:500:500:~/home/test:/bin/bash
test1:x:501:500:~/home/test1:/bin/bash
```

图片 8.50 8_114.png.jpg

首先建立一个test用户，则同时建立了一个test组。可以在/etc/passwd中有以test为开头的行，其中uid和gid都为500，然后又建立一个test1账号，使其加入test组，查看/etc/passwd文件发现test和test1用户的gid都为50

0。（也许你对/etc/passwd文件、增加一个用户以及uid和gid等概念不熟悉，没有关系，在以后的章节中会做介绍，在这里只需要你明白即可）

2) 打开磁盘的quota功能

默认linux并没有对任何分区做quota的支持，所以需要我们手动打开磁盘的quota功能，你是否记得，在前面内容中分析/etc/fstab文件的第四列时讲过这个quota选项（usrquota, grpquota）。没错，要想打开这个磁盘的quota支持就是需要修改这个第四列的。用vim编辑/etc/fstab 加入一行，如下图：

```
[root@localhost ~]# vim /etc/fstab
LABEL=/                /                ext3    defaults    1 1
LABEL=/boot            /boot            ext3    defaults    1 2
tmpfs                  /dev/shm         tmpfs   defaults    0 0
devpts                 /dev/pts         devpts  gid=5,mode=620 0 0
sysfs                  /sys             sysfs   defaults    0 0
proc                   /proc            proc    defaults    0 0
LABEL=SWAP-hda2        swap             swap    defaults    0 0
/dev/hdb1              /home            ext3    defaults,usrquota,grpquota 0 0
```

图片 8.51 8_115.png.jpg

vim命令将会在后续章节详细介绍，前面介绍过如何进入编辑模式以及如何保存文件。如果你的linux系统已经有/home这一行，那么直接修改第四列，加上usrquota,grpquota（中间没有空格）。接下来需要重新挂载/home。

```
[root@localhost ~]# umount /home
[root@localhost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/hda3       7.3G  2.9G  4.1G  42% /
/dev/hda1       99M   12M   83M  12% /boot
tmpfs           115M   0    115M   0% /dev/shm
[root@localhost ~]# mount -a
[root@localhost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/hda3       7.3G  2.9G  4.1G  42% /
/dev/hda1       99M   12M   83M  12% /boot
tmpfs           115M   0    115M   0% /dev/shm
/dev/hdb1       93M   5.6M   83M   7% /home
```

图片 8.52 8_116.png.jpg

另外你也可以这样实现重新挂载/home

```
[root@localhost ~]# mount -o remount -a
```

图片 8.53 8_117.png.jpg

如何查看是否启用了quota呢？

```
[root@localhost ~]# cat /etc/mtab
/dev/hda3 / ext3 rw 0 0
proc /proc proc rw 0 0
sysfs /sys sysfs rw 0 0
devpts /dev/pts devpts rw,gid=5,mode=620 0 0
/dev/hda1 /boot ext3 rw 0 0
tmpfs /dev/shm tmpfs rw 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
/dev/hdb1 /home ext3 rw,usrquota,grpquota 0 0
```

图片 8.54 8_118.png.jpg

只要查看/etc/mtab文件中/home所在那行是否有usrquota,grpquota即可。笔者的/dev/hdb1现在已经支持了quota

3) 扫描磁盘的使用者使用状况, 并产生重要的aquota.group与aquota.user

这一步就需要用到quotacheck了, aquota.group与aquota.user分别是组以及用户磁盘配额需要的配置文件。如果没有这两个文件, 则磁盘配额是不会生效的。

```
[root@localhost ~]# quotacheck -augv
quotacheck: Scanning /dev/hdb1 [/home] quotacheck: Cannot stat old user quota file: No such file or directory
quotacheck: Cannot stat old group quota file: No such file or directory
quotacheck: Cannot stat old user quota file: No such file or directory
quotacheck: Cannot stat old group quota file: No such file or directory
done
quotacheck: Checked 15 directories and 13 files
quotacheck: Old file not found.
quotacheck: Old file not found.
[root@localhost ~]# ll /home/
total 33
-rw-r--r-- 1 root root 0 May 6 04:54 2.txt
-rw----- 1 root root 7168 May 8 21:56 aquota.group
-rw----- 1 root root 7168 May 8 21:56 aquota.user
drwx----- 2 root root 12288 May 5 07:39 lost+found
drwx----- 4 test test 1024 May 8 21:10 test
drwx----- 4 test1 test 1024 May 8 21:10 test1
```

图片 8.55 8_119.png.jpg

当首次使用quotacheck命令时, 会提示“cannot stat old user quota file ……”其实这是在提示你在/home目录下没有aquota.user以及aquota.group两个文件。没有关系, 因为以前并没有配置过磁盘配额, 当然没有这两个文件了。当执行完quotacheck命令后, 会在/home目录下生成这两个文件的。

4) 启动quota配额

```
[root@localhost ~]# quotaon -av
/dev/hdb1 [/home]: group quotas turned on
/dev/hdb1 [/home]: user quotas turned on
```

图片 8.56 8_120.png.jpg

5) 编辑用户磁盘配额

先来设定test账户的配额, 然后直接把test的配额拷贝给test1即可。这里就需要用到edquota了。

```
[root@localhost ~]# edquota -u test
Disk quotas for user test (uid 500):
  Filesystem          blocks          soft          hard          inodes          soft          hard
  /dev/hdb1            22              0             0             11              0             0
```

图片 8.57 8_121.png.jpg

讲上面内容修改为

```
Disk quotas for user test (uid 500):
  Filesystem          blocks          soft          hard          inodes          soft          hard
  /dev/hdb1            22            20000         30000         11              0             0
```

图片 8.58 8_122.png.jpg

其中单位是Kb，所以soft 值大约为20Mb，hard值为30Mb，保存这个文件，保存的方式跟vim一个文件的方式一样的。

```
[root@localhost ~]# edquota -p test test1
```

图片 8.59 8_123.png.jpg

将test的配额复制给test1。下面继续设定宽限时间。

```
[root@localhost ~]# edquota -t
Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
  Filesystem          Block grace period      Inode grace period
  /dev/hdb1            1days                   1days
```

图片 8.60 8_124.png.jpg

默认是7days 在这里我们改为1days。下面查看一下test以及test1用户的配额吧。

```
[root@localhost ~]# quota -uv test test1
Disk quotas for user test (uid 500):
  Filesystem  blocks  quota  limit  grace  files  quota  limit  grace
  /dev/hdb1   22     20000 30000          11     0      0
Disk quotas for user test1 (uid 501):
  Filesystem  blocks  quota  limit  grace  files  quota  limit  grace
  /dev/hdb1   22     20000 30000          11     0      0
```

图片 8.61 8_125.png.jpg

6) 编辑组磁盘配额

```
[root@localhost ~]# edquota -g test
Disk quotas for group test (gid 500):
  Filesystem          blocks          soft          hard          inodes          soft          hard
  /dev/hdb1            44            40000         50000         22              0             0
```

图片 8.62 8_126.png.jpg

设定组test的soft配额值为40M，hard值为50M。下面查看组test的配额。


```
[root@localhost ~]# quota -gv test
Disk quotas for group test (gid 500):
  Filesystem  blocks    quota   limit   grace   files   quota   limit   grace
  /dev/hdb1     44  40000  50000         22     0     0
```

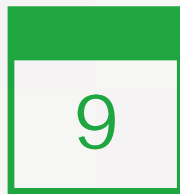
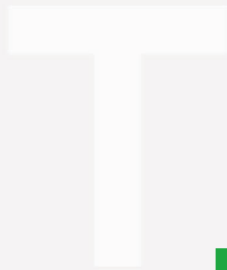
图片 8.63 8_127.png.jpg

7) 设定开机启动

前面已经讲到启动磁盘配额命令是 `quotaon -aug`，所以要想开机启动，只需将这条命令加入到 `/etc/rc.d/rc.local` 文件即可。

```
[root@localhost ~]# echo "quotaon -aug " >>/etc/rc.d/rc.local
```

图片 8.64 8_128.png.jpg



文本编辑工具 vim



前面多次提到过vim这个东西，它是linux中必不可少的一个工具。没有它很多工作都无法完成。早期的Unix都是使用的vi作为系统默认的编辑器的。你也许会有疑问，vi与vim有什么区别？可以这样简单理解，vim是vi的升级版。很多linux系统管理员都习惯用vi，那是因为他们接触linux的时候用的就是vi，vim后来才比较流行。所以，无所谓用vi和vim，只要你能达到你想要的目的即可。

在笔者看来vi和vim最大的区别就是编辑一个文本时，vi不会显示颜色，而vim会显示颜色。显示颜色更易于用户进行编辑。其他功能没有什么区别。所以在linux系统下，使用vi还是vim完全取决于你的个人爱好而已。笔者从一开始学linux就一直使用vim，所以也会一直以vim的角色来教授给你。

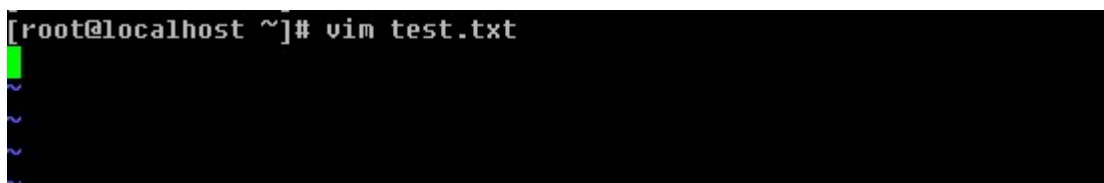
vim的三种模式：一般模式、编辑模式、命令模式。这需要你牢记的，因为以前笔者刚刚从事linux工作的时候去面试，很多单位的笔试题就有这个知识点。

* 一般模式：当你vim filename 编辑一个文件时，一进入该文件就是一般模式了。在这个模式下，你可以做的操作有，上下移动光标；删除某个字符；删除某行；复制、粘贴一行或者多行。

* 编辑模式：一般模式下，是不可以修改某一个字符的，只能到编辑模式了。从一般模式进入编辑模式，只需你按一个键即可（i,l,a,A,o,O,r,R）。当进入编辑模式时，会在屏幕的最下一行出现“INSERT或REPLACE”的字样。从编辑模式回到一般模式只需要按一下键盘左上方的ESC键即可。

* 命令模式：在一般模式下，输入“:”或者“/”即可进入命令模式。在该模式下，你可以搜索某个字符或者字符串，也可以保存、替换、退出、显示行号等等。

下面笔者教你如何在一个空白文档中写入一段文字，然后保存。



图片 9.1 9_1.png.jpg

输入vim test.txt直接回车进入一般模式。然后按“i”字母进入编辑模式



图片 9.2 9_7.png.jpg

会看到窗口的左下方出现“INSERT”字样，说明已经进入了编辑模式，此时就可以写入内容了。

```

Hello, this is my first text file.
And this is my first time to using vim.
It's easy to use vim.
I like to using vim, do you like it?
~
~
~
~
~
-- INSERT --
4,37

```

图片 9.3 9_8.png.jpg

等编辑完内容后，按ESC退出编辑模式，进入一般模式。此时在左下方的”INSERT”字样消失，然后按” :” 进入命令模式，最后输入wq保存并退出vim。

```

Hello, this is my first text file.
And this is my first time to using vim.
It's easy to use vim.
I like to using vim, do you like it?
~
~
~
~
~
:wq

```

图片 9.4 9_9.png.jpg

这时，看一下test.txt文档的内容吧。

```

[root@localhost ~]# cat test.txt
Hello, this is my first text file.
And this is my first time to using vim.
It's easy to use vim.
I like to using vim, do you like it?

```

图片 9.5 9_10.png.jpg

其实vim为全键盘操作的编辑器，所以在各个模式下都有很多功能键盘的。下面笔者列举一下，其中笔者认为常用的会用红色标出，需要你多加练习，另外不常用的你也要知道的。

一般模式下移动光标	
h或向左方向键	光标向左移动一个字符
j或者向下方向键	光标向下移动一个字符
K或者向上方向键	光标向上移动一个字符
l或者向右方向键	光标向右移动一个字符
Ctrl + f 或者pageUP键	屏幕向前移动一页
Ctrl + b 或者pageDOWN键	屏幕向后移动一页
Ctrl + d	屏幕向前移动半页
Ctrl + u	屏幕向后移动半页

+	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空格 (n是数字)	按下数字n然后按空格, 则光标向右移动n个字符, 如果该行字符数小于n, 则光标继续从下行开始向右移动, 一直到n
0 (数字0) 或者Shift+6	移动到本行行首
Shift+4	即'\$'移动到本行行尾
H	光标移动到当前屏幕的最顶行
M	光标移动到当前屏幕的中央那一行
L	光标移动到当前屏幕的最底行
G	光标移动到文本的最末行
nG (n是数字)	移动到该文本的第n行
gg	移动带该文本的首行
n回车 (n是数字)	光标向下移动n行

一般模式下查找与替换

/word	向光标之后寻找一个字符串名为word的字符串, 当找到第一个word后, 按"n"继续搜后一个
?word	向光标之前寻找一个字符串名为word的字符串, 当找到第一个word后, 按"n"继续搜前一个
:n1,n2s/word1/word2/g	在n1和n2行间查找word1这个字符串并替换为word2, 你也可以把"/"换成"#"
:1,\$s/word1/word2/g	从第一行到最末行, 查找word1并替换成word2
:1,\$s/word1/word2/gc	加上c的作用是, 在替换前需要用户确认

一般模式下删除复制粘贴

x,X	x为向后删除一个字符, X为向前删除一个字符
nx (n为数字)	向后删除n个字符
dd	删除光标所在的那一行
ndd (n为数字)	删除光标所在的向下n行
d1G	删除光标所在行到第一行的所有数据
dG	删除光标所在行到末行的所有数据
yy	复制光标所在的那一行
nyy	复制从光标所在行起向下n行
p,P	p复制的数据从光标下一行粘贴, P则从光标上一行粘贴
y1G	复制光标所在行到第一行的所有数据
yG	复制光标所在行到末行的所有数据
J	讲光标所在行与下一行的数据结合成同一行
u	还原过去的操作

进入编辑模式

i	在当前字符前插入字符
I	在当前行行首插入字符
a	在当前字符后插入字符
A	在当前行行末插入字符
o	在当前行下插入新的一行
O	在当前行上插入新的一行

r	替换光标所在的字符，只替换一次
R	一直替换光标所在的字符，一直到按下ESC

命令模式	
------	--

:w	将编辑过的文本保存
:w!	若文本属性为只读时，强制保存
:q	退出vim
:q!	不管编辑或未编辑都不保存退出
:wq	保存，退出
:e!	将文档还原成最原始状态
ZZ	若文档没有改动，则不储存离开，若文档改动过，则储存后离开，等同于:wq
:w [filename]	编辑后的文档另存为filename
:r [filename]	在当前光标所在行的下面读入filename文档的内容
:set nu	在每行的行首显示行号
:set nonu	取消行号
n1,n2 w [filename]	将n1到n2的内容另存为filename这个文档
:! command	暂时离开vim运行某个linux命令，例如 :! ls /home 暂时列出/home目录下的文件，然后会提示按回车回到vim

暂时就讲这么多了。如果你全部掌握，你就是vim高手啦。如果你觉得太多，只要记住笔者标红部分即可，其他的用时再过来查就ok啦。下面笔者给你留一个小作业，希望你认真完成！

1. 请把/etc/init.d/iptables 复制到/root/目录下，并重命名为test.txt
2. 用vim打开test.txt并设置行号
3. 分别向下、向右、向左、向右移动5个字符
4. 分别向下、向上翻两页
5. 把光标移动到第49行
6. 让光标移动到行末，再移动到行首
7. 移动到test.txt文件的最后一行
8. 移动到文件的首行
9. 搜索文件中出现的 iptables 并数一下一共出现多少个
10. 把从第一行到第三行出现的iptables 替换成iptables
11. 还原上一步操作
12. 把整个文件中所有的iptables替换成iptables

13. 把光标移动到50行，删除字符” \$”
14. 还原上一步操作
15. 删除第50行
16. 还原上一步操作
17. 删除从37行到42行的所有内容
18. 还原上一步操作
19. 复制48行并粘贴到52行下面
20. 还原上一步操作（按两次u）
21. 复制从37行到42行的内容并粘贴到44行上面
22. 还原上一步操作（按两次u）
23. 把37行到42行的内容移动到19行下面
24. 还原上一步操作（按两次u）
25. 光标移动到首行，把/bin/sh 改成 /bin/bash
26. 在第一行下面插入新的一行，并输入” # Hello!”
27. 保存文档并退出



T



10

文档的压缩与打包



在windows下我们接触最多的压缩文件就是.rar格式的了。但在linux下这样的格式是不能识别的，它有自己的压缩工具。但有一种文件在windows和linux下都能使用那就是.zip格式的文件了。压缩的好处不用笔者介绍相信你也晓得吧，它不仅能节省磁盘空间而且在传输的时候还能节省网络带宽呢。

在linux下最常见的压缩文件通常都是以.tar.gz 为结尾的，除此之外还有.tar, .gz, .bz2, .zip等等。以前也介绍过linux系统中的后缀名其实要不要无所谓，但是对于压缩文件来讲必须要带上。这是为了判断压缩文件是由哪种压缩工具所压缩，而后才能去正确的解压缩这个文件。以下介绍常见的后缀名所对应的压缩工具。

.gz gzip 压缩工具压缩的文件

.bz2 bzip2 压缩工具压缩的文件

.tar tar 打包程序打包的文件(tar并没有压缩功能，只是把一个目录合并成一个文件)

.tar.gz 可以理解为先用tar打包，然后再gzip压缩

.tar.bz2 同上，先用tar打包，然后再bzip2压缩

【gzip】

语法：gzip [-d#] filename 其中#为1-9的数字

-d：解压缩时使用

-#：压缩等级，1压缩最差，9压缩最好，6为默认

```
[root@localhost ~]# mkdir test
[root@localhost ~]# cd test
[root@localhost ~]# mv test.txt test
[root@localhost ~]# cd test
[root@localhost test]# ls
test.txt
[root@localhost test]# gzip test.txt
[root@localhost test]# ls
test.txt.gz
```

图片 10.1 10_1.png.jpg

压缩test.txt后，则变成了test.txt.gz

```
[root@localhost test]# gzip -d test.txt.gz
[root@localhost test]# ls
test.txt
```

图片 10.2 10_7.png.jpg

用-d解压缩

要注意的是，gzip不可以压缩目录

```
[root@localhost ~]# gzip test
gzip: test is a directory -- ignored
```

图片 10.3 10_8.png.jpg

【bzip2】

语法: bzip2 [-dz] filename

-d : 解压缩

-z : 压缩

```
[root@localhost test]# bzip2 -z test.txt
[root@localhost test]# ls
test.txt.bz2
```

图片 10.4 10_9.png.jpg

其实-z参数是可以省略掉的,你不妨试试

```
[root@localhost test]# bzip2 -d test.txt.bz2
[root@localhost test]# ls
test.txt
```

图片 10.5 10_10.png.jpg

跟gzip的解压类似,也是用-d解压。

【tar】

语法: tar [-zjxcvfpP] filename

-z : 是否同时用gzip压缩

-j : 是否同时用bzip2压缩

-x : 解包或者解压缩

-t : 查看tar包里面的文件

-c : 建立一个tar包或者压缩文件包

-v : 可视化

-f : 后面跟文件名,压缩时跟-f文件名,意思是压缩后的文件名为filename,解压时跟-f文件名,意思是解压filename。请注意,如果是多个参数组合的情况下带有-f,请把f写到最后面。

-p : 使用原文件的属性,压缩前什么属性压缩后还什么属性。(不常用)

-P：可以使用绝对路径。（不常用）

--exclude filename：在打包或者压缩时，不要将filename文件包括在内。（不常用）

```
[root@localhost test]# mkdir test111
[root@localhost test]# touch test111/test2.txt
[root@localhost test]# echo "nihao">test111/test2.txt
[root@localhost test]# ls
test111  test.txt.bz2
[root@localhost test]# tar -cvf test111.tar test111
test111/
test111/test2.txt
[root@localhost test]# ls
test111  test111.tar  test.txt.bz2
```

图片 10.6 10_11.png.jpg

首先在test目录下建立test111目录，然后在test111目录下建立test2.txt，并写入”nihao”到test2.txt中，接着是用tar把test111打包成test111.tar。请记住-f参数后跟的是打包后的文件名。

```
[root@localhost test]# rm -rf test111
[root@localhost test]# ls
test111.tar  test.txt.bz2
[root@localhost test]# tar -xvf test111.tar
test111/
test111/test2.txt
[root@localhost test]# ls
test111  test111.tar  test.txt.bz2
```

图片 10.7 10_21.png.jpg

删除原来的test111目录，然后解包test111.tar，不管是打包还是解包，原来的文件是不会删除的。

```
[root@localhost test]# tar -zcvf test111.tar.gz test111
test111/
test111/test2.txt
[root@localhost test]# ls
test111  test111.tar  test111.tar.gz  test.txt.bz2
```

图片 10.8 10_22.png.jpg

打包的同时使用gzip压缩

```
[root@localhost test]# tar -tf test111.tar.gz
test111/
test111/test2.txt
[root@localhost test]# tar -tf test111.tar
test111/
test111/test2.txt
```

图片 10.9 10_23.png.jpg

用-tf 跟包名来查看包或者压缩包内的文件都有哪些

```
[root@localhost test]# rm -rf test111
[root@localhost test]# ls
test111.tar test111.tar.gz test.txt.bz2
[root@localhost test]# tar -zxvf test111.tar.gz
test111/
test111/test2.txt
[root@localhost test]# ls
test111 test111.tar test111.tar.gz test.txt.bz2
```

图片 10.10 10_24.png.jpg

先删除test111,然后用tar -zxvf 来解压.tar.gz的压缩包。

```
[root@localhost test]# tar -jcvf test111.tar.bz2 test111
test111/
test111/test2.txt
[root@localhost test]# ls
test111 test111.tar test111.tar.bz2 test111.tar.gz test.txt.bz2
[root@localhost test]# tar -tf test111.tar.bz2
test111/
test111/test2.txt
```

图片 10.11 10_25.png.jpg

-jcvf 打包的同时用bzip2压缩, -tf同样可以查看.tar.bz2的压缩包

```
[root@localhost test]# rm -rf test111
[root@localhost test]# tar -jxvf test111.tar.bz2
test111/
test111/test2.txt
[root@localhost test]# ls
test111 test111.tar test111.tar.bz2 test111.tar.gz test.txt.bz2
```

图片 10.12 10_26.png.jpg

-jxvf解压缩.tar.bz2的压缩包

```
[root@localhost test]# rm -rf test111.tar
[root@localhost test]# tar -cvf test111.tar test111 --exclude test2.txt
test111/
test111/test3.txt
[root@localhost test]# tar -tf test111.tar
test111/
test111/test3.txt
[root@localhost test]# tar -cvf test111-2.tar test111
test111/
test111/test2.txt
test111/test3.txt
[root@localhost test]# tar -tf test111-2.tar
test111/
test111/test2.txt
test111/test3.txt
```

图片 10.13 10_27.png.jpg

--exclude参数的作用就是打包的时候过滤掉某些文件, 如果想过滤多个文件怎么办

```
[root@localhost test]# tar -cvf test111-3.tar test111 --exclude test2.txt --exclude test3.txt test111/
```

图片 10.14 10_28.png.jpg

只能是继续跟 --exclude filename了。



安装 RPM 包或者安装源码包



在windows下安装一个软件很轻松，只要双击.exe的文件，安装提示连续“下一步”即可，然而linux系统下安装一个软件似乎并不那么轻松了，因为我们不是在图形界面下。所以你要学会如何在linux下安装一个软件。

在前面的内容中多次提到的yum，这个yum是Redhat所特有的安装RPM程序包的工具，使用起来相当方便。因为使用RPM安装某一个程序包有可能会因为该程序包依赖另一个程序包而无法安装。而使用yum工具就可以连同依赖的程序包一起安装。当然CentOS同样可以使用yum工具，而且在CentOS中你可以免费使用yum，但Red hat中只有当你付费后才能使用yum，默认是无法使用yum的。在介绍yum之前先说一说RPM相关的东西。

【RPM工具】

RPM是” Redhat Package Manager” 的缩写，根据名字也能猜到这是Redhat公司开发出来的。RPM 是以一种数据库记录的方式来将你所需要的套件安装到你的Linux 主机的一套管理程序。也就是说，你的linux系统中存在着一个关于RPM的数据库，它记录了安装的包以及包与包之间依赖相关性。RPM包是预先在linux机器上编译好并打包好的文件，安装起来非常快捷。但是也有一些缺点，比如安装的环境必须与编译时的环境一致或者相当；包与包之间存在着相互依赖的情况；卸载包时需要先把依赖的包卸载掉，如果依赖的包是系统所必须的，那就不能卸载这个包，否则会造成系统崩溃。

如果你的光驱中还有系统安装盘的话，你可以通过” mount /dev/cdrom /mnt” 命令把光驱挂载到/mnt目录下，那么你会在/mnt/CentOS目录下看到很多.rpm的文件，这就是RPM包了。

```
[root@localhost test]# ls /mnt/CentOS/ | less
a2ps-4.13b-57.2.el5.i386.rpm
acl-2.2.39-3.el5.i386.rpm
acpid-1.0.4-9.el5.i386.rpm
adaptx-0.9.13-3jpp.1.i386.rpm
adaptx-doc-0.9.13-3jpp.1.i386.rpm
adaptx-javadoc-0.9.13-3jpp.1.i386.rpm
adjtimex-1.20-2.1.i386.rpm
agg-2.4-2.1.i386.rpm
agg-devel-2.4-2.1.i386.rpm
aide-0.13.1-4.el5.i386.rpm
alacarte-0.10.0-1.fc6.noarch.rpm
```

图片 11.1 11_1.png.jpg

每一个rpm包的名称都由” - “和” .” 分成了若干部分。就拿 a2ps-4.13b-57.2.el5.i386.rpm 这个包来解释一下，a2ps 为包名；4.13b则为版本信息；57.2.el5为发布版本号；i386为运行平台。其中运行平台常见的有i386, i586, i686, x86_64，需要你注意的是cpu目前是分32位和64位的，i386,i586和i686都为32位平台，x86_64则代表为64位的平台。另外有些rpm包并没有写具体的平台而是noarch，这代表这个rpm包没有硬件平台限制。例如 alacarte-0.10.0-1.fc6.noarch.rpm 。下面介绍一下rpm常用的命令。

1) 安装一个rpm包

```
[root@localhost CentOS]# rpm -ivh alacarte-0.10.0-1.fc6.noarch.rpm
Preparing... ##### [100%]
 1:alacarte ##### [100%]
```

图片 11.2 11_7.png.jpg

-i : 安装的意思

-v : 可视化

-h : 显示安装进度

另外在安装一个rpm包时常用的附带参数有:

--force 强制安装, 即使覆盖属于其他包的文件也要安装

--nodeps 当要安装的rpm包依赖其他包时, 即使其他包没有安装, 也要安装这个包

2) 升级一个rpm包

rpm -Uvh filename -U : 即升级的意思

3) 卸载一个rpm包

rpm -e filename 这里的filename是通过rpm的查询功能所查询到的, 稍后会作介绍。

```
[root@localhost CentOS]# rpm -qa |grep alacarte
alacarte-0.10.0-1.fc6
[root@localhost CentOS]# rpm -e alacarte-0.10.0-1.fc6
```

图片 11.3 11_8.png.jpg

卸载时后边跟的filename和安装时的是有区别的。上面命令提到的“|”在linux系统中用的非常多也非常有用, 它是一个管道符, 用来把前面运行的结果传递给后面的命令。以后会做详细介绍, 而后出现的grep命令则是用来过滤某个关键词的工具, 在后续章节中会做详细介绍。

4) 查询一个包是否安装

rpm -q rpm包名 (这里的包名, 是不带有平台信息以及后缀名的)

```
[root@localhost CentOS]# rpm -q alacarte-0.10.0-1.fc6
alacarte-0.10.0-1.fc6
[root@localhost CentOS]# rpm -q alacarte-0.10.0-1.fc6.noarch.rpm
package alacarte-0.10.0-1.fc6.noarch.rpm is not installed
```

图片 11.4 11_9.png.jpg

如果加上了平台信息以及后缀名反而不能查出来。你还可以查询当前系统中所安装的所有rpm包。


```
[root@localhost CentOS]# rpm -qa |head
kernel-headers-2.6.18-164.el5
gnome-mime-data-2.4.2-3.1
xorg-x11-util-macros-1.0.2-4.fc6
centos-release-notes-5.4-4
bsh-manual-1.3.0-9jpp.1
zlib-1.2.3-3
popt-1.10.2.3-18.el5
libSM-1.0.1-3.1
expat-1.95.8-8.2.1
libusb-0.1.12-5.1
```

图片 11.5 11_10.png.jpg

因为太多，所以笔者列出前十个。

5) 得到一个rpm包的相关信息

rpm -qi 包名 (同样不需要加平台信息与后缀名)

```
[root@localhost CentOS]# rpm -qi gnome-mime-data-2.4.2-3.1
Name       : gnome-mime-data           Relocations: (not relocatable)
Version    : 2.4.2                     Vendor: CentOS
Release    : 3.1                   Build Date: Sat 06 Jan 2007 01:33:50 PM CST
Install Date: Wed 27 Apr 2011 12:22:05 AM CST   Build Host: builder1.centos.org
Group      : System Environment/Libraries   Source RPM: gnome-mime-data-2.4.2-3.1.src.rpm
Size       : 3558958                License: GPL
Signature  : DSA/SHA1, Wed 04 Apr 2007 08:22:31 AM CST, Key ID a8a447dce8562897
URL        : http://www.gnome.org
Summary    : MIME type data files for GNOME desktop
Description:
gnome-mime-data provides the file type recognition data files for gnome-vfs
```

图片 11.6 11_11.png.jpg

6) 列出一个rpm包安装的文件

rpm -ql 包名

```
[root@localhost CentOS]# rpm -ql vim-enhanced-7.0.109-6.el5
/etc/profile.d/vim.csh
/etc/profile.d/vim.sh
/usr/bin/ex
/usr/bin/rvim
/usr/bin/vim
/usr/bin/vimdiff
/usr/bin/vimtutor
/usr/share/man/man1/rvim.1.gz
/usr/share/man/man1/vimdiff.1.gz
/usr/share/man/man1/vimtutor.1.gz
```

图片 11.7 11_21.png.jpg

通过上面的命令可以看出vim是通过安装vim-enhanced-7.0.109-6.el5这个rpm包得来的。那么反过来如何通过一个文件去查找是由安装哪个rpm包得来的?

7) 列出某一个文件属于哪个rpm包

rpm -qf 文件的绝对路径

```
[root@localhost CentOS]# rpm -qf /usr/bin/vim
vim-enhanced-7.0.109-6.el5
```

图片 11.8 11_22.png.jpg

前面讲过如何查找一个文件（可执行命令）的绝对路径

```
[root@localhost CentOS]# which vim
/usr/bin/vim
```

图片 11.9 11_23.png.jpg

所以你也可以把这两条命令连起来写

```
[root@localhost CentOS]# rpm -qf `which vim`
vim-enhanced-7.0.109-6.el5
```

图片 11.10 11_24.png.jpg

看到了吗，which vim 这条命令是由两个反引号引起来的，这代表引用反引号里面的命令所产生的结果。关于rpm工具的使用还有很多内容，笔者就不一一列举了，只要你掌握上面这些内容，完全够你平时工作用的了。

【yum工具】

介绍完rpm工具后，还需要你掌握最常用的yum工具，这个工具比rpm工具好用多了，当然前提是你使用的linux系统是支持yum的。yum最大的优势在于可以联网去下载所需要的rpm包，然后自动安装，在这个工程中如果要安装的rpm包有依赖关系，yum会帮你解决掉这些依赖关系依次安装所有rpm包。下面笔者介绍常用的yum命令。

1) 列出所有可用的rpm包 “yum list “

```
[root@5d6d-web-pp-1fctc ~]# yum list |head -n 15
Loading "fastestmirror" plugin
Loading "downloadonly" plugin
Loading mirror speeds from cached hostfile
 * addons: mirrors.163.com
 * base: mirrors.163.com
 * update: mirrors.163.com
 * extras: mirrors.163.com
Installed Packages
Agent.x86_64                1.0-0                installed
GConf2.x86_64              2.14.0-9.el5        installed
GConf2.i386                 2.14.0-9.el5        installed
MAKEDEV.x86_64             3.23-1.2            installed
MegaCli.i386                2.00.12-1           installed
NetworkManager.x86_64     1:0.6.4-8.el5       installed
ORBit2.i386                 2.14.3-4.el5        installed
```

图片 11.11 11_25.png.jpg

限于篇幅，笔者只列举出来前7个包信息。从上例中可以看到有“mirrors.163.com”信息出现，这是在告诉用户，它是从mirrors.163.com这里下载到的rpm包资源。如果你使用的是CentOS则你可以从/etc/yum.repos.d/CentOS-Base.repo这个文件下看到相关的配置信息。从上面的例子中你还可以看到最左侧是rpm包名字，中间是版本信息，最右侧是安装信息，如果安装了就显示installed，未安装则显示base或者extras，如果是该rpm包已安装但需要升级则显示updates。

2) 搜索一个rpm包 “yum search [相关关键词]”

```
[root@5d6d-web-pp-1fctc ~]# yum search vim
Loading "fastestmirror" plugin
Loading "downloadonly" plugin
Loading mirror speeds from cached hostfile
 * addons: mirrors.163.com
 * base: mirrors.163.com
 * update: mirrors.163.com
 * extras: mirrors.163.com
vim-enhanced.x86_64 : A version of the VIM editor which includes recent enhancements.
vim-minimal.x86_64 : A minimal version of the VIM editor.
vim-X11.x86_64 : The VIM version of the vi editor for the X Window System.
vim-common.x86_64 : The common files needed by any version of the VIM editor.
vim-common.x86_64 : The common files needed by any version of the VIM editor.
vim-enhanced.x86_64 : A version of the VIM editor which includes recent enhancements.
vim-minimal.x86_64 : A minimal version of the VIM editor.
```

图片 11.12 11_26.png.jpg

除了这样搜索外，笔者常用的是利用grep来过滤

```
[root@5d6d-web-pp-1fctc ~]# yum list |grep vim
vim-common.x86_64                2:7.0.109-3.e15.3      installed
vim-enhanced.x86_64             2:7.0.109-3.e15.3      installed
vim-minimal.x86_64              2:7.0.109-3.e15.3      installed
vim-X11.x86_64                  2:7.0.109-7.e15        base
vim-common.x86_64                2:7.0.109-7.e15        base
vim-enhanced.x86_64             2:7.0.109-7.e15        base
vim-minimal.x86_64              2:7.0.109-7.e15        base
```

图片 11.13 11_27.png.jpg

相信你会也喜欢用后者吧，这样看起来简明的多。

3) 安装一个rpm包 “yum install [-y] [rpm包名]”

如果不加-y选项，则会以与用户交互的方式安装，首先是列出需要安装的rpm包信息，然后会问用户是否需要安装，输入y则安装，输入n则不安装。而笔者嫌这样太麻烦，所以直接加上-y选项，这样就省略掉了问用户是否安装的那一步。

```

[root@5d6d-web-pp-1fctc ~]# yum install vim-common
Loading "fastestmirror" plugin
Loading "downloadonly" plugin
Loading mirror speeds from cached hostfile
 * addons: mirrors.163.com
 * base: mirrors.163.com
 * update: mirrors.163.com
 * extras: mirrors.163.com
Setting up Install Process
Parsing package install arguments
Resolving Dependencies
--> Running transaction check
--> Processing Dependency: vim-common = 2:7.0.109-3.el5.3 for package: vim-enhanced
---> Package vim-common.x86_64 2:7.0.109-7.el5 set to be updated
--> Running transaction check
---> Package vim-enhanced.x86_64 2:7.0.109-7.el5 set to be updated
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                Arch      Version      Repository    Size
=====
Updating:
vim-common              x86_64    2:7.0.109-7.el5  base          6.4 M
Updating for dependencies:
vim-enhanced            x86_64    2:7.0.109-7.el5  base          1.3 M

Transaction Summary
=====
Install      0 Package(s)
Update      2 Package(s)
Remove      0 Package(s)

Total download size: 7.7 M
Is this ok [y/N]: █

```

图片 11.14 11_28.png.jpg

4) 卸载一个rpm包 “yum remove [-y] [rpm包名]”

```

[root@localhost ~]# yum remove vim-common
Loaded plugins: fastestmirror
Setting up Remove Process
Resolving Dependencies
--> Running transaction check
--> Package vim-common.i386 2:7.0.109-7.el5 set to be erased
--> Processing Dependency: vim-common = 2:7.0.109-7.el5 for package: vim-enhanced
--> Running transaction check
--> Package vim-enhanced.i386 2:7.0.109-7.el5 set to be erased
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                Arch          Version           Repository        Size
=====
Removing:
vim-common              i386          2:7.0.109-7.el5  installed        15 M
Removing for dependencies:
vim-enhanced            i386          2:7.0.109-7.el5  installed        2.6 M

Transaction Summary
=====
Install      0 Package(s)
Update      0 Package(s)
Remove      2 Package(s)

Is this ok [y/N]: █

```

图片 11.15 11_29.png.jpg

卸载和安装一样，你也可以直接加上 `-y` 选项来省略掉和用户交互的步骤。在这里笔者要提醒你一下，卸载某个 rpm 包一定要看清楚了，不要连其他重要的 rpm 包一起卸载了，以免影响正常的业务。

4) 升级一个 rpm 包 “`yum update [-y] [rpm包]`”

```

[root@localhost ~]# yum update libsane-hpaio
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * addons: centos.ustc.edu.cn
 * base: centos.ustc.edu.cn
 * extras: centos.ustc.edu.cn
 * updates: centos.ustc.edu.cn
Setting up Update Process
Resolving Dependencies
--> Running transaction check
---> Package libsane-hpaio.i386 0:1.6.7-6.e15_6.1 set to be updated
--> Processing Dependency: hplip = 1.6.7-6.e15_6.1 for package: lib
sane-hpaio
--> Running transaction check
---> Package hplip.i386 0:1.6.7-6.e15_6.1 set to be updated
--> Processing Dependency: hpijs = 1:1.6.7-6.e15_6.1 for package: h
plip
--> Running transaction check
---> Package hpijs.i386 1:1.6.7-6.e15_6.1 set to be updated
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                Arch      Version                Repository  Size
=====
Updating:
libsane-hpaio          i386      1.6.7-6.e15_6.1       updates    58 k
Installing for dependencies:
hplip                  i386      1.6.7-6.e15_6.1       updates    7.8 M
Updating for dependencies:
hpijs                  i386      1:1.6.7-6.e15_6.1     updates    222 k

Transaction Summary
=====
Install      1 Package(s)
Update      2 Package(s)
Remove      0 Package(s)

Total download size: 8.1 M
Is this ok [y/N]: █

```

图片 11.16 11_44.png.jpg

以上介绍了如何使用yum搜索、安装、卸载以及升级一个rpm包，如果你掌握了这些那么你就已经可以解决日常工作中遇到的与rpm包相关问题了。当然yum工具还有好多其他好用的命令，笔者不在列举出来，如果你感兴趣就去man一下吧。除此之外，笔者还会教你一些关于yum的小应用。

1 使用本地的光盘来制作一个yum源

有时候你的linux系统不能联网，当然就不能很便捷的使用联网的yum源了，这时候就需要你自己会利用linux系统光盘制作一个yum源。具体步骤如下：

a. 挂载光盘

```
[root@fortest Server]# mount -t iso9660 -o loop /dev/cdrom /mnt
```

b. 删除/etc/yum.repos.d目录所有的repo文件

```
[root@fortest Server]# rm -rf /etc/yum.repos.d/*
```

c. 创建新文件dvd.repo

```
[root@fortest Server]# vim /etc/yum.repos.d/dvd.repo
```

加入以下内容:

```
[dvd]
```

```
name=install dvd
```

```
baseurl=file:///mnt
```

```
enabled=1
```

```
gpgcheck=0
```

d. 刷新repos,生成缓存

```
[root@fortest Server]#yum makecache
```

然后就可以使用yum命令安装你所需要的软件包了

2 利用yum工具下载一个rpm包

有时, 我们需要下载一个rpm包, 只是下载下来, 拷贝给其他机器使用, 前面也介绍过yum安装rpm包的时候, 首先得下载这个rpm包然后再去安装, 所以使用yum完全可以做到只下载而不安装。

a. 首选要安装 yum-downloadonly

```
# yum install -y yum-downloadonly.noarch
```

b. 下载一个rpm包而不安装

```
# yum install test.rpm -y --downloadonly //这样虽然下载了, 但是并没有保存到我们想要的目录下, 那么如何指定目录呢?
```

c. 下载到指定目录

```
# yum install test.rpm -y --downloadonly --downloadaddir=/usr/local/src
```

```
[root@localhost ~]# yum install zsh.i386 -y --downloadonly --downloadaddir=/usr/local/src
Loaded plugins: downloadonly, fastestmirror
Loading mirror speeds from cached hostfile
 * addons: centos.ustc.edu.cn
 * base: centos.ustc.edu.cn
 * extras: centos.ustc.edu.cn
 * updates: centos.ustc.edu.cn
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package zsh.i386 0:4.2.6-5.el5 set to be updated
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                               Arch
=====
Installing:
zsh                                     i386

Transaction Summary
=====
Install      1 Package(s)
Update      0 Package(s)
Remove      0 Package(s)

Total size: 1.7 M
Downloading Packages:

exiting because --downloadonly specified
[root@localhost ~]# ls /usr/local/src/
zsh-4.2.6-5.el5.i386.rpm
```

图片 11.17 11_45.png.jpg

【安装源码包】

其实，在linux下面安装一个源码包是最常用的，笔者在日常的管理工作中，大部分软件都是通过源码安装的。安装一个源码包，是需要我们自己去把源代码编译成二进制的可执行文件。如果你读得懂这些源代码，那么你就可以去修改这些源代码自定义功能，然后再去编译成你想要的。使用源码包的好处除了可以自定义修改源代码外还可以定制相关的功能，因为源码包在编译的时候是可以附加额外的选项的。

源码包的编译用到了linux系统里的编译器，常见的源码包一般都是用C语言开发的，这也是因为C语言为linux上最标准的程序语言。Linux上的C语言编译器叫做gcc，利用它就可以把C语言变成可执行的二进制文件。所以如果你的机器上没有安装gcc就没有办法去编译源码。你可以使用 `yum install -y gcc` 来完成安装。

安装一个源码包，通常需要三个步骤：

1. ./config 在这一步可以定制功能，加上相应的选项即可，具有有什么选项可以通过” ./config --help ” 命令来查看。在这一步会自动检测你的linux系统与相关的套件是否有编译该源码包时需要的库，因为一旦缺少某个库就不能完成编译。只有检测通过后才会生成一个Makefile文件。

2. make 使用这个命令会根据Makefile文件中预设的参数进行编译，这一步其实就是gcc在工作了。

3. make install 安装步骤，生成相关的软件存放目录和配置文件的过程。

上面介绍的3步并不是所有的源码包软件都一样的，笔者以前也曾经遇到过，安装步骤并不是这样，也就是说源码包的安装并非具有一定的标准安装步骤。这就需要你拿到源码包解压后，然后进入到目录找相关的帮助文档，通常会以INSTALL或者README为文件名。所以，你一定要去看一下。下面笔者会编译安装一个源码包来帮你更深刻的去理解如何安装源码包。

1. 下载一个源码包

```
[root@localhost src]# cd /usr/local/src/
[root@localhost src]# wget http://syslab.comsenz.com/downloads/linux/httpd-2.2.16.tar.gz
```

图片 11.18 11_46.png.jpg

这里要提一下，建议以后你把所有下载的源码包放到/usr/local/src/目录下，这个并不是必须的，只是一个约定。方便你和你的同事将来更好的去运维这台服务器。wget即为下载的命令，后边跟源码包的下载地址。该地址为笔者从网上找的一个apache的下载地址。

2. 解压源码包

```
[root@localhost src]# tar zxvf httpd-2.2.16.tar.gz
```

图片 11.19 11_47.png.jpg

一般的源码包都是一个压缩包，如何解压一个.tar.gz的包上一章讲过的。

3. 配置相关的选项，并生成Makefile

```
[root@localhost src]# cd httpd-2.2.16
[root@localhost httpd-2.2.16]# ./configure --help |less
'configure' configures this package to adapt to many kinds of systems.

Usage: ./configure [OPTION]... [VAR=VALUE]...

To assign environment variables (e.g., CC, CFLAGS...), specify them as
VAR=VALUE.  See below for descriptions of some of the useful variables.

Defaults for the options are specified in brackets.

Configuration:
  -h, --help                display this help and exit
  --help=short              display options specific to this package
```

图片 11.20 11_48.png.jpg

使用 `./config --help` 可以查看可用的选项。一般常用的有 `--prefix=PREFIX` “这个选项的意思是定义软件包安装到哪里。到这里，笔者再提一个小小的约定，通常源码包都是安装在 `/usr/local/` 目录下的。比如，我们把 `apache` 安装在 `/usr/local/apache2` 下，那么这里就应该这样写 `--prefix=/usr/local/apache2`”。其他还有好多选项，如果你有耐心你可以挨个去看一看都有什么作用。

```
[root@localhost httpd-2.2.16]# ./configure --prefix=/usr/local/apache2
checking for chosen layout... Apache
```

图片 11.21 11_49.png.jpg

笔者在这里只定义了 `apache` 的安装目录，其他都是默认。回车后，开始执行 `check` 操作。

```
checking os2.h usability... no
checking os2.h presence... no
checking for os2.h... no
checking windows.h usability... no
checking windows.h presence... no
checking for windows.h... no
checking for mmap... yes
checking for munmap... yes
checking for shm_open... yes
checking for shm_unlink... yes
checking for shmget... yes
checking for shmat... yes
checking for shmdt... yes
```

图片 11.22 11_50.png.jpg

等 `check` 结束后生成了 `Makefile` 文件

```
[root@localhost httpd-2.2.16]# ls -l Makefile
-rw-r--r-- 1 root root 8954 May 15 08:49 Makefile
```

图片 11.23 11_51.png.jpg

除了查看有没有生成 `Makefile` 文件来判定有没有完成 `./config` 的操作外，还可以通过这个命令 `echo $?` 来判定，如果是 0，则表示上一步操作成功完成，否则就是没有成功。

```
[root@localhost httpd-2.2.16]# echo $?
0
```

图片 11.24 11_52.png.jpg

4. 进行编译

```
[root@localhost httpd-2.2.16]# make
Making all in srclib
make[1]: Entering directory `/usr/local/src/httpd-2.2.16/srclib'
Making all in apr
make[2]: Entering directory `/usr/local/src/httpd-2.2.16/srclib/apr'
make[3]: Entering directory `/usr/local/src/httpd-2.2.16/srclib/apr'
/bin/sh /usr/local/src/httpd-2.2.16/srclib/apr/libtool --silent --mode=compile
cc -g -O2 -pthread -DHAVE_CONFIG_H -DLINUX=2 -D_REENTRANT -D_GNU_SOURCE -D_LA
GEFILE64_SOURCE -I./include -I/usr/local/src/httpd-2.2.16/srclib/apr/include/
rch/unix -I./include/arch/unix -I/usr/local/src/httpd-2.2.16/srclib/apr/include
arch/unix -I/usr/local/src/httpd-2.2.16/srclib/apr/include -o passwd/apr_getpa
s.lo -c passwd/apr/getpass.c && touch passwd/apr/getpass.lo
```

图片 11.25 11_53.png.jpg

这一步操作，就是把源代码编译成二进制的可执行文件，这一步也是最漫长的一步，编译时间的长短取决于源代码的多少和机器配置。

5. 安装

```
make[1]: Leaving directory `/usr/local/src/httpd-2.2.16'
[root@localhost httpd-2.2.16]# echo $?
0
```

图片 11.26 11_54.png.jpg

在安装前，先确认上一步操作是否成功完成。

```
[root@localhost httpd-2.2.16]# make install
Making install in srclib
make[1]: Entering directory `/usr/local/src/httpd-2.2.16/srclib'
Making install in apr
make[2]: Entering directory `/usr/local/src/httpd-2.2.16/srclib/apr'
make[3]: Entering directory `/usr/local/src/httpd-2.2.16/srclib/apr'
make[3]: Nothing to be done for `local-all'.
make[3]: Leaving directory `/usr/local/src/httpd-2.2.16/srclib/apr'
/usr/local/src/httpd-2.2.16/srclib/apr/build/mkdir.sh /usr/local/apac
r/local/apache2/bin /usr/local/apache2/build \
        /usr/local/apache2/lib/pkgconfig /usr/local/apac
mkdir /usr/local/apache2
mkdir /usr/local/apache2/lib
mkdir /usr/local/apache2/bin
mkdir /usr/local/apache2/build
```

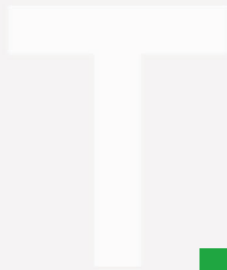
图片 11.27 11_55.png.jpg

make install 会创建相应的目录以及文件。当完成安装后，会在/usr/local目录下多了一个apache2目录，这就是apache所安装的目录了。

```
[root@localhost httpd-2.2.16]# ls -ld /usr/local/apache2/
drwxr-xr-x 15 root root 4096 May 15 09:01 /usr/local/apache2/
```

图片 11.28 11_56.png.jpg

其实在日常的源码安装工作中，并不是每个都像笔者这样顺利完成安装的，遇到错误不能完成安装的情况是很多的。通常都是因为缺少某一个库文件导致的。这就需要你仔细琢磨报错信息或者查看当前目录下的config.log去得到相关的信息。另外，如果自己不能解决那就去网上google一下吧，通常你会得到你想要的答案。



12

学习 shell 脚本之前的基础知识



日常的linux系统管理工作中必不可少的就是shell脚本，如果你不会写shell脚本，那么你不算一个合格的管理员。目前很多单位在招聘linux系统管理员时，shell脚本的编写是必考的项目。有的单位甚至用shell脚本的编写能力来衡量这个linux系统管理员的经验是否丰富。笔者讲这些的目的只有一个，那就是让你认真对待shell脚本，从一开始就要把基础知识掌握牢固，然后要不断的练习，只要你shell脚本写的好，相信你的linux求职路就会轻松的多。笔者在这一章中并不会多么详细的介绍shell脚本，而只是带你进入shell脚本的世界，如果你很感兴趣那么请到网上下载相关的资料或者到书店购买相关书籍吧。

在学习shell脚本之前，需要你了解很多关于shell的知识，这些知识是编写shell脚本的基础，所以希望你能够熟练的掌握。

【什么是shell】

简单点理解，就是系统跟计算机硬件交互时使用的中间介质，它只是系统的一个工具。实际上，在shell和计算机硬件之间还有一层东西那就是系统内核了。打个比方，如果把计算机硬件比作一个人的躯体，而系统内核则是人的大脑，至于shell，把它比作人的五官似乎更加贴切些。回到计算机上来，用户直接面对的不是计算机硬件而是shell，用户把指令告诉shell，然后shell再传输给系统内核，接着内核再去支配计算机硬件去执行各种操作。

笔者接触的linux发布版本（Redhat/CentOS）系统默认安装的shell叫做bash，即Bourne Again Shell，它是sh（Bourne Shell）的增强版本。Bourne Shell是最早行起来的一个shell，创始人叫Steven Bourne，为了纪念他所以叫做Bourne Shell，简称sh。那么这个bash有什么特点呢？

1) 记录命令历史

我们敲过的命令，linux是会有记录的，预设可以记录1000条历史命令。这些命令保存在用户的家目录中的.bash_history文件中。有一点需要你知道的是，只有当用户正常退出当前shell时，在当前shell中运行的命令才会保存至.bash_history文件中。

与命令历史有关的有一个有意思的字符那就是”!”了。常用的有这么几个应用：（1）!!（连续两个”!”），表示执行上一条指令；（2）!n（这里的n是数字），表示执行命令历史中第n条指令，例如”!100”表示执行命令历史中第100个命令；（3）!字符串（字符串大于等于1），例如!ta，表示执行命令历史中最近一次以ta为开头的指令。

```
[root@localhost ~]# ls
123 1.txt 456 anaconda-ks.cfg file1 install.log install.log.syslog
[root@localhost ~]# !!
ls
123 1.txt 456 anaconda-ks.cfg file1 install.log install.log.syslog
[root@localhost ~]# history |grep 996
 996 abc=123
1008 history |grep 996
[root@localhost ~]# !996
abc=123
[root@localhost ~]# !ab
abc=123
```

图片 12.1 12_1.png.jpg

2) 指令和文件名补全

在本教程最开始笔者就介绍过这个功能了，记得吗？对了就是按tab键，它可以帮你补全一个指令，也可以帮你补全一个路径或者一个文件名。连续按两次tab键，系统则会把所有的指令或者文件名都列出来。

3) 别名

前面也出现过alias的介绍，这个就是bash所特有的功能之一了。我们可以通过alias把一个常用的并且很长的指令别名一个简洁易记的指令。如果不想用了，还可以用unalias解除别名功能。直接敲alias会看到目前系统预设的alias：

```
[root@localhost ~]# alias
alias cp='cp -i'
alias l.='ls -d .* --color=tty'
alias ll='ls -l --color=tty'
alias ls='ls --color=tty'
alias mv='mv -i'
alias rm='rm -i'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-ti
lde'
```

图片 12.2 12_7.png.jpg

看到了吧，系统预设的alias指令也就这几个而已，你也可以自定义你想要的指令别名。alias语法很简单，alias [命令别名]=[‘具体的命令’]。

4) 通配符

在bash下，可以使用*来匹配零个或多个字符，而用?匹配一个字符。

```
[root@localhost ~]# ls -d test*
test test1.txt test3 test4 test.txt
[root@localhost ~]# ls -d test?
test3 test4
```

图片 12.3 12_8.png.jpg

5) 输入输出重定向

输入重定向用于改变命令的输入，输出重定向用于改变命令的输出。输出重定向更为常用，它经常用于将命令的结果输入到文件中，而不是屏幕上。输入重定向的命令是<，输出重定向的命令是>，另外还有错误重定向2>，以及追加重定向>>，稍后会详细介绍。

6) 管道符

前面已经提过管道符”|”，就是把前面的命令运行的结果丢给后面的命令。

7) 作业控制。

当运行一个进程时，你可以使它暂停（按Ctrl+z），然后使用fg命令恢复它，利用bg命令使他到后台运行，你也可以使它终止（按Ctrl+c）。

【变量】

前面章节中笔者曾经介绍过环境变量PATH，这个环境变量就是shell预设的一个变量，通常shell预设的变量都是大写的。变量，说简单点就是使用一个较简单的字符串来替代某些具有特殊意义的设定以及数据。就拿PATH来讲，这个PATH就代替了所有常用命令的绝对路径的设定。因为有了PATH这个变量，所以我们运行某个命令时不再去输入全局路径，直接敲命令名即可。你可以使用echo命令显示变量的值。

```
[root@localhost ~]# echo $PATH
/usr/lib/qt-3.3/bin:/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin
[root@localhost ~]# echo $PWD
/root
[root@localhost ~]# echo $HOME
/root
[root@localhost ~]# echo $LOGNAME
root
```

图片 12.4 12_9.png.jpg

除了PATH, HOME, LOGNAME外，系统预设的环境变量还有哪些呢？


```

[root@localhost ~]# env
HOSTNAME=localhost
TERM=xterm
SHELL=/bin/bash
HISTSIZE=1000
KDE_NO_IPV6=1
SSH_CLIENT=10.0.2.34 2222 22
QTDIR=/usr/lib/qt-3.3
QTINC=/usr/lib/qt-3.3/include
SSH_TTY=/dev/pts/0
USER=root
LS_COLORS=no=00:fi=00:di=00;34:ln=00;36:pi=40;33:so=00;35:bd=40;33;01
1:*.taz=00;31:*.lzh=00;31:*.zip=00;31:*.z=00;31:*.Z=00;31:*.gz=00;31:
KDEDIR=/usr
MAIL=/var/spool/mail/root
PATH=/usr/lib/qt-3.3/bin:/usr/kerberos/sbin:/usr/kerberos/bin:/usr/lo
INPUTRC=/etc/inputrc
PWD=/root
LANG=en_US.UTF-8
KDE_IS_PRELINKED=1
SHLVL=1
HOME=/root
LOGNAME=root
QTLIB=/usr/lib/qt-3.3/lib
CWS_RSH=ssh
SSH_CONNECTION=10.0.2.34 2222 10.0.2.60 22
LESSOPEN=|/usr/bin/lesspipe.sh %s
G_BROKEN_FILENAMES=1
_=/bin/env

```

图片 12.5 12_10.png.jpg

使用env命令即可全部列出系统预设的全部系统变量了。不过登录的用户不一样这些环境变量的值也不一样。当前显示的就是root这个账户的环境变量了。下面笔者简单介绍一下常见的环境变量：

PATH 决定了shell将到哪些目录中寻找命令或程序

HOME 当前用户主目录

HISTSIZE 历史记录数

LOGNAME 当前用户的登录名

HOSTNAME 指主机的名称

SHELL 前用户Shell类型

LANG 语言相关的环境变量，多语言可以修改此环境变量

MAIL 当前用户的邮件存放目录

PWD 当前目录

env命令显示的变量只是环境变量，系统预设的变量其实还有很多，你可以使用set命令把系统预设的全部变量都显示出来。

```
[root@localhost ~]# set
BASH=/bin/bash
BASH_ARGC=()
BASH_ARGV=()
BASH_LINENO=()
BASH_SOURCE=()
BASH_VERSINFO=( [0]="3" [1]="2" [2]="25" [3]="1" [4]="release" [5]="i686
-redhat-linux-gnu")
BASH_VERSION='3.2.25(1)-release'
COLORS=/etc/DIR_COLORS.xterm
COLUMNS=71
CUS_RSH=ssh
DIRSTACK=()
EUID=0
```

图片 12.6 12_11.png.jpg

限于篇幅，笔者在上例中并没有把所有显示结果都截图。set不仅可以显示系统预设的变量，也可以连同用户自定义的变量显示出来。用户自定义变量？是的，用户自己同样可以定义变量。

```
[root@localhost ~]# myname=Aming
[root@localhost ~]# echo $myname
Aming
[root@localhost ~]# set |grep myname
myname=Aming
```

图片 12.7 12_21.png.jpg

虽然你可以自定义变量，但是该变量只能在当前shell中生效，不信你再登录一个shell试试？

```
[root@localhost ~]# bash
[root@localhost ~]# echo $myname

[root@localhost ~]# exit
[root@localhost ~]# echo $myname
Aming
```

图片 12.8 12_22.png.jpg

使用bash命令即可再打开一个shell，此时先前设置的myname变量已经不存在了，退出当前shell回到原来的shell，myname变量还在。那要想设置的变量一直生效怎么办？有两种情况：

1) 要想系统内所有用户登录后都能使用该变量

需要在/etc/profile文件最末行加入“export myname=Aming”然后运行“source /etc/profile”就可以生效了。此时你再运行bash命令或者直接su - test账户看看。

```
[root@localhost ~]# source /etc/profile
[root@localhost ~]# bash
[root@localhost ~]# echo $myname
Aming
[root@localhost ~]# su - test
[test@localhost ~]$ echo $myname
Aming
```

图片 12.9 12_23.png.jpg

2) 只想让当前用户使用该变量

需要在用户主目录下的.bashrc文件最后一行加入“export myname=Aming”然后运行“source .bashrc”就可以生效了。这时候再登录test账户，myname变量则不会生效了。上面用的source命令的作用是，讲目前设定的配置刷新，即不用注销再登录也能生效。

笔者在上例中使用“myname=Aming”来设置变量myname，那么在linux下设置自定义变量有哪些规则呢？

- 设定变量的格式为“a=b”，其中a为变量名，b为变量的内容，等号两边不能有空格；
- 变量名只能由英、数字以及下划线组成，而且不能以数字开头；
- 当变量内容带有特殊字符（如空格）时，需要加上单引号；

```
[root@localhost test]# myname='Aming Li'
[root@localhost test]# echo $myname
Aming Li
```

图片 12.10 12_24.png.jpg

有一种情况，需要你注意，就是变量内容中本身带有单引号，这就需要用到双引号了。

```
[root@localhost ~]# myname="Aming's"
[root@localhost ~]# echo $myname
Aming's
```

图片 12.11 12_25.png.jpg

- 如果变量内容中需要用到其他命令运行结果则可以使用反引号；

```
[root@localhost test]# myname=`pwd`
[root@localhost test]# echo $myname
/root/test
```

图片 12.12 12_26.png.jpg

- 变量内容可以累加其他变量的内容，需要加双引号；

```
[root@localhost test]# myname="$LOGNAME"Aming
[root@localhost test]# echo $myname
rootAming
```

图片 12.13 12_27.png.jpg

在这里如果你不小心把双引号加错为单引号，将得不到你想要的结果

```
[root@localhost ~]# myname='$LOGNAME'Aming
[root@localhost ~]# echo $myname
$LOGNAMEAming
```

图片 12.14 12_28.png.jpg

通过上面几个例子也许你能看得出，单引号和双引号的区别：用双引号时不会取消掉里面出现的特殊字符的本身作用（这里的\$），而使用单引号则里面的特殊字符全部失去它本身的作用。

在前面的例子中笔者多次使用了bash命令，如果在当前shell中运行bash指令后，则会进入一个新的shell，这个shell就是原来shell的子shell了，不妨你用pstree指令来查看一下。

```
[root@localhost ~]# pstree |grep bash
|-sshd---sshd---bash+-grep
[root@localhost ~]# bash
[root@localhost ~]# pstree |grep bash
|-sshd---sshd---bash---bash+-grep
```

图片 12.15 12_29.png.jpg

pstree这个指令会把linux系统中所有进程通过树形结构打印出来。限于篇幅笔者没有全部列出，你可以直接输入pstree查看即可。在父shell中设定一个变量后，进入子shell后该变量是不会生效的，如果想让这个变量在子shell中生效则要用到export指令，笔者曾经在前面用过。

```
[root@localhost ~]# abc=123
[root@localhost ~]# echo $abc
123
[root@localhost ~]# bash
[root@localhost ~]# echo $abc

[root@localhost ~]# exit
[root@localhost ~]# export abc
[root@localhost ~]# bash
[root@localhost ~]# echo $abc
123
```

图片 12.16 12_44.png.jpg

export其实就是声明一下这个变量的意思，让该shell的子shell也知道变量abc的值是123.如果export后面不加任何变量名，则它会声明所有的变量。

```
[root@localhost ~]# export
declare -x CVS_RSH="ssh"
declare -x G_BROKEN_FILENAMES="1"
declare -x HISTSIZE="1000"
declare -x HOME="/root"
declare -x HOSTNAME="localhost"
declare -x INPUTRC="/etc/inputrc"
declare -x KDEDIR="/usr"
declare -x KDE_IS_PRELINKED="1"
declare -x KDE_NO_IPV6="1"
declare -x LANG="en_US.UTF-8"
declare -x LESSOPEN="|/usr/bin/lesspipe.sh %s"
declare -x LOGNAME="root"
declare -x LS_COLORS="no=00:fi=00:di=00;34:ln=00;36:pi=40;33:so=00;35:bd=40;33;0
1:cd=40;33;01:or=01;05;37;41:mi=01;05;37;41:ex=00;32:*.cmd=00;32:*.exe=00;32:*.c
om=00;32:*.bat=00;32:*.sh=00;32:*.csh=00;32:*.tar=00;31:*.tgz=00;31:
*.arj=00;31:*.taz=00;31:*.lzh=00;31:*.zip=00;31:*.z=00;31:*.Z=00;31:*.gz=00;31:*
.bz2=00;31:*.bz=00;31:*.tz=00;31:*.rpm=00;31:*.cpio=00;31:*.jpg=00;35:*.gif=00;3
5:*.bmp=00;35:*.xbm=00;35:*.xpm=00;35:*.png=00;35:*.tif=00;35:"
declare -x MAIL="/var/spool/mail/root"
declare -x OLDPWD
declare -x PATH="/usr/lib/qt-3.3/bin:/usr/kerberos/sbin:/usr/kerberos/bin:/usr/l
ocal/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin"
declare -x PWD="/root"
declare -x QTDIR="/usr/lib/qt-3.3"
declare -x QTINC="/usr/lib/qt-3.3/include"
declare -x QTLIB="/usr/lib/qt-3.3/lib"
declare -x SHELL="/bin/bash"
declare -x SHLVL="2"
declare -x SSH_CLIENT="10.0.2.34 3337 22"
declare -x SSH_CONNECTION="10.0.2.34 3337 10.0.2.60 22"
declare -x SSH_TTY="/dev/pts/0"
declare -x TERM="xterm"
declare -x USER="root"
declare -x abc="123"
declare -x myname="Aming"
```

图片 12.17 12_45.png.jpg

在最后面连同我们自定义的变量都被声明了。

前面光讲如何设置变量，如果想取消某个变量怎么办？只要输入“unset 变量名”即可。

```
[root@localhost ~]# echo $abc
123
[root@localhost ~]# unset abc
[root@localhost ~]# echo $abc
```

图片 12.18 12_46.png.jpg

用unset abc后，再echo \$abc则不再输出任何内容。

【系统环境变量与个人环境变量的配置文件】

上面讲了很多系统的变量，那么在linux系统中，这些变量被存到了哪里呢，为什么用户一登陆shell就自动有了这些变量呢？

`/etc/profile`：这个文件预设了几个重要的变量，例如PATH, USER, LOGNAME, MAIL, INPUTRC, HOSTNAME, HISTSIZE, umask等等。

`/etc/bashrc`：这个文件主要预设umask以及PS1。这个PS1就是我们在敲命令时，前面那串字符了，例如笔者的linux系统PS1就是 `[root@localhost ~]#`，你不妨看一下PS1的值。

```
[root@localhost ~]# echo $PS1
[\u@\h \W]\$
```

图片 12.19 12_47.png.jpg

`\u`就是用户，`\h` 主机名，`\W` 则是当前目录，`\$`就是那个‘#’了，如果是普通用户则显示为‘\$’

除了两个系统级别的配置文件外，每个用户的主目录下还有几个这样的隐藏文件：

`.bash_profile`：定义了用户的个人化路径与环境变量的文件名称。每个用户都可使用该文件输入专用于自己使用的shell信息,当用户登录时,该文件仅仅执行一次。

`.bashrc`：该文件包含专用于你的shell的bash信息,当登录时以及每次打开新的shell时,该该文件被读取。例如你可以将用户自定义的alias或者自定义变量写到这个文件中。

`.bash_history`：记录命令历史用的。

`.bash_logout`：当退出shell时，会执行该文件。可以把一些清理的工作放到这个文件中。

【linux shell中的特殊符号】

你在学习linux的过程中，也许你已经接触过某个特殊符号，例如“*”，它是一个通配符号，代表零个或多个字符或数字。下面笔者就说一说常用到的特殊字符。

`**1. ***`：代表零个或多个字符或数字。

```
[root@localhost ~]# ls -d test*
test test1.txt test3 test4 test.txt
```

图片 12.20 12_48.png.jpg

test后面可以没有任何字符，也可以有多个字符，总之有或没有都能匹配出来。

2. `?`：只代表一个任意的字符

```
[root@localhost ~]# touch testa testb testaa
[root@localhost ~]# ls -d test?
test3 test4 testa testb
```

图片 12.21 12_49.png.jpg

不管是数字还是字母，只要是一个都能匹配出来。

3. #：这个符号在linux中表示注释说明的意思，即”#”后面的内容linux忽略掉。

```
[root@localhost ~]# #alsdjflaksjdf1kasjdf
[root@localhost ~]# echo $?
0
[root@localhost ~]# ls test3 # list
test2 test4
```

图片 12.22 12_50.png.jpg

在命令的开头或者中间插入”#”，linux都会忽略掉的。这个符号在shell脚本中用的很多。

**4. *：脱意字符，将后面的特殊符号（例如”*”）还原为普通字符。

```
[root@localhost ~]# ls test\*
ls: test*: No such file or directory
```

图片 12.23 12_51.png.jpg

5. |：管道符，前面多次说过，它的作用在于将符号前面命令的结果丢给符号后面的命令。这里提到的后面的命令，并不是所有的命令都可以的，一般针对文档操作的命令比较常用，例如cat, less, head, tail, grep, cut, sort, wc, uniq, tee, tr, split, sed, awk等等，其中grep, sed, awk为正则表达式必须掌握的工具，在后续内容中详细介绍。

6. \$：除了用于变量前面的标识符外，还有一个妙用，就是和’!’结合起来使用。

```
[root@localhost ~]# ls test.txt
test.txt
[root@localhost ~]# ls !$
ls test.txt
test.txt
```

图片 12.24 12_52.png.jpg

’!\$’表示上条命令中最后一个变量（也许称为变量不合适，总之就是上条命令中最后出现的那个东西）例如上边命令最后是test.txt那么在当前命令下输入!\$则代表test.txt。

1) grep：过滤一个或多个字符，将会在后续内容中详细介绍其用法。

```
[root@localhost ~]# cat /etc/passwd |grep root
root:x:0:0:root,root,000,001:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
```

图片 12.25 12_53.png.jpg

2) cut: 截取某一个字段

语法: cut -d “分隔字符” [-cf] n 这里的n是数字

-d: 后面跟分隔字符, 分隔字符要用双引号括起来

-c: 后面接的是第几个字符

-f: 后面接的是第几个区块

```
[root@localhost ~]# cat /etc/passwd |cut -d ":" -f 1
root
bin
daemon
adm
lp
sync
shutdown
halt
```

图片 12.26 12_54.png.jpg

-d 后面跟分隔字符, 这里使用冒号作为分割字符, -f 1 就是截取第一段, -f和1之间的空格可有可无。

```
[root@localhost ~]# head -n2 /etc/passwd |cut -c2
o
i
[root@localhost ~]# head -n2 /etc/passwd |cut -c1
r
b
[root@localhost ~]# head -n2 /etc/passwd |cut -c1-10
root:x:0:0
bin:x:1:1:
[root@localhost ~]# head -n2 /etc/passwd |cut -c5-10
:x:0:0
x:1:1:
```

图片 12.27 12_55.png.jpg

-c 后面可以是1个数字n, 也可以是一个区间n1-n2, 还可以是多个数字n1,n2,n3

```
[root@localhost ~]# head -n2 /etc/passwd |cut -c1,2,5
ro:
bix
```

图片 12.28 12_56.png.jpg

3) sort: 用做排序

语法: sort [-t 分隔符] [-kn1,n2] [-nru] 这里的n1 < n2

-t 分隔符: 作用跟cut的-d一个意思

-n : 使用纯数字排序

-r : 反向排序

-u : 去重复

-kn1,n2 : 由n1区间排序到n2区间, 可以只写-kn1, 即对n1字段排序

```
[root@localhost ~]# head -n5 /etc/passwd | sort
adm:x:3:4:adm:/var/adm:/sbin/nologin
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
root:x:0:0:root,root,000,001:/root:/bin/bash
```

图片 12.29 12_57.png.jpg

```
[root@localhost ~]# head -n5 /etc/passwd | sort -t: -k3n
root:x:0:0:root,root,000,001:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
[root@localhost ~]# head -n5 /etc/passwd | sort -t: -k3,5n
root:x:0:0:root,root,000,001:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

图片 12.30 12_78.png.jpg

```
[root@localhost ~]# head -n5 /etc/passwd | sort -t: -k3nr
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
bin:x:1:1:bin:/bin:/sbin/nologin
root:x:0:0:root,root,000,001:/root:/bin/bash
```

图片 12.31 12_79.png.jpg

4) wc: 统计文档的行数、字符数、词数, 常用的选项为:

-l : 统计行数

-m : 统计字符数

-w : 统计词数

```
[root@localhost ~]# echo "hello world" >123.txt
[root@localhost ~]# echo "hello1 world2" >>123.txt
[root@localhost ~]# cat 123.txt
hello world
hello1 world2
[root@localhost ~]# cat 123.txt |wc -l
2
[root@localhost ~]# cat 123.txt |wc -m
26
[root@localhost ~]# cat 123.txt |wc -w
4
```

图片 12.32 12_80.png.jpg

5) uniq: 去重复的行, 笔者常用的选项只有一个:

-c: 统计重复的行数, 并把行数写在前面

```
[root@localhost ~]# echo "hello1 world2" >>123.txt
[root@localhost ~]# cat 123.txt
hello world
hello1 world2
hello1 world2
[root@localhost ~]# cat 123.txt |uniq
hello world
hello1 world2
[root@localhost ~]# cat 123.txt |uniq -c
 1 hello world
 2 hello1 world2
```

图片 12.33 12_81.png.jpg

有一点需要注意, 在进行uniq之前, 需要先用sort排序然后才能uniq, 否则你将得不到你想要的, 笔者上面的试验当中已经是排序过所以省略掉那步了。

6) tee: 后跟文件名, 类似与重定向">", 但是比重定向多了一个功能, 在把文件写入后面所跟的文件中的同时, 还显示在屏幕上。

```
[root@localhost ~]# echo "asldkfjalskdjf" |tee 1.txt
asldkfjalskdjf
[root@localhost ~]# cat 1.txt
asldkfjalskdjf
```

图片 12.34 12_82.png.jpg

7) tr: 替换字符, 常用来处理文档中出现的特殊符号, 如DOS文档中出现的^M符号。常用的选项有两个:

-d: 删除某个字符, -d 后面跟要删除的字符

-s: 把重复的字符去掉

最常用的就是把小写变大写: tr '[a-z]' '[A-Z]'

```
[root@localhost ~]# head -n1 /etc/passwd |tr '[a-z]' '[A-Z]'
ROOT:X:0:0:ROOT,ROOT,000,001:/ROOT:/BIN/BASH
```

图片 12.35 12_83.png.jpg

当然替换一个字符也是完全可以的。

```
[root@localhost ~]# cp /etc/passwd 1.txt
[root@localhost ~]# cat 1.txt |grep root
root:x:0:0:root,root,000,001:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
[root@localhost ~]# cat 1.txt |grep root | tr 'r' 'R'
Root:x:0:0:Root,Root,000,001:/Root:/bin/bash
opeRatoR:x:11:0:opeRatoR:/Root:/sbin/nologin
```

图片 12.36 12_84.png.jpg

不过替换、删除以及去重复都是针对一个字符来讲的，有一定局限性。如果是针对一个字符串就不再管用了，所以笔者建议只是简单了解这个tr即可，以后你还会学到更多可以实现针对字符串操作的工具。

```
[root@localhost ~]# cat 1.txt |grep root
root:x:0:0:root,root,000,001:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
[root@localhost ~]# cat 1.txt |grep root |tr -d 'o'
rt:x:0:0:rt,rt,000,001:/rt:/bin/bash
peratr:x:11:0:peratr:/rt:/sbin/nlgin
[root@localhost ~]# cat 1.txt |grep root |tr -s 'o'
rot:x:0:0:rot,rot,000,001:/rot:/bin/bash
operator:x:11:0:operator:/rot:/sbin/nologin
```

图片 12.37 12_85.png.jpg

8) split：切割文档，常用选项：

-b：依据大小来分割文档，单位为byte

```
[root@localhost ~]# split -b 500 /etc/passwd passwd
[root@localhost ~]# ls passwd*
passwdaa passwdab passwdac
```

图片 12.38 12_86.png.jpg

格式如上例，后面的passwd为分割后文件名的前缀，分割后的文件名为passwdaa, passwdab, passwdac …

-l：依据行数来分割文档

```
[root@localhost ~]# wc -l /etc/passwd
26 /etc/passwd
[root@localhost ~]# rm -rf passwd*
[root@localhost ~]# split -l 10 /etc/passwd passwd
[root@localhost ~]# wc -l passwd*
 10 passwdaa
 10 passwdab
  6 passwdac
```

图片 12.39 12_87.png.jpg

6. ; : 分号。平时我们都是在一行中敲一个命令，然后回车就运行了，那么想在一行中运行两个或两个以上的命令如何呢？则需要在命令之间加一个” ;”了。

```
[root@localhost ~]# ls -d test* ; touch test5; ls -d test*
test test1.txt test3 test4 testa testaa testb test.txt
test test1.txt test3 test4 test5 testa testaa testb test.txt
```

图片 12.40 12_88.png.jpg

7. ~: 用户的家目录，如果是root则是 /root，普通用户则是 /home/username

```
[root@localhost ~]# cd ~
[root@localhost ~]# pwd
/root
[root@localhost ~]# su test
[test@localhost root]$ cd ~
[test@localhost ~]$ pwd
/home/test
```

图片 12.41 12_89.png.jpg

8. & : 如果想把一条命令放到后台执行的话，则需要加上这个符号。通常用于命令运行时间非常长的情况。

```
[root@localhost ~]# sleep 100 &
[1] 2133
[root@localhost ~]#
[root@localhost ~]# jobs
[1]+  Running                  sleep 100 &
```

图片 12.42 12_90.png.jpg

使用jobs可以查看当前shell中后台执行的任务。用fg可以调到前台执行。这里的sleep命令就是休眠的意思，后面跟数字，单位为秒，常用语循环的shell脚本中。

```
[root@localhost ~]# fg
sleep 100
```

图片 12.43 12_91.png.jpg

此时你按一下CTRL +z 使之暂停，然后再输入bg可以再次进入后台执行。

```
[1]+  Stopped                  sleep 100
[root@localhost ~]# bg
[1]+ sleep 100 &
```

图片 12.44 12_92.png.jpg

如果是多任务情况下，想要把任务调到前台执行的话，fg后面跟任务号，任务号可以使用jobs命令得到。

```
[root@localhost ~]# sleep 20 &
[1] 2136
[root@localhost ~]# sleep 30 &
[2] 2137
[root@localhost ~]# jobs
[1]-  Running                  sleep 20 &
[2]+  Running                  sleep 30 &
[root@localhost ~]# fg 1
sleep 20
```

图片 12.45 12_93.png.jpg

9. >, >>, 2>, 2>>: 前面讲过重定向符号> 以及>> 分别表示取代和追加的意思, 然后还有两个符号就是这里的 2> 和 2>> 分别表示错误重定向和错误追加重定向, 当我们运行一个命令报错时, 报错信息会输出到当前的屏幕, 如果想重定向到一个文本里, 则要用2>或者2>>。

```
[root@localhost ~]# ls aaaa
ls: aaaa: No such file or directory
[root@localhost ~]# ls aaaa >error
ls: aaaa: No such file or directory
[root@localhost ~]# cat error
[root@localhost ~]# ls aaaa 2>error
[root@localhost ~]# cat error
ls: aaaa: No such file or directory
```

图片 12.46 12_94.png.jpg

10. []: 中括号, 中间为字符组合, 代表中间字符中的任意一个

```
[root@localhost ~]# ls -d test*
test test1.txt test3 test4 test5 testa testaa testb test.txt
[root@localhost ~]# ls test[12a]
testa
[root@localhost ~]# ls test[a-c]
testa testb
[root@localhost ~]# ls test[0-9]
test4 test5

test3:
test2 test4
```

图片 12.47 12_95.png.jpg

11. && 与 ||

在上面刚刚提到了分号, 用于多条命令间的分隔符。另外还有两个可以用于多条命令中间的特殊符号, 那就是“&&”和“||”。下面笔者把这几种情况全列出:

- 1) command1 ; command2
- 2) command1 && command2
- 3) command1 || command2

使用” ;” 时，不管command1是否执行成功都会执行command2；使用” &&” 时，只有command1执行成功后，command2才会执行，否则command2不执行；使用” ||” 时，command1执行成功后command2 不执行，否则去执行command2，总之command1和command2总有一条命令会执行。

```
[root@localhost ~]# rm -rf test*
[root@localhost ~]# touch test1 test3
[root@localhost ~]# ls test2 && touch test2
ls: test2: No such file or directory
[root@localhost ~]# ls test2
ls: test2: No such file or directory
[root@localhost ~]# ls test2 || touch test2
ls: test2: No such file or directory
[root@localhost ~]# ls test2
test2
```

图片 12.48 12_96.png.jpg



T



13

正则表达式



这部分内容可以说是学习shell脚本之前必学的内容。如果你这部分内容学的越好，那么你的shell脚本编写能力就会越强。所以不要嫌这部分内容啰嗦，也不要怕麻烦，要用心学习。一定要多加练习，练习多了就能熟练掌握了。

在计算机科学中，正则表达式是这样解释的：它是指一个用来描述或者匹配一系列符合某个句法规则的字符串的单个字符串。在很多文本编辑器或其他工具里，正则表达式通常被用来检索和/或替换那些符合某个模式的文本内容。许多程序设计语言都支持利用正则表达式进行字符串操作。对于系统管理员来讲，正则表达式贯穿在我们的日常运维工作中，无论是查找某个文档，抑或查询某个日志文件分析其内容，都会用到正则表达式。

其实正则表达式，只是一种思想，一种表示方法。只要我们使用的工具支持表示这种思想那么这个工具就可以处理正则表达式的字符串。常用的工具有grep, sed, awk 等，下面笔者就介绍一下这三种工具的使用方法。

【 grep / egrep 】

笔者在前面的内容中多次提到并用到grep命令，可见它的重要性。所以好好学习一下这个重要的命令吧。你要知道的是grep连同下面讲的sed, awk都是针对文本的行才操作的。

语法：grep [-cinvABC] 'word' filename

-c：打印符合要求的行数

-i：忽略大小写

-n：在输出符合要求的行的同时连同行号一起输出

-v：打印不符合要求的行

-A：后跟一个数字（有无空格都可以），例如 -A2则表示打印符合要求的行以及下面两行

-B：后跟一个数字，例如 -B2 则表示打印符合要求的行以及上面两行

-C：后跟一个数字，例如 -C2 则表示打印符合要求的行以及上下各两行


```
[root@localhost ~]# grep -A 2 halt /etc/passwd
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
[root@localhost ~]# grep -B 2 halt /etc/passwd
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
[root@localhost ~]# grep -C 2 halt /etc/passwd
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
[root@localhost ~]# grep -C2 halt /etc/passwd
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
```

图片 13.1 13_1.png.jpg

以下，笔者举几个小例子帮助你好好掌握这个grep工具的用法。

a. 过滤出带有某个关键词的行并输出行号

```
[root@localhost ~]# grep -n 'root' /etc/passwd
1:root:x:0:0:root,root,000,001:/root:/bin/bash
12:operator:x:11:0:operator:/root:/sbin/nologin
```

图片 13.2 13_7.png.jpg

b. 过滤不带有某个关键词的行，并输出行号

```
[root@localhost ~]# grep -vn 'nologin' /etc/passwd
1:root:x:0:0:root,root,000,001:/root:/bin/bash
6:sync:x:5:0:sync:/sbin:/bin/sync
7:shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
8:halt:x:7:0:halt:/sbin:/sbin/halt
10:news:x:9:13:news:/etc/news:
25:test:x:500:500:test,test's Office,12345,67890:/home/test:/bin/bash
26:test1:x:501:500:~/home/test1:/bin/bash
```

图片 13.3 13_8.png.jpg

c. 过滤出所有包含数字的行

```
[root@localhost ~]# cat test.txt
aaaaaaaaaaaa
121aaaaaaaaaaaaaa
121212121212121212
bbbbbbbbbbbbbbbbbbbb
3333dddd4444444444
eeeeeeeeeeee11111111
[root@localhost ~]# grep [0-9] test.txt
121aaaaaaaaaaaaaa
121212121212121212
3333dddd4444444444
eeeeeeeeeeee11111111
[root@localhost ~]# grep -v [0-9] test.txt
aaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbb
```

图片 13.4 13_9.png.jpg

在前面也提到过这个“[]”的应用，如果是数字的话就用[0-9]这样的形式，当然有时候也可以用这样的形式[15]即只含有1或者5，注意，它不会认为是15。如果要过滤出数字以及大小写字母则要这样写[0-9a-zA-Z]。另外[]还有一种形式，就是[^字符]表示除[]内的字符之外的字符。

```
[root@localhost ~]# grep '[^r]oo' /etc/passwd
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
exim:x:93:93::/var/spool/exim:/sbin/nologin
```

图片 13.5 13_10.png.jpg

这就表示筛选包含oo字符串，但是不包含r字符。

d. 过滤出文档中以某个字符开头或者以某个字符结尾的行

```
[root@localhost ~]# grep '^r' /etc/passwd
root:x:0:0:root,root,000,001:/root:/bin/bash
[root@localhost ~]# grep 'h$' /etc/passwd
root:x:0:0:root,root,000,001:/root:/bin/bash
test:x:500:500:test,test's Office,12345,67890:/home/test:/bin/bash
test1:x:501:500:./home/test1:/bin/bash
[root@localhost ~]# grep 'sh$' /etc/passwd
root:x:0:0:root,root,000,001:/root:/bin/bash
test:x:500:500:test,test's Office,12345,67890:/home/test:/bin/bash
test1:x:501:500:./home/test1:/bin/bash
[root@localhost ~]# grep '^test' /etc/passwd
test:x:500:500:test,test's Office,12345,67890:/home/test:/bin/bash
test1:x:501:500:./home/test1:/bin/bash
```

图片 13.6 13_11.png.jpg

在正则表达式中，“^”表示行的开始，“\$”表示行的结尾，那么空行则表示“^\$”，如果你只想筛选出非空行，则可以使用“grep -v ‘^\$’ filename”得到你想要的结果。现在想一下，如何打印出不以英文字母开头的行呢？

```
[root@localhost ~]# echo "123" >test.txt
[root@localhost ~]# echo "abc" >>test.txt
[root@localhost ~]# echo "456" >>test.txt
[root@localhost ~]# echo "abc456" >>test.txt
[root@localhost ~]# grep '^^[a-zA-Z]' test.txt
123
456
[root@localhost ~]# echo ".*&abc456" >>test.txt
[root@localhost ~]# grep '^^[a-zA-Z]' test.txt
123
456
.*&abc456
```

图片 13.7 13_21.png.jpg

e. 过滤任意一个字符与重复字符

```
[root@localhost ~]# grep 'r..o' /etc/passwd
operator:x:11:0:operator:/root:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
```

图片 13.8 13_22.png.jpg

“.”表示任意一个字符，上例中，就是把符合r与o之间有两个任意字符的行过滤出来。

“*”表示零个或多个前面的字符。

```
[root@localhost ~]# grep 'ooo*' /etc/passwd
root:x:0:0:root,root,000,001:/root:/bin/bash
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
exim:x:93:93::/var/spool/exim:/sbin/nologin
```

图片 13.9 13_23.png.jpg

‘ooo’表示oo,ooo,oooo…或者更多的‘o’。现在你是否想到了‘.’这个组合表示什么意义？

```
[root@localhost ~]# grep '.*' /etc/passwd |wc -l
26
[root@localhost ~]# wc -l /etc/passwd
26 /etc/passwd
```

图片 13.10 13_24.png.jpg

‘.*’表示零个或多个任意字符，空行也包含在内。

f. 指定要过滤字符出现的次数

图片 13.26 13_54.png.jpg

现在思考一下，如何删除文档中的所有数字或者字母？

```
[root@localhost ~]# sed 's/[0-9]//g' test.txt
rot:x::/rot:/bin/bash
operator:x::operator:/root:/sbin/nologin
operator:x::operator:/rooot:/sbin/nologin
rooooot:x::/rooooot:/bin/bash

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

图片 13.27 13_55.png.jpg

有意思吧，[0-9]表示任意的数字。这里你也可以写成[a-zA-Z]甚至[0-9a-zA-Z]

```
[root@localhost ~]# sed 's/[a-zA-Z]//g' test.txt
::0:0:/://
::11:0:/://
::11:0:/://
::0:0:/://
11111111111111111111111111111111

[root@localhost ~]# sed 's/[0-9a-zA-Z]//g' test.txt
:::/://
::::/://
::::/://
::::/://
```

图片 13.28 13_56.png.jpg

g. 调换两个字符串的位置

```
[root@localhost ~]# sed 's/\(rot\) \(.*\) \(bash\) /\3\2\1/' test.txt
bash:x:0:0:/rot:/bin/rot
operator:x:11:0:operator:/root:/sbin/nologin
operator:x:11:0:operator:/rooot:/sbin/nologin
rooooot:x:0:0:/rooooot:/bin/bash
11111111111111111111111111111111
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

图片 13.29 13_57.png.jpg

这个就需要解释一下了，上例中用'()'把所想要替换的字符括起来成为一个整体，因为括号在sed中属于特殊符号，所以需要在前面加脱意字符'\', 替换时则写成'\1', '\2', '\3'的形式。除了调换两个字符串的位置外，笔者还常常用到在某一行前或者后增加指定内容。

13. 在test.txt 20行到末行最前面加'aaa:' ;

现在给出以上练习题的答案, 你如果实在想不出如何操作, 那你看看答案吧, 请尽量多想一下。

1. /bin/cp /etc/passwd /root/test.txt ; sed -n '1,/span>p test.txt

2. sed -n '3,10p test.txt

3. sed -n '/root/'p test.txt

4. sed '15,/span>d test.txt

5. sed '/bash/'d test.txt

6. sed 's/root/toor/g' test.txt

7. sed 's/sbin/nologin#bin/login#g' test.txt

8. sed '5,10s/[0-9]//g' test.txt

9. sed 's/[^0-9a-zA-Z]//g' test.txt

10. sed 's/([a-zA-Z][a-zA-Z])([a-zA-Z])([a-zA-Z])([a-zA-Z][a-zA-Z]*\$)\4\2\3\1/ test.txt

11. sed 's#([0-9][0-9])([0-9][0-9])([0-9])([a-zA-Z])([a-zA-Z][a-zA-Z]\$)#\1\5\3\4\2#' test.txt

12. sed 's#([0-9][0-9])([0-9][0-9])([0-9].*\$)#\1\3\2#' test.txt

13. sed '20,\$s/^.*/aaa:&/' test.txt

【awk工具的使用】

上面也提到了awk和sed一样是流式编辑器, 它也是针对文档中的行来操作的, 一行一行的去执行。awk比sed更加强, 它能做到sed能做到的, 同样也能做到sed不能做到的。awk工具其实是很复杂的, 有专门的书籍来介绍它的应用, 但是笔者认为学那么复杂没有必要, 只要能处理日常管理工作中问题即可。何必让自己的脑袋装那么多东西来为难自己? 毕竟用的也不多, 即使现在教会了你很多, 你也学会了, 如果很久不用肯定就忘记了。鉴于此, 笔者仅介绍比较常见的awk应用, 如果你感兴趣的话, 再去深入研究吧。

a. 截取文档中的某个段

```
[root@localhost ~]# head -n2 test.txt |awk -F':' '{print $1}'
root
bin
```

图片 13.31 13_79.png.jpg

解释一下, -F 选项的作用是指定分隔符, 如果不加-F指定, 则以空格或者tab为分隔符。

```
[root@localhost ~]# head -n2 /etc/inittab
#
# inittab          This file describes how the INIT process should set up
[root@localhost ~]# head -n2 /etc/inittab |awk '{print $2,$3}'

inittab This
```

图片 13.32 13_80.png.jpg

Print为打印的动作，用来打印出某个字段。\$1为第一个字段，\$2为第二个字段，依次类推，有一个特殊的就是\$0，它表示整行。

```
[root@localhost ~]# head -n2 test.txt |awk -F':' '{print $0}'
root:x:0:0:root,root,000,001:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
```

图片 13.33 13_81.png.jpg

注意awk的格式，-F后紧跟单引号，然后里面为分隔符，print的动作要用' {}'括起来，否则会报错。print还可以打印自定义的内容，但是自定义的内容要用双引号括起来。

```
[root@localhost ~]# head -n2 test.txt |awk -F':' '{print $1"@"$2"@"$3}'
root@x@0
bin@x@1
```

图片 13.34 13_82.png.jpg

b. 匹配字符或字符串

```
[root@localhost ~]# awk '/root/' test.txt
root:x:0:0:root,root,000,001:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
```

图片 13.35 13_83.png.jpg

跟sed很类似吧，不过还有比sed更强大的匹配。

```
[root@localhost ~]# awk -F':' '$1~/root/' test.txt
root:x:0:0:root,root,000,001:/root:/bin/bash
```

图片 13.36 13_84.png.jpg

可以让某个段去匹配，这里的' ~'就是匹配的意思，继续往下看

```
[root@localhost ~]# awk -F':' '/root/ {print $3} /test/ {print $3}' test.txt
0
11
500
501
```

图片 13.37 13_85.png.jpg

awk还可以多次匹配，如上例中匹配完root，再匹配test，它还可以只打印所匹配的段。

```
[root@localhost ~]# awk -F':' '$1~/root/ {print $1}' test.txt
root
```

图片 13.38 13_86.png.jpg

不过这样没有啥意义，笔者只是为了说明awk确实比sed强大。

d. 条件操作符

```
[root@localhost ~]# awk -F':' '$3=="0"' test.txt
root:x:0:0:root,root,000,001:/root:/bin/bash
```

图片 13.39 13_87.png.jpg

awk中是可以逻辑符号判断的，比如‘==’就是等于，也可以理解为“精确匹配”。另外也有‘>’，‘>=’，‘<’，‘<=’，‘!=’等等，值得注意的是，即使\$3为数字，awk也不会把它当数字看待，它会认为是一个字符。所以不要妄图去拿\$3当数字去和数字做比较。

```
[root@localhost ~]# cat test.txt |awk -F':' '$3>="500"'
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
nobody:x:99:99:Nobody:/:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
exim:x:93:93:/:/var/spool/exim:/sbin/nologin
avahi:x:70:70:Avahi daemon:/:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
test:x:500:500:test,test's Office,12345,67890:/home/test:/bin/bash
test1:x:501:500:/:/home/test1:/bin/bash
```

图片 13.40 13_88.png.jpg

这样是得不到我们想要的效果的。这里只是字符与字符之间的比较，‘6’是>‘500’的。

```
[root@localhost ~]# cat test.txt |awk -F':' '$7!="sbin/nologin"'
root:x:0:0:root,root,000,001:/root:/bin/bash
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
news:x:9:13:news:/etc/news:
test:x:500:500:test,test's Office,12345,67890:/home/test:/bin/bash
test1:x:501:500:/:/home/test1:/bin/bash
```

图片 13.41 13_89.png.jpg

上例中用的是‘!=’即不匹配。

```
[root@localhost ~]# awk -F':' '$3<$4' test.txt
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
```

图片 13.42 13_90.png.jpg

另外还可以使用“&&”和“||”表示“并且”和“或者”的意思。

```
[root@localhost ~]# awk -F':' '$3>"5" && $3<"7"' test.txt
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
test:x:500:500:test,test's Office,12345,67890:/home/test:/bin/bash
test1:x:501:500:~/home/test1:/bin/bash
```

图片 13.43 13_91.png.jpg

也可以是或者的关系

```
[root@localhost ~]# awk -F':' '$3<"1" || $3>"8"' test.txt
root:x:0:0:root,root,000,001:/root:/bin/bash
news:x:9:13:news:/etc/news:
nobody:x:99:99:Nobody:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
exim:x:93:93:~/var/spool/exim:/sbin/nologin
```

图片 13.44 13_92.png.jpg

d. awk的内置变量

常用的变量有：

NF：用分隔符分隔后一共有多少段；

NR：行数

```
[root@localhost ~]# head -n5 test.txt |awk -F':' '{print NF}'
7
7
7
7
7
[root@localhost ~]# head -n5 test.txt |awk -F':' '{print $NF}'
/bin/bash
/sbin/nologin
/sbin/nologin
/sbin/nologin
/sbin/nologin
```

图片 13.45 13_93.png.jpg

上例中，打印总共的段数以及最后一段的值。

```
[root@localhost ~]# awk 'NR>=20' test.txt
exim:x:93:93::/var/spool/exim:/sbin/nologin
avahi:x:70:70:Avahi daemon:/:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
xfs:x:43:43:X Font Server:/etc/X11/fs:/sbin/nologin
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
test:x:500:500:test,test's Office,12345,67890:/home/test:/bin/bash
test1:x:501:500:~/home/test1:/bin/bash
apache:x:48:48:Apache:/var/www:/sbin/nologin
```

图片 13.46 13_94.png.jpg

可以使用NR作为条件，来打印出指定的行。

```
[root@localhost ~]# awk -F':' 'NR>=20 && $1~/ssh/' test.txt
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
```

图片 13.47 13_95.png.jpg

e. awk中的数学运算

```
[root@localhost ~]# head -n 5 test.txt |awk -F':' '$1=="root"'
root x 0 0 root,root,000,001 /root /bin/bash
root x 1 1 bin /bin /sbin/nologin
root x 2 2 daemon /sbin /sbin/nologin
root x 3 4 adm /var/adm /sbin/nologin
root x 4 7 lp /var/spool/lpd /sbin/nologin
```

图片 13.48 13_96.png.jpg

awk比较强的地方，还在于能把某个段改成指定的字符串，下面还有更强的呢！

```
[root@localhost ~]# head -n 2 test.txt |awk -F':' '{ $7=$3+$4; print $3,$4,$7}'
0 0 0
1 1 2
```

图片 13.49 13_97.png.jpg

当然还可以计算某个段的总和。

```
[root@localhost ~]# cat test.txt |awk -F':' '{(tot+=$3)};END {print tot}'
1767
```

图片 13.50 13_101.png.jpg

这里的END要注意一下，表示所有的行都已经执行，这是awk特有的语法，其实awk连同sed都可以写成一个脚本文件，而且他们有特有的语法，在awk中使用if判断、for循环都是可以的，只是笔者认为日常管理工作中没有必要使用那么复杂的语句而已。

```
[root@localhost ~]# awk -F':' '{if ($1=="root") print $0}' test.txt
root:x:0:0:root,root,000,001:/root:/bin/bash
```

图片 13.51 13_102.png.jpg

注意这里 '()' 的使用。

基本上，正则表达的内容就这些了。但是笔者要提醒你一下，笔者介绍的这些仅仅是最基本的东西，并没有提啊深入的去讲sed和awk，但是完全可以满足日常工作的需要，有时候也许你会碰到比较复杂的需求，如果真遇到了就去请教一下google吧。下面出几道关于awk的练习题，希望你认真完成。

1. 用awk 打印整个test.txt（以下操作都是用awk工具实现，针对test.txt）；
2. 查找所有包含 'bash' 的行；
3. 用 ':' 作为分隔符，查找第三段等于0的行；
4. 用 ':' 作为分隔符，查找第一段为 'root' 的行，并把该段的 'root' 换成 'toor'（可以连同sed一起使用）；
5. 用 ':' 作为分隔符，打印最后一段；
6. 打印行数大于20的所有行；
7. 用 ':' 作为分隔符，打印所有第三段小于第四段的行；
8. 用 ':' 作为分隔符，打印第一段以及最后一段，并且中间用 '@' 连接（例如，第一行应该是这样的形式 "root@/bin/bash"）；
9. 用 ':' 作为分隔符，把整个文档的第四段相加，求和；

下面给出答案：

1. `awk " test.txt`
2. `awk '/bash/' test.txt`
3. `awk -F:' '$3=="0" test.txt`
4. `awk -F:' '$1=="root" test.txt |sed 's/root/toor/'`
5. `awk -F:' ' test.txt`
6. `awk -F:' 'NR>20' test.txt`
7. `awk -F:' '$3<$4' test.txt`
8. `awk -F:' ' test.txt`
9. `awk -F:' '{(sum+= $4)}; END ' test.txt`



T



14

shell 脚本



终于到shell脚本这章了，在以前笔者卖了好多关子说shell脚本怎么怎么重要，确实shell脚本在linux系统管理员的运维工作中非常非常重要。下面笔者就带你正式进入shell脚本的世界吧。

到现在为止，你明白什么是shell脚本吗？如果明白最好了，不明白也没有关系，相信随着学习的深入你就会越来越了解到底什么是shell脚本。首先它是一个脚本，并不能作为正式的编程语言。因为是跑在linux的shell中，所以叫shell脚本。说白了，shell脚本就是一些命令的集合。举个例子，我想实现这样的操作：1) 进入到/tmp/目录；2) 列出当前目录中所有的文件名；3) 把所有当前的文件拷贝到/root/目录下；4) 删除当前目录下所有的文件。简单的4步在shell窗口中需要你敲4次命令，按4次回车。这样是不是很麻烦？当然这4步操作非常简单，如果是更加复杂的命令设置需要几十次操作呢？那样的话一次又一次敲键盘会很麻烦。所以不妨把所有的操作都记录到一个文档中，然后去调用文档中的命令，这样一步操作就可以完成。其实这个文档呢就是shell脚本了，只是这个shell脚本有它特殊的格式。

Shell脚本能帮助我们很方便的去管理服务器，因为我们可以指定一个任务计划定时去执行某一个shell脚本实现我们想要需求。这对于linux系统管理员来说是一件非常值得自豪的事情。现在的139邮箱很好用，发邮件的同时还可以发一条邮件通知的短信给用户，利用这点，我们就可以在我们的linux服务器上部署监控的shell脚本，比如网卡流量有异常了或者服务器web服务器停止了就可以发一封邮件给管理员，同时发送给管理员一个报警短信这样可以让我们及时的知道服务器出问题了。

有一个问题需要约定一下，凡是自定义的脚本建议放到/usr/local/sbin/目录下，这样做的目的是，一来可以更好的管理文档；二来以后接管你的管理员都知道自定义脚本放在哪里，方便维护。

【shell脚本的基本结构以及如何执行】

```
[root@localhost ~]# vim test.sh
#!/bin/bash

## This is my first shell script.
## Writen by Aming 2011-05-20.

date
echo "Hello World."
```

图片 14.1 14_1.png.jpg

Shell脚本通常都是以.sh为后缀名的，这个并不是说不带.sh这个脚本就不能执行，只是大家的一个习惯而已。所以，以后你发现了.sh为后缀的文件那么它一定会是一个shell脚本了。test.sh中第一行一定是“#!/bin/bash”它代表的意思是，该文件使用的是bash语法。如果不设置该行，那么你的shell脚本就不能被执行。’#’表示注释，在前面讲过的。后面跟一些该脚本的相关注释内容以及作者和创建日期或者版本等等。当然这些注释并非必须的，如果你懒的很，可以省略掉，但是笔者不建议省略。因为随着你工作时间的增加，你写的shell脚本也会越来越多，如果有一天你回头查看你写的某个脚本时，很有可能忘记该脚本是用来干什么的以及什么时候写的。所以写上注释是有必要的。另外系统管理员并非你一个，如果是其他管理员查看你的脚本，他看不懂岂不是很郁闷。该脚本再往下面则为要运行的命令了。

```
[root@localhost ~]# sh test.sh
Fri May 20 11:20:03 CST 2011
Hello World.
```

图片 14.2 14_7.png.jpg

Shell脚本的执行很简单，直接” sh filename “即可，另外你还可以这样执行

```
[root@localhost ~]# chmod +x test.sh
[root@localhost ~]# ./test.sh
Fri May 20 11:21:17 CST 2011
Hello World.
```

图片 14.3 14_8.png.jpg

默认我们用vim编辑的文档是不带有执行权限的，所以需要加一个执行权限，那样就可以直接使用’ .filename’ 执行这个脚本了。另外使用sh命令去执行一个shell脚本的时候是可以加-x选项来查看这个脚本执行过程的，这样有利于我们调试这个脚本哪里出了问题。

```
[root@localhost ~]# sh -x test.sh
+ date
Fri May 20 11:24:53 CST 2011
+ echo 'Hello World.'
Hello World.
```

图片 14.4 14_9.png.jpg

该shell脚本中用到了’ date’ 这个命令，它的作用就是用来打印当前系统的时间。其实在shell脚本中date使用率非常高。有几个选项笔者常常在shell脚本中用到：

```
[root@localhost ~]# date "+%Y%m%d %H:%M:%S"
20110520 11:59:51
```

图片 14.5 14_10.png.jpg

%Y表示年，%m表示月，%d表示日期，%H表示小时，%M表示分钟，%S表示秒

```
[root@localhost ~]# date "+%y%m%d"
110520
```

图片 14.6 14_11.png.jpg

注意%y和%Y的区别。

```
[root@localhost ~]# date -d "-1 day" "+%Y%m%d"
20110519
[root@localhost ~]# date -d "+1 day" "+%Y%m%d"
20110521
```

图片 14.7 14_21.png.jpg

-d 选项也是经常要用到的，它可以打印n天前或者n天后的日期，当然也可以打印n个月/年前或者后的日期。

```
[root@localhost ~]# date -d "-1 month" "+%Y%m%d"
20110420
[root@localhost ~]# date -d "-1 year" "+%Y%m%d"
20100520
```

图片 14.8 14_22.png.jpg

另外星期几也是常用的

```
[root@localhost ~]# date +%w
5
```

图片 14.9 14_23.png.jpg

【shell脚本中的变量】

在shell脚本中使用变量显得我们的脚本更加专业更像是一门语言，开个玩笑，变量的作用当然不是为了专业。如果你写了一个长达1000行的shell脚本，并且脚本中出现了某一个命令或者路径几百次。突然你觉得路径不对想换一下，那岂不是要更改几百次？你固然可以使用批量替换的命令，但是也是很麻烦，并且脚本显得臃肿了很多。变量的作用就是用来解决这个问题的。

```
[root@localhost ~]# vim test2.sh
#!/bin/bash

# In this script we will use variables.
# Written by Aming 2011-05-20.

d=`date +%H:%M:%S`
echo "the script begin at $d"
echo "now we will sleep 2 seconds."
sleep 2
d1=`date +%H:%M:%S`
echo "the script end at $d1"
~
```

图片 14.10 14_24.png.jpg

在test2.sh中使用到了反引号，你是否还记得它的作用？'d'和'd1'在脚本中作为变量出现，定义变量的格式为“变量名=变量的值”。当在脚本中引用变量时需要加上'\$'符号，这跟前面讲的在shell中自定义变量是一致的。下面看看脚本执行结果吧。

```
[root@localhost ~]# sh test2.sh
the script begin at 12:10:41
now we will sleep 2 seconds.
the script end at 12:10:43
```

图片 14.11 14_25.png.jpg

下面我们用shell计算两个数的和。

```
[root@localhost ~]# vim test3.sh
#!/bin/bash

# Shell script - test3.sh
# For get tow numbers' sum
# Aming 2011-05-20

a=1
b=2
sum=$((a+b))

echo "sum is $sum"
~
```

图片 14.12 14_26.png.jpg

数学计算要用 `[]` 括起来并且外头要带一个 `'$'`。脚本结果为：

```
[root@localhost ~]# sh test3.sh
sum is 3
```

图片 14.13 14_27.png.jpg

Shell脚本还可以和用户交互。

```
[root@localhost ~]# vim test4.sh
#!/bin/bash

# Shell script test4.sh
# Using "read" in the script
# Aming 2011-05-20

echo "Please input a number:"
read x
echo "Please input another number:"
read y
sum=$((x+y))
echo "The sum of tow numbers is: $sum."
~
```

图片 14.14 14_28.png.jpg

这就用到了`read`命令了，它可以从标准输入获得变量的值，后跟变量名。”`read x`”表示`x`变量的值需要用户通过键盘输入得到。脚本执行过程如下：

```
[root@localhost ~]# sh test4.sh
Please input a number:
1
Please input another number:
10
The sum of tow numbers is: 11.
```

图片 14.15 14_29.png.jpg

我们不妨加上`-x`选项再来看看这个执行过程：

```
[root@localhost ~]# sh -x !$
sh -x test4.sh
+ echo 'Please input a number:'
Please input a number:
+ read x
2
+ echo 'Please input another number:'
Please input another number:
+ read y
3
+ sum=5
+ echo 'The sum of tow numbers is: 5.'
The sum of tow numbers is: 5.
```

图片 14.16 14_44.png.jpg

在test4.sh中还有更加简洁的方式。

```
[root@localhost ~]# vim test5.sh
#!/bin/bash

# Shell script test5.sh
# Using "read" in the script
# Aming 2011-05-20

read -p "Please input a number: " x
read -p "Please input another number: " y
sum=$((x+y))
echo "The sum of tow numbers is: $sum."
```

图片 14.17 14_45.png.jpg

read -p 选项类似echo的作用。执行如下：

```
[root@localhost ~]# sh test5.sh
Please input a number: 1
Please input another number: 2
The sum of tow numbers is: 3.
```

图片 14.18 14_46.png.jpg

你有没有用过这样的命令” /etc/init.d/iptables restart “前面的/etc/init.d/iptables 文件其实就是一个shell脚本，为什么后面可以跟一个” restart”？这里就涉及到了shell脚本的预设变量。实际上，shell脚本在执行的时候后边是可以跟变量的，而且还可以跟多个。不妨笔者写一个脚本，你就会明白了。

```
[root@localhost ~]# vim test6.sh
#!/bin/bash

## test6.sh
## .....
## Aming 2011-05-20

sum=$(( $1+$2 ))
echo $sum
```

图片 14.19 14_47.png.jpg

执行过程如下：

```
[root@localhost ~]# sh -x test6.sh 1 2
+ sum=3
+ echo 3
3
```

图片 14.20 14_48.png.jpg

在脚本中，你会不会奇怪，哪里来的\$1和\$2，这其实就是shell脚本的预设变量，其中\$1的值就是在执行的时候输入的1，而\$2的值就是执行的时候输入的\$2，当然一个shell脚本的预设变量是没有限制的，这回你明白了吧。另外还有一个\$0，不过它代表的是脚本本身的名字。不妨把脚本修改一下。

```
[root@localhost ~]# vim test6.sh
#!/bin/bash

## test6.sh
## .....
## Aming 2011-05-20

echo "$0 $1 $2"
```

图片 14.21 14_49.png.jpg

执行结果想必你也猜到了吧。

```
[root@localhost ~]# sh test6.sh 1 2
test6.sh 1 2
```

图片 14.22 14_50.png.jpg

【shell脚本中的逻辑判断】

如果你学过C或者其他语言，相信你不会对if陌生，在shell脚本中我们同样可以使用if逻辑判断。在shell中if判断的基本语法为：

1) 不带else

if 判断语句; then

command

fi

```
[root@localhost ~]# vim if1.sh
#!/bin/bash
# Shell script if1.sh
# Using "if" in script.
# Aming 2011-05-20

read -p "Please input your score: " a
if ((a<60)) ; then
    echo "You didn't pass the exam."
fi
```

图片 14.23 14_51.png.jpg

在if1.sh中出现了((a<60))这样的形式，这是shell脚本中特有的格式，用一个小括号或者不用都会报错，请记住这个格式，即可。执行结果为：

```
[root@localhost ~]# sh if1.sh
Please input your score: 30
You didn't pass the exam.
```

图片 14.24 14_52.png.jpg

2) 带有else

if 判断语句 ; then

command

else

command

fi

```
[root@localhost ~]# vim if2.sh
#!/bin/bash
# Shell script if2.sh
# Using "if" in script.
# Aming 2011-05-20

read -p "Please input your score: " a
if ((a<60)) ; then
    echo "You didn't pass the exam."
else
    echo "Good! You passed the exam."
fi
```

图片 14.25 14_53.png.jpg

执行结果为：

```
[root@localhost ~]# sh if2.sh
Please input your score: 90
Good! You passed the exam.
```

图片 14.26 14_54.png.jpg

3) 带有elif

if 判断语句一 ; then

command

elif 判断语句二; then

command

else

command

fi

```
[root@localhost ~]# vim if3.sh
#!/bin/bash
# Shell script if3.sh
# Using "if" in script.
# Aming 2011-05-20

read -p "Please input your score: " a
if ((a<60)) ; then
    echo "You didn't pass the exam."
elif ((a>60)) && ((a<85)); then
    echo "Good! You passed the exam."
else
    echo "Very good! Your Score is very high!"
fi
```

图片 14.27 14_55.png.jpg

这里的 && 表示“并且”的意思，当然你也可以使用 || 表示“或者”，执行结果：

```
[root@localhost ~]# sh if3.sh
Please input your score: 99
Very good! Your Score is very high!
[root@localhost ~]# sh if3.sh
Please input your score: 50
You didn't pass the exam.
[root@localhost ~]# sh if3.sh
Please input your score: 70
Good! You passed the exam.
```

图片 14.28 14_56.png.jpg

以上只是简单的介绍了if语句的结构。在判断数值大小除了可以用“(())”的形式外，还可以使用“[]”。但是就不能使用>, <, = 这样的符号了，要使用 -lt（小于），-gt（大于），-le（小于等于），-ge（大于等于），-eq（等于），-ne（不等于）。


```
[root@localhost ~]# a=10; if [ $a -lt 5 ]; then echo ok; fi
[root@localhost ~]# a=10; if [ $a -gt 5 ]; then echo ok; fi
ok
[root@localhost ~]# a=10; if [ $a -ge 10 ]; then echo ok; fi
ok
[root@localhost ~]# a=10; if [ $a -le 10 ]; then echo ok; fi
ok
[root@localhost ~]# a=10; if [ $a -eq 10 ]; then echo ok; fi
ok
[root@localhost ~]# a=10; if [ $a -ne 10 ]; then echo ok; fi
```

图片 14.29 14_57.png.jpg

再看看if中使用 && 和 ||的情况。

```
[root@localhost ~]# a=10; if [ $a -lt 1 ] || [ $a -gt 5 ]; then echo ok; fi
ok
[root@localhost ~]# a=8; if [ $a -gt 1 ] && [ $a -lt 10 ]; then echo ok; fi
ok
```

图片 14.30 14_71.png.jpg

shell 脚本中if还经常判断关于档案属性，比如判断是普通文件还是目录，判断文件是否有读写执行权限等。常用的也就几个选项：

- e：判断文件或目录是否存在
- d：判断是不是目录，并是否存在
- f：判断是否是普通文件，并存在
- r：判断文档是否有读权限
- w：判断是否有写权限
- x：判断是否可执行

使用if判断时，具体格式为：if [-e filename]; then

```
[root@localhost ~]# if [ -d /home ]; then echo ok; fi
ok
[root@localhost ~]# if [ -f /home ]; then echo ok; fi
[root@localhost ~]# if [ -f test.txt ]; then echo ok; fi
ok
[root@localhost ~]# if [ -e test.txt ]; then echo ok; fi
ok
[root@localhost ~]# if [ -e test.txt1 ]; then echo ok; fi
[root@localhost ~]# if [ -r test.txt ]; then echo ok; fi
ok
[root@localhost ~]# if [ -w test.txt ]; then echo ok; fi
ok
[root@localhost ~]# if [ -x test.txt ]; then echo ok; fi
```

图片 14.31 14_72.png.jpg

在shell 脚本中，除了用if来判断逻辑外，还有一种常用的方式，那就是case了。具体格式为：

```
case 变量 in
```

```
value1)
```

```
command
```

```
::
```

```
value2)
```

```
command
```

```
::
```

```
value3)
```

```
command
```

```
::
```

```
*)
```

```
command
```

```
::
```

```
esac
```

上面的结构中，不限制value的个数，*则代表除了上面的value外的其他值。下面笔者写一个判断输入数值是奇数或者偶数的脚本。

```
[root@localhost ~]# vim case.sh
#!/bin/bash

# Using "case" in shell script.
# Aming 2011-05-20

read -p "Please input a number: " n
a=${n%2}
case $a in
1)
    echo "The number is odd"
;;
0)
    echo "The number is even"
;;
esac
```

图片 14.32 14_73.png.jpg

\$a 的值或为1或为0，执行结果为：

```
[root@localhost ~]# sh case.sh
Please input a number: 100
The number is even
[root@localhost ~]# sh case.sh
Please input a number: 101
The number is odd
```

图片 14.33 14_74.png.jpg

也可以看一下执行过程：

```
[root@localhost ~]# sh -x case.sh
+ read -p 'Please input a number: ' n
Please input a number: 100
+ a=0
+ case $a in
+ echo 'The number is even'
The number is even
```

图片 14.34 14_75.png.jpg

case脚本常用于编写系统服务的启动脚本，例如/etc/init.d/iptables中就用到了，你不妨去查看一下。

【shell脚本中的循环】

Shell脚本中也算是一门简易的编程语言了，当然循环是不能缺少的。常用到的循环有for循环和while循环。下面就分别介绍一下两种循环的结构。

```
[root@localhost ~]# vim for.sh
#!/bin/bash
# Shell script for.sh.
# Using "for"
# Aming 2011-05-20
for i in `seq 1 5`; do
    echo $i
done
```

图片 14.35 14_76.png.jpg

脚本中的seq 1 5 表示从1到5的一个序列。你可以直接运行这个命令试下。脚本执行结果为：

```
[root@localhost ~]# sh for.sh
1
2
3
4
5
```

图片 14.36 14_77.png.jpg

通过这个脚本就可以看到for循环的基本结构：

for 变量名 in 循环的条件; do

command

done

```
[root@localhost ~]# for i in 1 2 3 4 5; do echo $i; done
1
2
3
4
5
```

图片 14.37 14_78.png.jpg

循环的条件那一部分也可以写成这样的形式，中间用空格隔开即可。你也可以试试，for i in ls ; do echo \$i; done 和 for i in cat test.txt ; do echo \$i; done

```
[root@localhost ~]# vim while.sh
#!/bin/bash
# while.sh
# Using while
# Aming 2011-05-20
a=10
while [ $a -ge 1 ]; do
    echo "$a"
    a=$((a-1))
done
```

图片 14.38 14_79.png.jpg

再来看看这个while循环，基本格式为：

```
while 条件; do
```

```
command
```

```
done
```

脚本的执行结果为：



```
[root@localhost ~]# sh while.sh
10
9
8
7
6
5
4
3
2
1
```

图片 14.39 14_80.png.jpg

另外你可以把循环条件忽略掉，笔者常常这样写监控脚本。

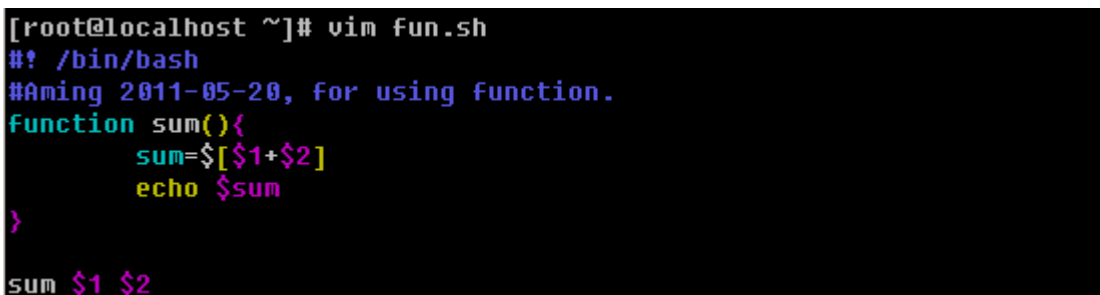
```
while ;; do
```

```
command
```

```
done
```

【shell脚本中的函数】

如果你学过开发，肯定知道函数的作用。如果你是刚刚接触到这个概念的话，也没有关系，其实很好理解的。函数就是把一段代码整理到了一个小单元中，并给这个小单元起一个名字，当用到这段代码时直接调用这个小单元的名字即可。有时候脚本中的某段代总是重复使用，如果写成函数，每次用到时直接用函数名代替即可，这样就节省了时间还节省了空间。



```
[root@localhost ~]# vim fun.sh
#!/bin/bash
#Aming 2011-05-20, for using function.
function sum(){
    sum=${$1+$2}
    echo $sum
}

sum $1 $2
```

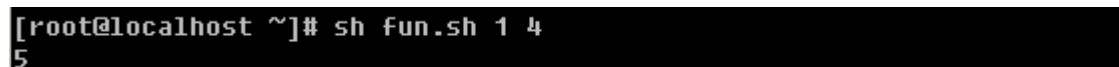
图片 14.40 14_81.png.jpg

fun.sh 中的sum() 为自定义的函数，在shell脚本中要用

```
function 函数名(){
    command
}
```

这样的格式去定义函数。

上个脚本执行过程如下：



```
[root@localhost ~]# sh fun.sh 1 4
5
```

图片 14.41 14_82.png.jpg

有一点笔者要提醒你一下，在shell脚本中，函数一定要写在最前面，不能出现在中间或者最后，因为函数是要被调用的，如果还没有出现就被调用，肯定是会出错的。

Shell脚本大体上就介绍这么多了，笔者所举的例子都是最基础的，所以即使你把所有例子完全掌握也不代表你的shell脚本编写能力有多么好。所以剩下的日子里你尽量要多练习，多写脚本，你写的脚本越多，你的能力就越强。希望你能够找专门介绍shell脚本的书籍深入的去研究一下它。随后笔者将给你留几个shell脚本的练习题，你最好不要偷懒。

1. 编写shell脚本，计算1-100的和；
2. 编写shell脚本，要求输入一个数字，然后计算出从1到输入数字的和，要求，如果输入的数字小于1，则重新输入，直到输入正确的数字为止；
3. 编写shell脚本，把/root/目录下的所有目录（只需要一级）拷贝到/tmp/目录下；
4. 编写shell脚本，批量建立用户user_00, user_01, ..., user_100并且所有用户同属于users组；
5. 编写shell脚本，截取文件test.log中包含关键词' abc' 的行中的第一列（假设分隔符为" :"），然后把截取的数字排序（假设第一列为数字），然后打印出重复次数超过10次的列；
6. 编写shell脚本，判断输入的IP是否正确（IP的规则是，n1.n2.n3.n4，其中1<n1<255, 0<n2<255, 0<n3<255, 0<n4<255）。

以下为练习题答案：

```
1. #! /bin/bash
```

```
sum=0
```

```
for i in seq 1 100 ; do
```

```
sum=$((i+$sum))
```

```
done
```

```
echo $sum
```

```
2. #!/bin/bash
```

```
n=0
```

```
while [ $n -lt "1" ]; do
```

```
read -p "Please input a number, it must greater than "1": " n
```

```
done
```

```
sum=0
```

```
for i in seq 1 $n ; do
```

```
sum=$((i+$sum))
```

```
done
```

```
echo $sum
```

```
3. #!/bin/bash
```

```
for f in ls /root/ ; do
```

```
if [ -d $f ] ; then
```

```
cp -r $f /tmp/
```

```
fi
```

```
done
```

```
4. #!/bin/bash
```

```
groupadd users

for i in `seq 0 9` ; do

useradd -g users user_0$i

done
```

```
for j in `seq 10 100` ; do

useradd -g users user_0$j

done
```

```
5. #!/bin/bash
```

```
awk -F:' '$0~/abc/ ' test.log >/tmp/n.txt

sort -n n.txt |uniq -c |sort -n >/tmp/n2.txt

awk '$1>10 ' /tmp/n2.txt
```

```
6. #!/bin/bash
```

```
checkip() {

if echo $1 |egrep -q '^[0-9].[0-9].[0-9].[0-9]/span> ; then

a=`echo $1 | awk -F. "`

b=`echo $1 | awk -F. "`

c=`echo $1 | awk -F. "`

d=`echo $1 | awk -F. "`
```

```
for n in $a $b $c $d; do
```

```
if [ $n -ge 255 ] || [ $n -le 0 ]; then
```



```
echo "the number of the IP should less than 255 and greate than 0"

return 2

fi

done

else

echo "The IP you input is something wrong, the format is like 192.168.100.1"

return 1

fi

}

rs=1

while [ $rs -gt 0 ]; do

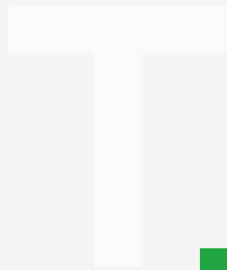
read -p "Please input the ip:" ip

checkip $ip

rs= echo $?

done

echo "The IP is right!"
```



15

linux 系统日常管理



笔者在前面介绍的内容都为linux系统基础类的，如果你现在把前面的内容全部很好的掌握了，那最好了。不过笔者要说的是，即使你完全掌握了，你现在还是不能作为一名合格的linux系统管理员的，毕竟系统管理员要会做的事情太多了。本章以及后面章节笔者会陆续教给你作为linux系统管理员所必备的知识。只要你熟练掌握那绝对可以胜任一个最初级的管理员职位，不过只是初级的，因为你还需要在日常的管理工作中获得成长。

【监控系统的状态】

1. w 查看当前系统的负载

```
[root@localhost ~]# w
 10:19:19 up 2:15, 3 users, load average: 0.26, 0.15, 0.05
USER      TTY      FROM          LOGIN@      IDLE        JCPU        PCPU        WHAT
root      tty1     -             10:13       16.00s      0.52s       0.52s      -bash
root      pts/0    10.0.2.34     10:12       5:58        0.19s       0.19s      -bash
```

图片 15.1 15_1.png.jpg

相信所有的linux管理员最常用的命令就是这个'w'了，该命令显示的信息还是蛮丰富的。第一行从左面开始显示的信息依次为：时间，系统运行时间，登录用户数，平均负载。第二行开始以及下面所有的行，告诉我们的信息是，当前登录的都有哪些用户，以及他们是从哪里登录的等等。其实，在这些信息当中，笔者认为我们最应该关注的应该是第一行中的'load average:'后面的三个数值。

第一个数值表示1分钟内系统的平均负载值；第二个数值表示5分钟内系统的平均负载值；第三个数值表示15分钟系统的平均负载值。这个值的意义是，单位时间段内CPU活动进程数。当然这个值越大就说明你的服务器压力越大。一般情况下这个值只要不超过你服务器的cpu数量就没有关系，如果你的服务器cpu数量为8，那么这个值若小于8，就说明你的服务器没有压力，否则就要关注一下了。到这里你肯定会问，如何查看服务器有几个cpu？

```
[root@localhost ~]# cat /proc/cpuinfo
processor       : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 15
model name    : Intel(R) Core(TM)2 Duo CPU       L7500  @ 1.60GHz
stepping      : 8
cpu MHz       : 1596.000
cache size    : 4096 KB
fdiv_bug      : no
hlt_bug       : no
f00f_bug      : no
coma_bug      : no
fpu           : yes
fpu_exception : yes
cpuid level   : 10
wp            : yes
flags         : fpu vme de pse tsc msr pae mce cx8 apic mtrr pge mca cmov pat
pse36 clflush dts acpi mmx fxsr sse sse2 ss nx constant_tsc up ida pni ds_cpl
bogomips     : 3192.00
```

图片 15.2 15_7.png.jpg

就是用这个命令了。'/proc/cpuinfo' 这个文件记录了cpu的详细信息。目前市面上的服务器通常都是2颗4核cpu，在linux看来，它就是8个cpu。查看这个文件时则会显示8段类似的信息，而最后一段信息中processor：后面跟的是'7'。所以查看当前系统有几个cpu，你可以使用这个命令：'grep -c 'processor' /proc/cpuinfo'

。

```
[root@localhost ~]# grep -c 'processor' /proc/cpuinfo
1
```

图片 15.3 15_8.png.jpg

2. vmstat 监控系统的状态

```
[root@localhost ~]# vmstat
procs -----memory----- ---swap-- -----io----- --system-- -----cpu-----
r  b   swpd   free   buff  cache   si   so   bi   bo   in   cs us sy id wa st
0  0     0 175296  6692  34272   0   0    5    2 1002   7  0  1 99  0  0
```

图片 15.4 15_9.png.jpg

上面讲的w查看的是系统整体上的负载，通过看那个数值可以知道当前系统有没有压力，但是具体是哪里（CPU，内存，磁盘等）有压力就无法判断了。通过vmstat就可以知道具体是哪里有压力。vmstat命令打印的结果共分为6部分：procs, memory, swap, io, system, cpu.请重点关注一下红色标出的项。

1) procs 显示进程相关信息

r: 表示运行和等待cpu时间片的进程数，如果长期大于服务器cpu的个数，则说明cpu不够用了；

b: 表示等待资源的进程数，比如等待I/O, 内存等，这列的值如果长时间大于1，则需要你关注一下了；

2) memory 内存相关信息

swpd : 表示切换到交换分区中的内存数量；

free : 当前空闲的内存数量；

buff : 缓冲大小，（即将写入磁盘的）；

cache : 缓存大小，（从磁盘中读取的）；

3) swap 内存交换情况

si : 由内存进入交换区的数量；

so: 由交换区进入内存的数量；

4) io 磁盘使用情况

bi：从块设备读取数据的量（读磁盘）；

bo：从块设备写入数据的量（写磁盘）；

5) system 显示采集间隔内发生的中断次数

in：表示在某一时间间隔中观测到的每秒设备中断数；

cs：表示每秒产生的上下文切换次数；

6) CPU 显示cpu的使用状态

us：显示了用户下所花费 cpu 时间的百分比；

sy：显示系统花费cpu时间百分比；

id：表示cpu处于空闲状态的时间百分比；

wa：表示I/O等待所占用cpu时间百分比；

st：表示被偷走的cpu所占百分比（一般都为0，不用关注）；

以上所介绍的各个参数中，笔者经常会关注r列，b列，和wa列，三列代表的含义在上边说得已经很清楚。IO部分的bi以及bo也是我要经常参考的对象。如果磁盘io压力很大时，这两列的数值会比较高。另外当si，so两列的数值比较高，并且在不断变化时，说明内存不够了，内存中的数据频繁交换到交换分区中，这往往对系统性能影响极大。

```
[root@localhost ~]# vmstat 1 5
procs -----memory----- ---swap-- ----io---- --system-- -----cpu-----
r  b   swpd   free   buff  cache   si   so   bi   bo   in   cs  us  sy  id  wa  st
0  0     0 175176  6788  34272   0   0    3    1 1001    6  0  0  99  0  0
0  0     0 175176  6788  34272   0   0    0    0 1005    6  0  0 100  0  0
0  0     0 175176  6788  34272   0   0    0    0 1005    7  0  0 100  0  0
0  0     0 175176  6788  34272   0   0    0    0 1005    6  0  1  99  0  0
0  0     0 175176  6788  34272   0   0    0    0 1009    9  0  0 100  0  0
```

图片 15.5 15_10.png.jpg

笔者用vmstat时，经常用这样的形式，'vmstat 1 5' 表示每隔1秒钟打印一次系统状态，连续打印5次。当然你也可以 'vmstat 1 ' 表示每隔1秒钟打印一次系统状态，一直打印，除非你按ctrl + c强制结束。

3. top 显示进程所占系统资源

```
[root@localhost ~]# top
top - 12:02:19 up 3:58, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 43 total, 1 running, 42 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.1%us, 0.3%sy, 0.0%ni, 99.4%id, 0.0%wa, 0.1%hi, 0.1%si, 0.0%st
Mem: 235128k total, 60008k used, 175120k free, 6800k buffers
Swap: 409648k total, 0k used, 409648k free, 34324k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	15	0	2068	648	556	S	0.0	0.3	0:01.66	init
2	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	migration/0
3	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/0
4	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
5	root	10	-5	0	0	0	S	0.0	0.0	0:00.17	events/0
6	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	khelper
7	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	kthread
10	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	kblockd/0
11	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	kacpid
72	root	19	-5	0	0	0	S	0.0	0.0	0:00.00	cqueue/0
75	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	khudd
77	root	10	-5	0	0	0	S	0.0	0.0	0:00.02	kseriod

图片 15.6 15_11.png.jpg

这个命令用于动态监控进程所占系统资源，每隔3秒变一次。这个命令的特点是把占用系统资源（CPU，内存，磁盘IO等）最高的进程放到最前面。top命令打印出了很多信息，包括系统负载（load average）、进程数（Tasks）、cpu使用情况、内存使用情况以及交换分区使用情况。其实上面这些内容可以通过其他命令来查看，所以用top重点查看的还是下面的进程使用系统资源详细状况。这部分东西反映的东西还是比较多的，不过需要你关注的也就是几项：%CPU, %MEM, COMMAND 这些项目所代表的意义，不用笔者介绍相信你能看懂吧。

```
[root@localhost ~]# top -bn1
top - 12:07:51 up 4:03, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 43 total, 1 running, 42 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.1%us, 0.3%sy, 0.0%ni, 99.4%id, 0.0%wa, 0.1%hi, 0.1%si, 0.0%st
Mem: 235128k total, 60068k used, 175060k free, 6856k buffers
Swap: 409648k total, 0k used, 409648k free, 34332k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	15	0	2068	648	556	S	0.0	0.3	0:01.66	init
2	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	migration/0
3	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/0
4	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
5	root	10	-5	0	0	0	S	0.0	0.0	0:00.17	events/0
6	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	khelper
7	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	kthread
10	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	kblockd/0
11	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	kacpid
72	root	19	-5	0	0	0	S	0.0	0.0	0:00.00	cqueue/0
75	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	khudd
77	root	10	-5	0	0	0	S	0.0	0.0	0:00.02	kseriod

图片 15.7 15_21.png.jpg

另外笔者使用top命令时还常常使用-bn1 这个组合选项，它表示非动态打印系统资源使用情况，可以用在脚本中，你不妨记一下，以后也许你会用得到。

4. sar 监控系统状态

sar 命令很强大，它可以监控系统所有资源状态，比如平均负载、网卡流量、磁盘状态、内存使用等等。它不同于其他系统状态监控工具的地方在于，它可以打印历史信息，可以显示当天从零点开始到当前时刻的系统状态信息。如果你系统没有安装这个命令，请使用” yum install -y sysstat” 命令安装。初次使用sar命令会报错，那是因为sar工具还没有生成相应的数据库文件（时时监控就不会了，因为不用去查询那个库文件）。它的数据库文件在” /var/log/sa/” 目录下，默认保存9天。因为这个命令太过复杂，所以笔者只介绍几个。

1) 查看网卡流量 ‘sar -n DEV ‘

```
[root@localhost tmp]# sar -n DEV
Linux 2.6.18-164.el5 (localhost)          05/21/2011

01:02:30 PM      IFACE      rxpck/s      txpck/s      rxbyt/s      txbyt/s      rxcmp/s
01:02:31 PM          lo          0.00          0.00          0.00          0.00          0.00
01:02:31 PM        eth0          0.99          0.99          59.41         136.63          0.00
01:02:31 PM        sit0          0.00          0.00          0.00          0.00          0.00
01:02:32 PM          lo          0.00          0.00          0.00          0.00          0.00
01:02:32 PM        eth0          4.00          3.00         240.00         478.00          0.00
01:02:32 PM        sit0          0.00          0.00          0.00          0.00          0.00
01:02:33 PM          lo          0.00          0.00          0.00          0.00          0.00
01:02:33 PM        eth0          2.00          1.00         120.00         186.00          0.00
01:02:33 PM        sit0          0.00          0.00          0.00          0.00          0.00
01:02:34 PM          lo          0.00          0.00          0.00          0.00          0.00
```

图片 15.8 15_22.png.jpg

IFACE这列表示设备名称，rxpck/s 表示每秒进入收取的包的数量，txpck/s 表示每秒发送出去的包的数量，rxbyt/s 表示每秒收取的数据量（单位Byte），txbyt/s表示每秒发送的数据量。后面几列不需要关注。如果有一天你所管理的服务器丢包非常严重，那么你就应该看一看这个网卡流量是否异常了，如果rxpck/s 那一列的数值大于4000，或者rxbyt/s那列大于5,000,000则很有可能是被攻击了，正常的服务器网卡流量不会高于这么多，除非是你自己在拷贝数据。上面的命令是查看网卡流量历史的，如何时时查看网卡流量呢？

```
[root@localhost tmp]# sar -n DEV 1 5
Linux 2.6.18-164.el5 (localhost)      05/21/2011

01:16:20 PM      IFACE      rxpck/s      txpck/s      rxbyt/s      txbyt/s      rxcmp/s
01:16:21 PM          lo          0.00          0.00          0.00          0.00          0.00
01:16:21 PM        eth0          1.00          0.00          60.00          0.00          0.00
01:16:21 PM        sit0          0.00          0.00          0.00          0.00          0.00

01:16:21 PM      IFACE      rxpck/s      txpck/s      rxbyt/s      txbyt/s      rxcmp/s
01:16:22 PM          lo          0.00          0.00          0.00          0.00          0.00
01:16:22 PM        eth0          3.96          4.95          237.62         841.58          0.00
01:16:22 PM        sit0          0.00          0.00          0.00          0.00          0.00

01:16:22 PM      IFACE      rxpck/s      txpck/s      rxbyt/s      txbyt/s      rxcmp/s
01:16:23 PM          lo          0.00          0.00          0.00          0.00          0.00
01:16:23 PM        eth0          4.00          5.00          240.00         850.00          0.00
01:16:23 PM        sit0          0.00          0.00          0.00          0.00          0.00

01:16:23 PM      IFACE      rxpck/s      txpck/s      rxbyt/s      txbyt/s      rxcmp/s
01:16:24 PM          lo          0.00          0.00          0.00          0.00          0.00
01:16:24 PM        eth0          3.00          5.00          180.00         850.00          0.00
01:16:24 PM        sit0          0.00          0.00          0.00          0.00          0.00
```

图片 15.9 15_23.png.jpg

另外也可以查看某一天的网卡流量历史，使用-f选项，后面跟文件名，如果你的系统格式Redhat或者CentOS那么sar的库文件一定是在/var/log/sa/目录下的。

```
[root@localhost tmp]# sar -n DEV -f /var/log/sa/sa21
Linux 2.6.18-164.el5 (localhost)      05/21/2011

01:02:30 PM      IFACE      rxpck/s      txpck/s      rxbyt/s      txbyt/s      rxcmp/s
01:02:31 PM          lo          0.00          0.00          0.00          0.00          0.00
01:02:31 PM        eth0          0.99          0.99          59.41          136.63          0.00
01:02:31 PM        sit0          0.00          0.00          0.00          0.00          0.00
01:02:32 PM          lo          0.00          0.00          0.00          0.00          0.00
01:02:32 PM        eth0          4.00          3.00          240.00         478.00          0.00
01:02:32 PM        sit0          0.00          0.00          0.00          0.00          0.00
01:02:33 PM          lo          0.00          0.00          0.00          0.00          0.00
01:02:33 PM        eth0          2.00          1.00          120.00         186.00          0.00
01:02:33 PM        sit0          0.00          0.00          0.00          0.00          0.00
01:02:34 PM          lo          0.00          0.00          0.00          0.00          0.00
01:02:34 PM        eth0          2.97          0.99          241.58         184.16          0.00
```

图片 15.10 15_24.png.jpg

2) 查看历史负载 ‘sar -q’


```
[root@localhost tmp]# sar -q
Linux 2.6.18-164.el5 (localhost)      05/21/2011

01:02:30 PM   runq-sz   plist-sz   ldavg-1   ldavg-5   ldavg-15
01:02:31 PM         0         44         0.00         0.00         0.00
01:02:32 PM         0         44         0.00         0.00         0.00
01:02:33 PM         0         44         0.00         0.00         0.00
01:02:34 PM         0         44         0.00         0.00         0.00
01:02:35 PM         0         44         0.00         0.00         0.00
01:19:53 PM         1         44         0.00         0.00         0.00
01:19:54 PM         0         44         0.00         0.00         0.00
Average:         0         44         0.00         0.00         0.00
```

图片 15.11 15_25.png.jpg

这个命令有助于我们查看服务器在过去的某个时间的负载状况。

关于sar的介绍笔者不愿写太多，毕竟介绍太多会给你带来更多的压力，其实笔者介绍这个命令的目的只是让你学会查看网卡流量（这是非常有用的）。如果你很感兴趣那就man一下吧，它的用法太多了。

5. free查看内存使用状况

```
[root@localhost tmp]# free
              total        used         free       shared    buffers     cached
Mem:           235128      140368         94760            0         11864      106764
-/+ buffers/cache:          21740         213388
Swap:          409648            0         409648
```

图片 15.12 15_26.png.jpg

只要你敲一个free然后回车就可以当前系统的总内存大小以及使用内存的情况。从上图中可看到当前系统内存总大小为235128（单位是k）已经使用120368，剩余94760。其实真正剩余并不是这个94760，而是第二行的213388，真正使用的也是第二行的21740。这是因为系统初始化时，就已经分配出很大一部分内存给缓存，这部分缓存用来随时提供给程序使用，如果程序不用，那这部分内存就空闲。所以，查看内存使用多少，剩余多少请看第二行的数据。另外你还可以加-m 或者-g选项分别以M或G为单位打印内存使用状况。

```
[root@localhost tmp]# free -m
              total        used         free       shared    buffers     cached
Mem:             229         137            92            0            11         104
-/+ buffers/cache:           21          208
Swap:            400            0          400
```

图片 15.13 15_27.png.jpg

6. ps 查看系统进程

作为系统管理员，一定要知道你所管理的系统都有那些进程在运行，在windows下只要打开任务管理器即可查看。在linux下呢？其实在上面介绍的top命令就可以，但是不够专业，当然还有专门显示系统进程的命令。

```
[root@localhost tmp]# ps aux
USER      PID %CPU %MEM    USZ    RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.2   2068    648 ?        Ss   08:04   0:01 init [3]
root         2  0.0  0.0     0     0 ?        S<   08:04   0:00 [migration/0]
root         3  0.0  0.0     0     0 ?        SN   08:04   0:00 [ksoftirqd/0]
root         4  0.0  0.0     0     0 ?        S<   08:04   0:00 [watchdog/0]
root         5  0.0  0.0     0     0 ?        S<   08:04   0:00 [events/0]
root         6  0.0  0.0     0     0 ?        S<   08:04   0:00 [khelper]
root         7  0.0  0.0     0     0 ?        S<   08:04   0:00 [kthread]
root        10  0.0  0.0     0     0 ?        S<   08:04   0:00 [kblockd/0]
root        11  0.0  0.0     0     0 ?        S<   08:04   0:00 [kacpid]
root        72  0.0  0.0     0     0 ?        S<   08:04   0:00 [cqueue/0]
root        75  0.0  0.0     0     0 ?        S<   08:04   0:00 [khubd]
root        77  0.0  0.0     0     0 ?        S<   08:04   0:00 [kseriod]
root       140  0.0  0.0     0     0 ?        S    08:04   0:00 [pdflush]
```

图片 15.14 15_28.png.jpg

对了，就是这个‘ps aux’。笔者也经常看到有的人喜欢用‘ps -elf’大同小异，显示的信息基本上是一样的。ps命令还有更多的用法，笔者不再做介绍，因为你只要会用这个命令就足够了，请man一下。下面介绍上图上出现的几个参数的意义。

PID：进程的id，这个id很有用，在linux中内核管理进程就得靠pid来识别和管理某一个程，比如我想终止某一个进程，则用‘kill 进程的pid’，有时并不能杀掉，则需要加一个-9选项了’ kill -9 进程pid’

STAT：表示进程的状态，进程状态分为以下几种（不要求记住，但要了解）

D 不能中断的进程（通常为IO）

R 正在运行中的进程

S 已经中断的进程，通常情况下，系统中大部分进程都是这个状态

T 已经停止或者暂停的进程，如果我们正在运行一个命令，比如说sleep 10，如果我们按一下ctrl -z 让他暂停，那么我们用ps查看就会显示T这个状态

W 这个好像是说，从内核2.6xx 以后，表示为没有足够的内存页分配

X 已经死掉的进程（这个好像从来不会出现）

Z 僵尸进程，杀不掉，打不死的垃圾进程，占系统一小点资源，不过没有关系。如果太多，就有问题了。一般不会出现。

< 高优先级进程

N 低优先级进程

L 在内存中被锁了内存分页

s 主进程

l 多线程进程

+ 代表在前台运行的进程

这个ps命令是笔者在工作中用的非常多的命令之一，所以请记住它吧。关于ps命令的使用，笔者经常会连同管道符一起使用，用来查看某个进程或者它的数量。

```
[root@localhost tmp]# ps aux |grep -c mingetty
6
[root@localhost tmp]# ps aux |grep mingetty
root      1654  0.0  0.1  1660  444 tty2      Ss+  08:05   0:00 /sbin/mingetty tty2
root      1655  0.0  0.1  1660  448 tty3      Ss+  08:05   0:00 /sbin/mingetty tty3
root      1656  0.0  0.1  1660  444 tty4      Ss+  08:05   0:00 /sbin/mingetty tty4
root      1663  0.0  0.1  1660  448 tty5      Ss+  08:05   0:00 /sbin/mingetty tty5
root      1671  0.0  0.1  1660  444 tty6      Ss+  08:05   0:00 /sbin/mingetty tty6
root      3726  0.0  0.2  3912  680 pts/1    S+   13:52   0:00 grep mingetty
```

图片 15.15 15_29.png.jpg

上面的6不对，需要减掉1，因为使用grep命令时，grep命令本身也算作了一个。

7. netstat 查看网络状况

```
[root@localhost tmp]# netstat -lnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 :::22                  :::*                    LISTEN
udp        0      0 0.0.0.0:68            0.0.0.0:*
udp        0      0 0.0.0.0:68            0.0.0.0:*
Active UNIX domain sockets (only servers)
Proto RefCnt Flags               Type                   State                  I-Node PID/Program name  Path

```

图片 15.16 15_44.png.jpg

netstat命令用来打印网络连接状况、系统所开放端口、路由表等信息。笔者最常用的关于netstat的命令就是这个netstat -lnp（打印当前系统启动哪些端口）以及netstat -an（打印网络连接状况）这两个命令非常有用，请一定要记住。

```
[root@localhost tmp]# netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 :::22                  :::*                    LISTEN
tcp        0    132 ::ffff:10.0.2.68:22    ::ffff:10.0.2.34:2311  ESTABLISHED
udp        0      0 0.0.0.0:68            0.0.0.0:*
udp        0      0 0.0.0.0:68            0.0.0.0:*
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags               Type                   State                  I-Node Path
unix    2      [ ]                 DGRAM                  1518 @/org/kernel/udev/udev
unix    6      [ ]                 DGRAM                  6390 /dev/log
unix    2      [ ]                 DGRAM                  9295
unix    2      [ ]                 DGRAM                  9013
unix    2      [ ]                 DGRAM                  7407
unix    2      [ ]                 DGRAM                  6398
```

图片 15.17 15_45.png.jpg

如果你所管理的服务器是一台提供web服务（80端口）的服务器，那么你就可以使用netstat -an |grep 80开查看当前连接web服务的有哪些IP了。

8. 抓包工具tcpdump

有时候，也许你会有这样的需求，想看一下某个网卡上都有哪些数据包，尤其是当你初步判定你的服务器上有流量攻击。这时，使用抓包工具来抓一下数据包，就可以知道有哪些IP在攻击你了。

```
[root@localhost ~]# tcpdump -nn -i eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
14:17:49.056051 IP 10.0.2.68.22 > 10.0.2.34.3337: P 1155928231:1155928347(116) a
14:17:49.057533 IP 10.0.2.68.22 > 10.0.2.34.3337: P 116:232(116) ack 1 win 1132
14:17:49.058192 IP 10.0.2.34.3337 > 10.0.2.68.22: . ack 232 win 63568
14:17:49.059124 IP 10.0.2.34.3337 > 10.0.2.68.22: . ack 232 win 63568
14:17:49.057103 IP 10.0.2.68.22 > 10.0.2.34.3337: P 232:380(148) ack 1 win 1132
14:17:49.057653 IP 10.0.2.68.22 > 10.0.2.34.3337: P 380:512(132) ack 1 win 1132
14:17:49.058095 IP 10.0.2.34.3337 > 10.0.2.68.22: . ack 512 win 63888
14:17:49.058566 IP 10.0.2.68.22 > 10.0.2.34.3337: P 512:628(116) ack 1 win 1132
14:17:49.059060 IP 10.0.2.34.3337 > 10.0.2.68.22: . ack 512 win 63888
14:17:49.059237 IP 10.0.2.68.22 > 10.0.2.34.3337: P 628:744(116) ack 1 win 1132
```

图片 15.18 15_46.png.jpg

如果你没有tcpdump 这个命令，需要用 ' yum install -y tcpdump ' 命令去安装一下。上图中第三列和第四列显示的信息为哪一个IP+port在连接哪一个IP+port，后面的信息是该数据包的相关信息，如果不懂也没有关系，毕竟你不是专门搞网络的，而这里需要你关注的只是第三列以及第四列。-i 选项后面跟设备名称，如果你想抓eth1网卡的包，后面则要跟eth1.至于-nn选项的作用是让第三列和第四列显示成IP+端口号的形式，如果不加-nn则显示的是主机名+服务名称。

【linux网络相关】

1. ifconfig 查看网卡IP

ifconfig类似与windows的ipconfig，不加任何选项和参数只打印当前网卡的IP相关信息（子网掩码、网关等）

```
[root@localhost ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:AD:1B:0E
          inet addr:10.0.2.68  Bcast:10.0.2.127  Mask:255.255.255.128
          inet6 addr: fe80::20c:29ff:fead:1b0e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:77494 errors:0 dropped:0 overruns:0 frame:0
          TX packets:32905 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8483092 (8.0 MiB)  TX bytes:5957774 (5.6 MiB)
          Interrupt:169 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:744 (744.0 b)  TX bytes:744 (744.0 b)
```

图片 15.19 15_47.png.jpg

当然ifconfig后面可以跟设备名，只打印指定设备的IP信息。

```
[root@localhost ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0C:29:AD:1B:0E
          inet addr:10.0.2.68  Bcast:10.0.2.127  Mask:255.255.255.128
          inet6 addr: fe80::20c:29ff:fead:1b0e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:77555 errors:0 dropped:0 overruns:0 frame:0
          TX packets:32940 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8488071 (8.0 MiB)  TX bytes:5962332 (5.6 MiB)
          Interrupt:169 Base address:0x2000
```

图片 15.20 15_48.png.jpg

在windows下设置IP非常简单，然而在命令窗口下如何设置？这就需要去修改配置文件/etc/sysconfig/network-scripts/ifcfg-eth0了，如果是eth1那么配置文件是/etc/sysconfig/network-scripts/ifcfg-eth1。

```
[root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
# Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
DEVICE=eth0
BOOTPROTO=none
HWADDR=00:0C:29:AD:1B:0E
ONBOOT=yes
IPADDR=10.0.2.68
NETMASK=255.255.255.192
GATEWAY=10.0.2.1
TYPE=Ethernet
```

图片 15.21 15_49.png.jpg

如果想修改IP的话，则只需要修改IPADDR，NETMASK以及GATEWAY即可。如果你的linux是通过dhcp服务器自动获得的IP，那么配置文件肯定和上图中的不一样，BOOTPROTO那里会是'dhcp'，如果你要配置成静

态IP的话，这里就需要写成' none'。关于如何设置IP以及子网掩码的这些知识属于网络相关的基础知识了，如果你对这方面比较陌生的话，建议你去看看网络相关的资料。当修改完IP后需要重启网络服务新IP才能生效，重启命令为' service network restart'

```
[root@localhost ~]# service network restart
Shutting down interface eth0:           [ OK ]
Shutting down loopback interface:       [ OK ]
Bringing up loopback interface:         [ OK ]
Bringing up interface eth0:             [ OK ]
```

图片 15.22 15_50.png.jpg

另外如果你有多个网卡的情况时，只想重启某一个网卡的话，还可以使用这个命令。

```
[root@localhost ~]# ifdown eth0 ; ifup eth0
```

图片 15.23 15_51.png.jpg

ifdown 即停掉网卡，ifup即启动网卡。有一点要提醒你的是，如果你远程登录你的服务器，当你使用ifdown eth0这个命令的时候，很有可能后面的命令ifup eth0不会被运行，这样导致你断网而无法连接服务器，所以请尽量使用service network restart 这个命令来重启网卡。

2. 给一个网卡设定多个IP

在linux系统中，网卡是可以设定多重IP的，笔者曾经管理的一台服务器的eth1就设定了5个IP，实在是够变态的。

```
[root@localhost network-scripts]# cp ifcfg-eth0 ifcfg-eth0:1
[root@localhost network-scripts]# vim ifcfg-eth0:1
# Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
DEVICE=eth0:1
BOOTPROTO=none
HWADDR=00:0C:29:AD:1B:0E
ONBOOT=yes
IPADDR=10.0.2.70
NETMASK=255.255.255.128
GATEWAY=10.0.2.1
TYPE=Ethernet
```

图片 15.24 15_52.png.jpg

把ifcfg-eth0复制成ifcfg-eth0:1 然后编辑ifcfg-eth0:1修改DEVICE以及IPADDR保存后重启网卡。

```
[root@localhost network-scripts]# ifdown eth0 ; ifup eth0
[root@localhost network-scripts]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:AD:1B:0E
          inet addr:10.0.2.68  Bcast:10.0.2.127  Mask:255.255.128
          inet6 addr: fe80::20c:29ff:fead:1b0e/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:79513  errors:0  dropped:0  overruns:0  frame:0
          TX packets:33857  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8651134 (8.2 MiB)  TX bytes:6176870 (5.8 MiB)
          Interrupt:169  Base address:0x2000

eth0:1    Link encap:Ethernet  HWaddr 00:0C:29:AD:1B:0E
          inet addr:10.0.2.70  Bcast:10.0.2.127  Mask:255.255.128
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:169  Base address:0x2000
```

图片 15.25 15_53.png.jpg

再次查看eth0上就有两个IP了。这里你要注意一下，文件名（ifcfg-eth0:1）写成什么都无所谓，但是文件内的DEVICE=eth0:1一定要按照这样的格式写，否则你启动不起来网卡。

3. 查看网卡连接状态

```
[root@www ~]# mii-tool
eth0: negotiated 100baseTx-FD flow-control, link ok
```

图片 15.26 15_54.png.jpg

mii-tool这个命令用来查看网卡是否连接，如图显示link ok等字样说明连接正常，否则会显示'no link'字样，下图是笔者所管理的一台服务器，eth1网卡没有连接。

```
eth0: negotiated 100baseTx-FD, link ok
eth1: no link
```

图片 15.27 15_55.png.jpg

如果你的机器是虚拟机，那么你使用该命令时应该显示成如下：

```
[root@localhost ~]# mii-tool
SIOCGMIIPHY on 'eth0' failed: Operation not supported
no MII interfaces found
```

图片 15.28 15_56.png.jpg

这是因为使用的是虚拟网卡，不支持这个工具查看。不用多关注此，你记住这个mii-tool命令即可，它可是会经常用到的。

4. 更改主机名

当装完系统后，默认主机名为localhost，使用hostname就可以知道你的linux的主机名是什么。

```
[root@localhost ~]# hostname
localhost
```

图片 15.29 15_57.png.jpg

同样使用hostname可以更改你的主机名。

```
[root@localhost ~]# hostname Aming
[root@localhost ~]# hostname
Aming
```

图片 15.30 15_71.png.jpg

下次登录时就会把命令提示符

```
root@localhost ~
```

图片 15.31 15_72.png.jpg

中的'localhost'更改成'Aming'。不过这样修改只是保存在内存中，下次重启还会变成未改之前的主机名，所以需要你还要去更改相关的配置文件'/etc/sysconfig/network'。

```
[root@localhost ~]# vim /etc/sysconfig/network
NETWORKING=yes
NETWORKING_IPV6=no
HOSTNAME=localhost.localdomain
```

图片 15.32 15_73.png.jpg

把HOSTNAME=localhost.localdomain 修改成你想要的主机名，这样再重启就会读取这个配置文件中的HOSTNAME。

5. 设置DNS

DNS是用来解析域名用的，平时我们访问网站都是直接输入一个网址，而dns把这个网址解析到一个IP。关于dns的概念，如果你很陌生的话，那就去网上查一下吧。在linux下面设置dns非常简单，只要把dns地址写到一个配置文件中即可。这个配置文件就是/etc/resolv.conf

```
[root@localhost ~]# cat /etc/resolv.conf
; generated by /sbin/dhclient-script
nameserver 10.0.9.2
nameserver 202.106.0.20
```

图片 15.33 15_74.png.jpg

resolv.conf有它固有的格式，一定要写成'nameserver IP'的格式，上面那行以';'为开头的行是一行注释，没有实际意义，建议写两个或多个nameserver，默认会用第一个nameserver去解析域名，当第一个解析不到时会使用第二个。在linux下面有一个特殊的文件/etc/hosts也能解析域名，不过是需要我们手动在里面添加IP+域名这些内容，它的作用是临时解析某个域名，非常有用。


```
[root@localhost ~]# vim /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1          localhost.localdomain localhost
::1               localhost6.localdomain6 localhost6
10.0.2.34         www.test.com.cn
```

图片 15.34 15_75.png.jpg

它的格式如上图，每一行作为一条记录，分成两部分，第一部分是IP，第二部分是域名。关于hosts文件，有几点需要你注意：

- 1) 一个IP后面可以跟多个域名，可以是几十个甚至上百个；
- 2) 每行只能有一个IP，也就是说一个域名不能对应多个IP；
- 3) 如果有多行中出现相同的域名（前面IP不一样），会按最前面出现的记录来解析。

【linux的防火墙】

1. selinux

Selinux是Redhat/CentOS系统特有的安全机制。不过因为这个东西限制太多，配置也特别繁琐所以几乎没有人去真正应用它。所以装完系统，我们一般都要把selinux关闭，以免引起不必要的麻烦。关闭selinux的方法为：

```
[root@localhost ~]# vim /etc/selinux/config
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - SELinux is fully disabled.
SELINUX=enforcing
# SELINUXTYPE= type of policy in use. Possible values are:
#     targeted - Only targeted network daemons are protected.
#     strict - Full SELinux protection.
SELINUXTYPE=targeted
```

图片 15.35 15_76.png.jpg

把 'SELINUX=enforcing' 改成 'SELINUX=disabled'，然后重启机器。临时关闭selinux的命令为

```
[root@localhost ~]# getenforce
Enforcing
[root@localhost ~]# setenforce 0
[root@localhost ~]# getenforce
Permissive
```

图片 15.36 15_77.png.jpg

getenforce命令可以得到selinux的状态，其中有两种（Enforcing|Permissive），前者表示开放，后者表示关闭，但是会发出警告。setenforce用来设置selinux的状态，后面跟0则设置成Permissive后面跟1设置成Enforci

ng。关闭selinux的命令为setenforce 0，但是这只是临时关闭，重启后恢复，想要永久生效，请更改配置文件/etc/selinux/config。

2. iptables

Iptables是linux上特有的防火墙机制，其功能非常强大，然而笔者在日常的管理工作中仅仅用到了一两个应用，这并不代表iptables不重要。作为一个网络管理员，iptables是必要要熟练掌握的。但是作为系统管理员，我们也应该会最基本的iptables操作，认识iptables的基本规则。

1) iptables的三个表

filter：这个表主要用于过滤包的，是系统预设的表，这个表也是笔者用的最多的。内建三个链INPUT、OUTPUT以及FORWARD。INPUT作用于进入本机的包；OUTPUT作用于本机送出的包；FORWARD作用于那些跟本机无关的包。

nat：主要用处是网络地址转换，也有三个链。PREROUTING 链的作用是在包刚刚到达防火墙时改变它的目的地址，如果需要的话。OUTPUT链改变本地产生的包的目的地址。POSTROUTING链在包就要离开防火墙之前改变其源地址。该表笔者用的不多，但有时候会用到。

mangle：这个表主要是用于给数据包打标记，然后根据标记去操作哪些包。这个表几乎不怎么用。除非你想成为一个高级网络工程师，否则你就没有必要花费很多心思在它上面。

2) iptables 基本语法

A. 查看规则以及清除规则

```
[root@localhost ~]# iptables -t nat -nVL
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target      prot opt in      out     source
Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target      prot opt in      out     source
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target      prot opt in      out     source
[root@localhost ~]# iptables -t filter -nVL
Chain INPUT (policy ACCEPT 1078 packets, 99620 bytes)
  pkts bytes target      prot opt in      out     source
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target      prot opt in      out     source
Chain OUTPUT (policy ACCEPT 575 packets, 73850 bytes)
  pkts bytes target      prot opt in      out     source
```

图片 15.37 15_78.png.jpg

如上图，-t 后面跟表名，-nV 即查看该表的规则，其中-n表示不针对IP反解析主机名；-L表示列出的意思；而-v表示列出的信息更加详细。如果不加-t，则打印filter表的相关信息。

```
[root@localhost ~]# iptables -nV
Chain INPUT (policy ACCEPT 1150 packets, 106K bytes)
  pkts bytes target    prot opt in     out     source destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target    prot opt in     out     source destination
Chain OUTPUT (policy ACCEPT 603 packets, 76954 bytes)
  pkts bytes target    prot opt in     out     source destination
```

图片 15.38 15_79.png.jpg

这个和-t filter 打印的信息是一样的。

关于清除规则的命令中，笔者用的最多就是

```
[root@localhost ~]# iptables -F
[root@localhost ~]# iptables -Z
```

图片 15.39 15_80.png.jpg

不加-t默认是针对表filter来操作的，-F 表示把所有规则全部删除；-Z表示把包以及流量计数器置零（这个笔者认为很有用）。

B. 增加/删除一条规则

```
iptables -A INPUT -s 10.0.2.35 -p tcp --sport 1234 -d 10.0.2.68 --dport 80 -j DROP
```

图片 15.40 15_81.png.jpg

这就是增加了一条规则，省略-t所以针对的是filter表。-A 表示增加一条规则，另外还有-I 表示插入一条规则，-D 删除一条规则；后面的INPUT即链名称，还可以是OUTPUT或者FORWARD；-s 后跟源地址；-p 协议（tcp, udp, icmp）；--sport/--dport 后跟源端口/目标端口；-d 后跟目的IP（主要针对内网或者外网）；-j 后跟动作（DROP即把包丢掉，REJECT即包拒绝；ACCEPT即允许包）。这样讲可能很乱，那笔者多举几个例子来帮你理解：

```
[root@localhost ~]# iptables -I INPUT -s 10.0.2.36 -j DROP
```

图片 15.41 15_82.png.jpg

上例表示：插入一条规则，把来自10.0.2.36的所有数据包丢掉。

```
[root@localhost ~]# iptables -D INPUT -s 10.0.2.36 -j DROP
```

图片 15.42 15_83.png.jpg

删除刚刚插入的规则。注意要删除一条规则时，必须和插入的规则一致，也就是说，两条iptables命令，除了-I和-D不一样外，其他地方都一样。

```
iptables -A INPUT -s 10.0.2.36 -p tcp --dport 80 -j DROP
```

图片 15.43 15_165.png.jpg

上例表示把来自10.0.2.36 并且是tcp协议到本机的80端口的数据包丢掉。这里要说的是，--dport/--sport 必须要和-p选项一起使用，否则会出错。

```
iptables -I OUTPUT -p tcp --dport 22 -d 10.0.2.34 -j DROP
```

图片 15.44 15_166.png.jpg

把发送到10.0.2.34的22端口的数据包丢掉。下面做一个小试验：

```
[root@localhost ~]# ping 10.0.2.34 -c 1
PING 10.0.2.34 (10.0.2.34) 56(84) bytes of data.
64 bytes from 10.0.2.34: icmp_seq=1 ttl=63 time=1.13 ms

--- 10.0.2.34 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.134/1.134/1.134/0.000 ms
[root@localhost ~]# iptables -I OUTPUT -p icmp -d 10.0.2.34 -j DROP
[root@localhost ~]# ping 10.0.2.34 -c 4
PING 10.0.2.34 (10.0.2.34) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted

--- 10.0.2.34 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3004ms
```

图片 15.45 15_167.png.jpg

一开始用本机ping 10.0.2.34是通的，然后使用iptables增加一条规则，使到10.0.2.34的icmp包丢掉，再ping 10.0.2.34则不通了。此时用' iptables -nvL' 查看iptalbes规则。

```
Chain OUTPUT (policy ACCEPT 1438 packets, 162K bytes)
pkts bytes target      prot opt in      out     source    destination
  4   336 DROP        icmp -- *       *        0.0.0.0/0 10.0.2.34
```

图片 15.46 15_168.png.jpg

会有一条这样的记录，看pkts那列显示4个数据包，因为我们ping 的时候给10.0.2.34发送了4个数据包，第二列表示这4个数据包一共有多大（336bytes）。此时使用' iptables -Z' 清零。

```
[root@localhost ~]# iptables -Z
[root@localhost ~]# iptables -nvL |grep icmp
  0     0 DROP        icmp -- *       *        0.0.0.0/0 10.0.2.34
```

图片 15.47 15_169.png.jpg

现在你明白 'iptables -Z' 的意义了吧。至于FORWORD链的应用笔者几乎没有用到过，所以不再举例。再总结一下各个选项的作用：

- A/-D：增加删除一条规则；
- I：插入一条规则，其实跟-A的效果一样；
- p：指定协议，可以是tcp, udp或者icmp；
- dport：跟-p一起使用，指定目标端口；
- sport：跟-p一起使用，指定源端口；
- s：指定源IP（可以是一个ip段）；
- d：指定目的IP（可以是一个ip段）；
- j：后跟动作，其中ACCEPT表示允许包，DROP表示丢掉包，REJECT表示拒绝包；
- i：指定网卡（不常用，但有时候能用到）；

```
[root@localhost ~]# iptables -A INPUT -s 10.0.2.0/24 -i eth0 -j ACCEPT
[root@localhost ~]# iptables -nvL |grep '2.0/24'
```

66	6107	ACCEPT	all	--	eth0	*	10.0.2.0/24	0.0.0.0/0
----	------	--------	-----	----	------	---	-------------	-----------

图片 15.48 15_170.png.jpg

上例中表示，把来自10.0.2.0/24这个网段的并且作用在eth0上的包放行。有时候你的服务器上iptables太多了，想删除某一条规则时，又不容易掌握当时创建时的规则。其实有一种比较简单的方法：

```
[root@localhost ~]# iptables -nvL --line-numbers
```

图片 15.49 15_171.png.jpg

查看结果如下：

```
Chain OUTPUT (policy ACCEPT 3315 packets, 354K bytes)
num  pkts bytes target    prot opt in     out     source          destination
1      0     0 ACCEPT    icmp -- *      *      0.0.0.0/0      10.0.2.34
```

图片 15.50 15_172.png.jpg

删除某一个规则的方法是：

```
[root@localhost ~]# iptables -D OUTPUT 1
```

图片 15.51 15_173.png.jpg

-D 后跟链名，然后是规则num，这个num就是查看iptables规则时第一列的值。

iptables还有一个选项经常用到，-P（大写）选项，表示预设策略。用法如下：

```
[root@localhost ~]# iptables -P INPUT DROP
```

图片 15.52 15_174.png.jpg

-P后面跟链名，策略内容或者为DROP或者为ACCEPT，默认是ACCEPT。注意：如果你在连接远程服务器，千万不要随便敲这个命令，因为一旦你敲完回车你就会断掉。

```
root@localhost ~]# iptables -nVL
Chain INPUT (policy DROP 100 packets, 17790 bytes)
 pkts bytes target      prot opt in      out     source      destination
```

图片 15.53 15_175.png.jpg

看到上图中红框标出的部分了吧，现在所有进来的数据包全部DROP了。这个策略一旦设定后，只能使用iptables -P ACCEPT才能恢复成原始状态，而不能使用-F参数。下面笔者针对一个小需求讲述一下这个iptables规则如何设定。

需求：只针对filter表，预设策略INPUT链DROP，其他两个链ACCEPT，然后针对10.0.2.0/24开通22端口，对所有网段开放80端口，对所有网段开放21端口。

这个需求不算复杂，但是因为有多条规则，所以最好写成脚本的形式。

```
vim /usr/local/sbin/iptables.sh
#! /bin/bash

# Aming 2011-05-25.
ipt="/sbin/iptables"
$ipt -F
$ipt -P INPUT DROP
$ipt -P OUTPUT ACCEPT
$ipt -P FORWARD ACCEPT
$ipt -A INPUT -s 10.0.2.0/24 -p tcp --dport 22 -j ACCEPT
$ipt -A INPUT -p tcp --dport 80 -j ACCEPT
$ipt -A INPUT -p tcp --dport 21 -j ACCEPT
```

图片 15.54 15_176.png.jpg

完成脚本的编写后，直接运行 ‘sh /usr/local/sbin/iptables.sh’ 即可。如果想开机启动时初始化防火墙规则，则需要在/etc/rc.d/rc.local 中添加一行 ‘sh /usr/local/sbin/iptables.sh’ 。

关于icmp的包有一个比较常见的应用。

```
[root@localhost ~]# iptables -I INPUT -p icmp --icmp-type 8 -j DROP
```

图片 15.55 15_177.png.jpg

--icmp-type 这个选项是要跟-p icmp 一起使用的，后面指定类型编号。这个8指的是能在本机ping通其他机器，而其他机器不能ping通本机。这个有必要记一下。

C. nat表的应用

其实，linux的iptables功能是十分强大的，笔者曾经的一个老师这样形容linux的网络功能：只有想不到没有做不到！也就是说只要你能够想到的关于网络的应用，linux都能帮你实现。在日常生活中相信你接触过路由器吧，它的功能就是分享上网。本来一根网线过来（其实只有一个公网IP），通过路由器后，路由器分配了一个网段（私网IP），这样连接路由器的多台pc都能连接internet而远端的设备认为你的IP就是那个连接路由器的公网IP。这个路由器的功能其实就是由linux的iptables实现的，而iptables又是通过nat表作用而实现的这个功能。

至于具体的原理以及过程，笔者不想阐述，请查看相关资料。笔者在这里只举一个例子来说明iptables如何实现这个功能。假设你的机器上有两块网卡eth0和eth1，其中eth0的IP为10.0.2.68，eth1的IP为192.168.1.1。eth0连接了internet但eth1没有连接，现在有另一台机器（192.168.1.2）和eth1是互通的，那么如何设置也能够让连接eth1的这台机器能够连接internet（即能和10.0.2.68互通）？

```
echo "1" >/proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth0 -j MASQUERADE
```

图片 15.56 15_178.png.jpg

其实就是这样简单的两个命令就能实现上面的需求。第一个命令涉及到了内核参数相关的配置文件，它的目的是为了打开路由转发功能，否则无法实现我们的应用。第二个命令则是iptables对nat表做了一个IP转发的操作，-o选项后跟设备名，表示出口的网卡，MASQUERADE表示伪装的意思。

关于nat表，笔者不想讲太多内容，你只要学会这个路由转发即可。其他的交给网络工程师去学习吧，毕竟你将来可是要做linux系统工程师的。

D. 保存以及备份iptables规则

也许你不知道，咱们设定的防火墙规则只是保存在内存中，并没有保存到某一个文件中，也就是说当系统重启后以前设定的规则就没有了，所以设定好规则后要先保存一下。

```
[root@localhost ~]# service iptables save
Saving firewall rules to /etc/sysconfig/iptables: [ OK ]
```

图片 15.57 15_179.png.jpg

它会提示你把规则保存在了/etc/sysconfig/iptables文件内。其实，这个文件就是iptables的配置文件了，你不妨查看一下它。

```
[root@localhost ~]# cat /etc/sysconfig/iptables
# Generated by iptables-save v1.3.5 on Wed May 25 13:06:50 2011
*nat
:PREROUTING ACCEPT [141:16042]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A POSTROUTING -s 192.168.1.0/255.255.255.0 -o eth0 -j MASQUERADE
COMMIT
# Completed on Wed May 25 13:06:50 2011
# Generated by iptables-save v1.3.5 on Wed May 25 13:06:50 2011
*filter
:INPUT ACCEPT [3293:259646]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [660:90056]
COMMIT
# Completed on Wed May 25 13:06:50 2011
```

图片 15.58 15_180.png.jpg

红线部分就是咱们刚才设定那条规则！有时可能因为我们设置防火墙规则有误导致服务器出问题，这时候不妨先备份一下这个配置文件，然后停止防火墙服务。

```
[root@localhost ~]# service iptables stop
```

图片 15.59 15_181.png.jpg

这样防火墙就失效了，但是一旦你重新设定规则后（哪怕只有一条），防火墙又开始工作了。

```
[root@localhost ~]# sh /usr/local/sbin/iptables.sh
[root@localhost ~]# iptables-save >myipt.rule
[root@localhost ~]# cat myipt.rule
# Generated by iptables-save v1.3.5 on Wed May 25 13:12:12 2011
*filter
:INPUT DROP [4:463]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [30:2792]
-A INPUT -s 10.0.2.0/255.255.255.0 -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 21 -j ACCEPT
COMMIT
# Completed on Wed May 25 13:12:12 2011
```

图片 15.60 15_182.png.jpg

我还可以使用 `iptables-save >filename` 这条命令来保存一个防火墙规则，这样就可以起到备份的作用了。要想恢复这个规则使用下面这个命令即可。

```
[root@localhost ~]# iptables-restore <myipt.rule
```

图片 15.61 15_183.png.jpg

【 linux系统的任务计划 】

这部分内容太重要了，其实大部分系统管理工作都是通过定期自动执行某一个脚本来完成的，那么如何定期执行某一个脚本呢？这就要借助linux的cron功能了。

```
usage: crontab [-u user] file
       crontab [-u user] [ -e | -l | -r ]
           (default operation is replace, per 1003.2)
       -e      (edit user's crontab)
       -l      (list user's crontab)
       -r      (delete user's crontab)
       -i      (prompt before deleting user's crontab)
       -s      (selinux context)
```

图片 15.62 15_184.png.jpg

关于cron任务计划功能的操作都是通过crontab这个命令来完成的。其中常用的选项有：

-u：指定某个用户，不加-u选项则为当前用户；

-e：制定计划任务；

-l：列出计划任务；

-r：删除计划任务。

```
[root@localhost ~]# crontab -e
no crontab for root - using an empty one
01 10 26 05 4 echo "ok" >/root/cron.log
```

图片 15.63 15_185.png.jpg

使用crontab -e 来制定计划任务，上面的例子表示在05月26日（这天必须是周四）的10点01分执行’ echo "ok" >/root/cron.log’ 这样的任务。

Cron的格式是这样的，每一行代表一个任务计划，总共分成两部分，前面部分为时间，后面部分要执行的命令。后面的命令不用多讲，至于前面的时间是有讲究的，这个时间共分为5段，用空格隔开（可以是多个空格），第一段表示分钟(0-59)，第二段表示小时(0-23)，第三段表示日(1-31)，第四段表示月(1-12)，第五段表示周(0-7,0或者7都可以表示为周日)。从左至右依次是：分，时，日，月，周（一定要牢记）！

crontab -e 实际上是打开了/var/spool/cron/username（如果是root则打开的是/var/spool/cron/root）这个文件。使用的是vim编辑器，所以要保存的话则在命令模式下输入:wq即可。但是，你千万不要直接去编辑那个文件，因为可能会出错，所以一定要使用crontab -e来编辑。查看已经设定的任务计划使用crontab -l

```
[root@localhost ~]# crontab -l
01 10 26 05 4 echo "ok" >/root/cron.log
```

图片 15.64 15_186.png.jpg

删除计划任务要用crontab -r

```
[root@localhost ~]# crontab -r
[root@localhost ~]# crontab -l
no crontab for root
```

图片 15.65 15_187.png.jpg

下面笔者给你出一些练习题，帮助你熟悉这个cron的应用。

1. 每天凌晨1点20分清除/var/log/slow.log这个文件；
2. 每周日3点执行' /bin/sh /usr/local/sbin/backup.sh' ；
3. 每月14号4点10分执行' /bin/sh /usr/local/sbin/backup_month.sh' ；
4. 每隔8小时执行' ntpdate time.windows.com' ；
5. 每天的1点，12点，18点执行' /bin/sh /usr/local/sbin/test.sh' ；
6. 每天的9点到18点执行' /bin/sh /usr/local/sbin/test2.sh' ；

答案：

1. 20 1 * * * echo " " >/var/log/slow.log
2. 0 30 * * 0 /bin/sh /usr/local/sbin/backup.sh
3. 10 04 14 * * /bin/sh /usr/local/sbin/backup_month.sh
4. 0 */8 * * * ntpdate time.windows.com
5. 0 1,12,18 * * /bin/sh /usr/local/sbin/test.sh
6. 0 9-18 * * * /bin/sh /usr/local/sbin/test2.sh

Cron的这部分内容并不难，你只要会了这6道练习题，你就算掌握它了。这里要简单说一下，每隔8小时，就是用全部小时（0-23）去除以8，你仔细想一下结果，其实算出来应该是0,8,16三个数。当遇到多个数（分钟、小时、月、周）例如第5题，则需要用逗号隔开。而时间段是可以用' - '的方式表示的。等设置好了所有的计划任务后需要查看一下crond服务是否启动，如果没有启动，需要启动它。

```
[root@localhost ~]# service crond status
crond (pid 1665) is running...
```

图片 15.66 15_188.png.jpg

如何启动稍后会做介绍。除了用户自定义的计划任务外，其实系统本身也有计划任务的。

```
[root@localhost ~]# cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

图片 15.67 15_189.png.jpg

系统会安装这个配置文件中的计划去执行内定的任务。

【linux的系统服务管理】

如果你对windows非常熟悉的话，相信你肯定配置过开机启动的服务，有些服务我们日常用不到则要把它停掉，一来可以节省资源，二来可以减少安全隐患。在linux上同样也有相关的工具来管理系统的服务。

1. ntsysv

用来配置哪些服务开启或者关闭，有点想图形界面，不过是使用键盘来控制的。如果没有这个命令请使用 yum install -y ntsysv 安装它。



图片 15.68 15_190.png.jpg

敲完这个命令后则显示出如上图中的画面。在屏幕的最上面有‘Red Hat’等字样，这是在告诉我们这个工具是由Red Hat公司开发的。按键盘的上下方向键可以调节红色光标，按空格可以选择开启或者不开启，如果前面的中括号内显示有‘*’则表示开启否则不开启。通过这个工具也可以看到目前系统中所有的服务。建议除‘cron d, iptables, network, sshd, syslog, irqbalance, sendmail, microcode_ctl’外其他服务全部停掉。选择好后，按‘tab’键选择ok然后回车。需要重启机器才能生效。

2. chkconfig

Linux系统所有的预设服务可以查看/etc/init.d/目录得到

```
[root@localhost ~]# ls /etc/init.d/
avahi-daemon  haldaemon  lisa        netfs       single
avahi-dnscfg  halt       lm_sensors  netplugd    sshd
cron          httpd      lvm2-monitor network     syslog
cups          iptables  mcstrans    pcsd        sysstat
exim          iptables  messagebus  rawdevices  tcsd
functions     killall   multipathd  rdisc       winbind
gpm           kudzu     netconsole  restorecond xfs
```

图片 15.69 15_191.png.jpg

其实这就是系统所有的预设服务了。为什么这样讲，因为系统预设服务都是可以通过这样的命令实现‘service 服务名 start|stop|restart’，这里的 service 名就是/etc/init.d/目录下的这些文件了。除了可以使用‘service cron d start’启动cron d外，还可以使用/etc/init.d/cron d start 来启动。

```
[root@localhost ~]# /etc/init.d/cron d restart
Stopping cron d: [ OK ]
Starting cron d: [ OK ]
[root@localhost ~]# service cron d restart
Stopping cron d: [ OK ]
Starting cron d: [ OK ]
```

图片 15.70 15_192.png.jpg

如上图，这两个命令出来的结果是一样的。

```
[root@localhost ~]# chkconfig --list
avahi-daemon    0:off  1:off  2:off  3:off  4:on   5:on   6:off
avahi-dnscfg    0:off  1:off  2:off  3:off  4:off  5:off  6:off
crond           0:off  1:off  2:on   3:on   4:on   5:on   6:off
cups            0:off  1:off  2:on   3:off  4:on   5:on   6:off
exim            0:off  1:off  2:on   3:off  4:on   5:on   6:off
gpm             0:off  1:off  2:on   3:off  4:on   5:on   6:off
haldaemon       0:off  1:off  2:off  3:off  4:on   5:on   6:off
httpd           0:off  1:off  2:off  3:off  4:off  5:off  6:off
ip6tables       0:off  1:off  2:on   3:off  4:on   5:on   6:off
iptables        0:off  1:off  2:on   3:on   4:on   5:on   6:off
kudzu           0:off  1:off  2:off  3:off  4:on   5:on   6:off
lisa            0:off  1:off  2:off  3:off  4:off  5:off  6:off
lm_sensors      0:off  1:off  2:on   3:off  4:on   5:on   6:off
lvm2-monitor    0:off  1:on   2:on   3:off  4:on   5:on   6:off
mcstrans        0:off  1:off  2:on   3:off  4:on   5:on   6:off
messagebus      0:off  1:off  2:off  3:off  4:on   5:on   6:off
multipathd      0:off  1:off  2:off  3:off  4:off  5:off  6:off
```

图片 15.71 15_193.png.jpg

再看看这个chkconfig命令，它不仅可以列出来所有的服务，还可以详细到每个级别。这里的级别（0,1,2,3,4,5,6）就是inittab里面介绍的那几个启动级别了。

```
[root@localhost ~]# chkconfig --list |grep crond
crond           0:off  1:off  2:on   3:on   4:on   5:on   6:off
```

图片 15.72 15_194.png.jpg

这样还可以查看某一个服务的启动情况。

```
[root@localhost ~]# chkconfig --level 3 crond off
[root@localhost ~]# chkconfig --list |grep crond
crond           0:off  1:off  2:on   3:off  4:on   5:on   6:off
```

图片 15.73 15_195.png.jpg

用--level 指定级别，后面是服务名，然后是off或者on，--level后还可以跟多个级别。

```
[root@localhost ~]# chkconfig --level 345 crond off
[root@localhost ~]# chkconfig --list |grep crond
crond           0:off  1:off  2:on   3:off  4:off  5:off  6:off
```

图片 15.74 15_196.png.jpg

另外还可以省略级别，默认是针对2,3,4,5级别操作。

```
[root@localhost ~]# chkconfig crond on
[root@localhost ~]# chkconfig --list |grep crond
crond           0:off  1:off  2:on   3:on   4:on   5:on   6:off
```

图片 15.75 15_197.png.jpg

另外这个chkconfig 还有一个功能就是可以把某个服务加入到系统服务，即可以使用service 服务名 start 这样的形式，并且可以在chkconfig --list 中查找到。当然也能删除掉。

```
[root@localhost ~]# chkconfig --del crond
[root@localhost ~]# chkconfig --list |grep crond
[root@localhost ~]# chkconfig --add crond
[root@localhost ~]# chkconfig --list |grep crond
crond          0:off  1:off  2:on   3:on   4:on   5:on   6:off
```

图片 15.76 15_198.png.jpg

这个功能常用在把自定义的启动脚本加入到系统服务当中。关于系统服务就讲这些内容，其实还有很多内容笔者没有介绍，道理很简单，一来讲多了你不能消化二来讲多了你也用不上。

【linux中的数据备份】

数据备份，不用说太多吧，毫无疑问很重要。笔者就曾经有过一次非常痛苦的经历，备份策略没有做好，结果磁盘坏掉数据丢失，简直是撕心裂肺的痛呀。还好数据重要性不是特别高，即使是不高也是丢失了数据，这是作为系统管理员最不应该出现的事故。所以，在你以后的系统维护工作中，一定要把数据备份当回事，认真对待。在linux上作为数据备份的工具很多，但笔者就只用一种那就是rsync 从字面上的意思你可以理解为remote sync（远程同步）这样可以让你理解的更深刻一些。Rsync不仅可以远程同步数据（类似于scp），当然还可以本地同步数据（类似于cp），但不同于cp或scp的一点是，rsync不像cp/scp一样会覆盖以前的数据（如果数据已经存在），它会先判断已经存在的数据和新数据有什么不同，只有不同时才会把不同的部分覆盖掉。如果你的linux上下面看例子吧。（如果没有rsync命令请使用yum install -y rsync安装）

```
[root@localhost ~]# rsync -arv 123 /tmp/123
building file list ... done
123/

sent 52 bytes  received 26 bytes  156.00 bytes/sec
total size is 0  speedup is 0.00
```

图片 15.77 15_199.png.jpg

上面例子表示把当前目录下的123同步到/tmp/目录下，并且同样也命名为123。如果是远程拷贝的话就是这样的形式了 IP:path（如：10.0.2.34:/root/）

```
[root@localhost ~]# rsync -arv 123 10.0.2.34:/root/
```

图片 15.78 15_200.png.jpg

当建立连接后，是需要输入密码的。如果手动去执行这些操作还好，但是如果是写在脚本中怎么办？这就涉及到添加信任关系了，该部分内容稍后会详细介绍。

1. rsync的命令格式

```
rsync [OPTION]... SRC DEST
```

```
rsync [OPTION]... SRC [USER@]HOST:DEST
```

```
rsync [OPTION]... [USER@]HOST:SRC DEST
```

```
rsync [OPTION]... [USER@]HOST::SRC DEST
```

```
rsync [OPTION]... SRC [USER@]HOST::DEST
```

笔者在一开始举的两个例子，第一个例子即为第一种格式，第二个例子即为第二种格式，但不同的是，笔者并没有加user@host 如果不加默认指的是root。第三种格式是从远程目录同步数据到本地。第四种以及第五种格式使用了两个冒号，这种方式和前面的方式的不同在于验证方式不同，稍后详细介绍。

2. rsync常用选项

-a：归档模式，表示以递归方式传输文件，并保持所有属性，等同于-rlptgoD，-a选项后面可以跟一个--no-OPTION 这个表示关闭-rlptgoD中的某一个例如 -a --no-l 等同于-rlptgoD

-r：对子目录以递归模式处理，主要是针对目录来说的，如果单独传一个文件不需要加-r，但是传输的是目录必须加-r选项

-v：打印一些信息出来，比如速率，文件数量等

-l：保留软链接

-L：向对待常规文件一样处理软链接，如果是SRC中有软连接文件，则加上该选项后将会把软连接指向的目标文件拷贝到DST

-p：保持文件权限

-o：保持文件属主信息

-g：保持文件属组信息

-D：保持设备文件信息

-t：保持文件时间信息

--delete：删除那些DST中SRC没有的文件

--exclude=PATTERN：指定排除不需要传输的文件，等号后面跟文件名，可以是万用字符模式（如*.txt）

-u：加上这个选项后将会把DST中比SRC还新的文件排除掉，不会覆盖

下面笔者将会针对这些选项做一些列小实验：

1) 建立目录以及文件

```
[root@localhost ~]# mkdir rsync
[root@localhost ~]# cd rsync
[root@localhost rsync]# mkdir test1
[root@localhost rsync]# cd test1
[root@localhost test1]# touch 1 2 3
[root@localhost test1]# ln -s /root/1.sh 1.sh
[root@localhost test1]# ls -l
total 16
-rw-r--r-- 1 root root  0 May 27 03:07 1
lrwxrwxrwx 1 root root 10 May 27 03:08 1.sh -> /root/1.sh
-rw-r--r-- 1 root root  0 May 27 03:07 2
-rw-r--r-- 1 root root  0 May 27 03:07 3
[root@localhost test1]# cd ..
```

图片 15.79 15_201.png.jpg

笔者建立这些文件的目的是为做试验做一些准备工作。

2) 使用-a选项

```
[root@localhost rsync]# rsync -a test1 test2
[root@localhost rsync]# ls test2
test1
[root@localhost rsync]# ls test2/test1
1 1.sh 2 3
```

图片 15.80 15_202.png.jpg

这里有一个问题，就是本来想把test1目录直接拷贝成test2目录，可结果rsync却新建了test2目录然后把test1放到test2当中。为了避免这样的情况发生，可以这样做：

```
[root@localhost rsync]# rm -rf test2
[root@localhost rsync]# rsync -a test1/ test2/
[root@localhost rsync]# ls test2
1 1.sh 2 3
```

图片 15.81 15_203.png.jpg

加一个斜杠就好了，所以笔者建议你在使用rsync备份目录时要养成加斜杠的习惯。在上面讲了-a选项等同于-rlptgoD，而且-a还可以和--no-OPTION一并使用。

```
[root@localhost rsync]# rm -rf test2
[root@localhost rsync]# rsync -av --no-r test1/ test2/
skipping directory test1/.
created directory test2

sent 29 bytes  received 20 bytes  98.00 bytes/sec
total size is 0  speedup is 0.00
[root@localhost rsync]# ls test2
```

图片 15.82 15_204.png.jpg

笔者加上-v选项来获得更多的信息，上例中因为没有使用-r选项导致只能拷贝目录但不能拷贝目录下面的内容（英文翻译过来就是“忽略了目录test1/.”，其中test1/.指的就是test1目录内部的所有文件），所以虽然创建了test2目录，但是test2目录为空。下面再看看那个-l选项的作用。

```
[root@localhost rsync]# rm -rf test2
[root@localhost rsync]# rsync -av --no-l test1/ test2
building file list ... done
created directory test2
./
1
skipping non-regular file "1.sh"
2
3
sent 231 bytes  received 92 bytes  646.00 bytes/sec
total size is 10  speedup is 0.03
```

图片 15.83 15_205.png.jpg

使用-v选项看来就是方便呀，上例告诉我们跳过了非普通文件1.sh，其实1.sh是一个软连接文件，如果不使用-l选项则不会理会软连接文件的。

```
[root@localhost rsync]# ls test2
1 2 3
```

图片 15.84 15_206.png.jpg

果真test2目录当中没有那个1.sh的影子。当然加上-l选项则会把软连接文件给拷贝过去，但是软连接的目标文件却没有拷贝过去，有时候咱们指向拷贝软连接文件所指向的目标文件，那这时候该怎么办呢？

3) 使用-L选项

```
[root@localhost rsync]# rm -rf test2
[root@localhost rsync]# rsync -avL test1/ test2/
building file list ... done
created directory test2
./
1
1.sh
2
3
sent 346 bytes  received 114 bytes  920.00 bytes/sec
total size is 77  speedup is 0.17
[root@localhost rsync]# ls -l test2/
total 20
-rw-r--r-- 1 root root  0 May 27 03:07 1
-rw-r--r-- 1 root root 77 May 20 15:01 1.sh
-rw-r--r-- 1 root root  0 May 27 03:07 2
-rw-r--r-- 1 root root  0 May 27 03:07 3
```

图片 15.85 15_207.png.jpg

一个-L就可以把SRC中软连接的目标文件给拷贝到DST

4) 使用-u选项

首先查看一下test1/1 和 test2/1的访问时间（肯定是一样的），然后使用touch修改一下test2/1的访问时间（此时test2/1要比test1/1的访问时间晚了一些），如果不加-u选项的话，会把test2/1的访问时间变成和test1/1的访问时间一样。这样讲也许你会迷糊，不妨看一看。

```
[root@localhost rsync]# ll test1/1 test2/1
-rw-r--r-- 1 root root 0 May 27 03:07 test1/1
-rw-r--r-- 1 root root 0 May 27 03:07 test2/1
[root@localhost rsync]# touch test2/1
[root@localhost rsync]# ll test1/1 test2/1
-rw-r--r-- 1 root root 0 May 27 03:07 test1/1
-rw-r--r-- 1 root root 0 May 27 03:30 test2/1
[root@localhost rsync]# rsync -avL test1/ test2/
building file list ... done
./
1

sent 139 bytes  received 48 bytes  374.00 bytes/sec
total size is 77  speedup is 0.41
[root@localhost rsync]# ll test1/1 test2/1
-rw-r--r-- 1 root root 0 May 27 03:07 test1/1
-rw-r--r-- 1 root root 0 May 27 03:07 test2/1
```

图片 15.86 15_208.png.jpg

看到了吧，本来test2/1的访问时间已经不同于test1/1的访问时间了，但是同步后访问时间又一致了。

```
[root@localhost rsync]# touch test2/1
[root@localhost rsync]# ll test1/1 test2/1
-rw-r--r-- 1 root root 0 May 27 03:07 test1/1
-rw-r--r-- 1 root root 0 May 27 03:33 test2/1
[root@localhost rsync]# rsync -avLu test1/ test2/
building file list ... done

sent 91 bytes  received 20 bytes  222.00 bytes/sec
total size is 77  speedup is 0.69
[root@localhost rsync]# ll test1/1 test2/1
-rw-r--r-- 1 root root 0 May 27 03:07 test1/1
-rw-r--r-- 1 root root 0 May 27 03:33 test2/1
```

图片 15.87 15_209.png.jpg

现在你明白-u选项的妙用了吧。

5) 使用--delete选项

```

[root@localhost rsync]# rm -f test1/1
[root@localhost rsync]# ls test1
1.sh 2 3
[root@localhost rsync]# rsync -avL test1/ test2/
building file list ... done
./

sent 86 bytes  received 26 bytes  224.00 bytes/sec
total size is 77  speedup is 0.69
[root@localhost rsync]# ls test2/
1 1.sh 2 3
[root@localhost rsync]# rsync -avL --delete test1/ test2/
building file list ... done
deleting 1
./

sent 86 bytes  received 26 bytes  224.00 bytes/sec
total size is 77  speedup is 0.69
[root@localhost rsync]# ls test2/
1.sh 2 3

```

图片 15.88 15_210.png.jpg

如果不使用--delete选项当SRC有文件删除时，DST是不会删除的，只有加上--delete选项后才能删除掉。还有一种情况就是如果在DST增加文件了，而SRC当中没有这些文件，同步时加上--delete选项后同样会删除新增的文件。

```

[root@localhost rsync]# touch test2/4
[root@localhost rsync]# ls test1
1.sh 2 3
[root@localhost rsync]# rsync -avL --delete test1/ test2/
building file list ... done
deleting 4
./

sent 86 bytes  received 26 bytes  224.00 bytes/sec
total size is 77  speedup is 0.69

```

图片 15.89 15_211.png.jpg

6) 使用--exclude 选项

```

[root@localhost rsync]# touch test1/4
[root@localhost rsync]# rsync -avL --exclude=4 test1/ test2/
building file list ... done
./

sent 86 bytes  received 26 bytes  224.00 bytes/sec
total size is 77  speedup is 0.69
[root@localhost rsync]# ls test1
1.sh 2 3 4
[root@localhost rsync]# ls test2
1.sh 2 3

```

图片 15.90 15_212.png.jpg

另外还可以使用万用字符*匹配

```
[root@localhost rsync]# touch test1/1.txt test1/2.txt
[root@localhost rsync]# ls test1
1.sh 1.txt 2 2.txt 3 4
[root@localhost rsync]# rsync -avL --exclude=*.txt test1/ test2/
building file list ... done
./
4

sent 139 bytes received 48 bytes 374.00 bytes/sec
total size is 77 speedup is 0.41
[root@localhost rsync]# ls test2
1.sh 2 3 4
```

图片 15.91 15_213.png.jpg

最后简单总结一下，平时你使用rsync同步数据的时候，使用-a选项基本上就可以达到我们想要的效果了，只是有时候会有个别的需求，会用到-a --no-OPTION, -u, -L, --delete, --exclude这些选项，但是笔者要求你把上面这些全部掌握，毕竟这才几个而已，大部分选项笔者都没有介绍。如果在以后的工作中遇到特殊需求了，就去查一下rsync的man文档吧。

3. rsync 应用实例

1) 通过ssh的方式

最上面介绍的5种方式当中，第二、第三（1个冒号）就属于通过ssh的方式，这种方式其实就是让用户去登录到远程机器，然后执行rsync的任务。

```
[root@Aming-1 ~]# rsync -avL rsync/test1/ root@10.0.2.69:/root/test2/
root@10.0.2.69's password:
building file list ... done
created directory /root/test2
./
1.sh
1.txt
2
2.txt
3
4

sent 460 bytes received 158 bytes 247.20 bytes/sec
total size is 77 speedup is 0.12
```

图片 15.92 15_214.png.jpg

这种方式就是前面介绍的第二种方式了，是通过ssh拷贝的数据，是要输入10.0.2.69那台机器root的密码的。

```
[root@Aming-1 ~]# rsync -avL root@10.0.2.69:/root/test2/ rsync/test3/
root@10.0.2.69's password:
receiving file list ... done
created directory rsync/test3
./
1.sh
1.txt
2
2.txt
3
4

sent 158 bytes  received 464 bytes  248.80 bytes/sec
total size is 77  speedup is 0.12
```

图片 15.93 15_215.png.jpg

这个则为第三种方式。这两种方式如果写到脚本里，备份起来就有麻烦了，因为要输入密码，脚本本来就是自动的，不可能做到的。但是不代表没有解决办法。那就是通过密钥验证，密钥不设立密码就ok了。还记得在前面笔者曾经介绍过通过密钥登录远程主机吗，下面要讲的内容就是那些东西了。

先提前说一下基本的主机信息：10.0.2.68（主机名Aming-1）和10.0.2.69（主机名Aming）需要从Aming-1上拷贝数据到Aming上。

A. 首先确认一下Aming-1上是否有这个文件 /root/.ssh/id_rsa.pub

```
[root@Aming-1 ~]# ls /root/.ssh/id_rsa.pub
/root/.ssh/id_rsa.pub
```

图片 15.94 15_216.png.jpg

如果没有安装以下的方法生成：

```
[root@Aming-1 ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
c5:0a:6c:73:d6:ba:ba:3f:68:b0:58:92:61:25:34:70 root@Aming-1
```

图片 15.95 15_217.png.jpg

在这个过程中会有一些交互的过程，因为笔者的/root/.ssh/id_rsa已经存在，所以会问是否覆盖，笔者选择覆盖，然后会提示要输入这个密钥的密码，出于安全考虑应该定义个密码，但是我们的目的就是为自动化同步数据，所以这里不输入任何密码，直接按回车，即密码为空。最后则生成了私钥(/root/.ssh/id_rsa)和公钥文件(/root/.ssh/id_rsa.pub)

B. 把公钥文件的内容拷贝到目标机器上

```
[root@Aming-1 ~]# cat /root/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA2wexw7eosZzvHCHGu9E4u1ae02PTJjS1bvQwFDr8nCL/6AI5
0W9xk0yI74ADHch1B21ctHcrk6Giu5kytS/JwcxwCrMxmcLekh/aH0hRa0fkMANWaVSU+U2k6jDNbnryYXET
p3di7FnJIECYAekIsLzwnRXCL12i/0hBD0y1qNVPzj/ERL3J5bC25IK/ZtXfdi20zptEdDzDYQmFngruNCCN
0gsD6rGpod3A+bj0HBNBGaKgm+h3NOXXrn3/vEiIgDZB6IQnWNIaRYybHNp6UbQAej/LUGpdX6QnhEZRJrnA
qAXU1gIUXq9r5vIFBJwjE3Y5eEQUA6C19bQPpXNbyw== root@Aming-1
```

图片 15.96 15_218.png.jpg

复制主机Aming-1的/root/.ssh/id_rsa.pub文件内容，并粘贴到主机Aming的/root/.ssh/authorized_keys中

```
[root@Aming ~]# vim /root/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA2wexw7eosZzvHCHGu9E4u1ae02PTJjS1bvQwFDr8nCL/
6AI50W9xk0yI74ADHch1B21ctHcrk6Giu5kytS/JwcxwCrMxmcLekh/aH0hRa0fkMANWaVSU+U2k6jDN
bnryYXETp3di7FnJIECYAekIsLzwnRXCL12i/0hBD0y1qNVPzj/ERL3J5bC25IK/ZtXfdi20zptEdDzD
YQmFngruNCCN0gsD6rGpod3A+bj0HBNBGaKgm+h3NOXXrn3/vEiIgDZB6IQnWNIaRYybHNp6UbQAej/L
UGpdX6QnhEZRJrnAqAXU1gIUXq9r5vIFBJwjE3Y5eEQUA6C19bQPpXNbyw== root@Aming-1
```

图片 15.97 15_219.png.jpg

在这一步也许你会遇到/root/.ssh目录不存在的问题，可以手动创建，并修改目录权限为700也可以执行ssh-keygen命令生成这个目录。保存/root/.ssh/authorized_keys文件后，再到主机Aming-1上执行

```
[root@Aming-1 ~]# ssh 10.0.2.69
Last login: Fri May 27 04:14:38 2011 from www.test.com.cn
[root@Aming ~]#
```

图片 15.98 15_220.png.jpg

你会发现，现在不用输入密码也可以登录主机Aming了。下面再从主机Aming-1上执行一下rsync命令试试吧。

```
[root@Aming-1 ~]# rsync -avL rsync/test1/ root@10.0.2.69:/root/test2/
building file list ... done

sent 121 bytes received 20 bytes 282.00 bytes/sec
total size is 77 speedup is 0.55
```

图片 15.99 15_221.png.jpg

2) 通过后台服务的方式

这种方式可以理解成这样，在远程主机上建立一个rsync的服务器，在服务器上配置好rsync的各种应用，然后本机作为rsync的一个客户端去连接远程的rsync服务器。下面笔者就介绍一下，如何去配置一台rsync服务器。

A. 建立并配置rsync的配置文件 /etc/rsyncd.conf

```
[root@Aming-1 ~]# vim /etc/rsyncd.conf
#port=873
log file=/var/log/rsyncd.log
pid file=/var/run/rsyncd.pid
#address=10.0.2.68

[test]
path=/root/rsync
use chroot=false
max connections=4
read only=false
list=true
uid=root
gid=root
auth users=test
secrets file=/etc/rsyncd.passwd
hosts allow=10.0.2.0/24
```

图片 15.100 15_222.png.jpg

其中配置文件分为两部分全部配置部分和模块配置部分，全局部分就是几个参数而已，就像笔者的rsyncd.conf中port, log file, pid file, address这些都属于全局配置，而[test] 以下部分就是模块配置部分了。一个配置文件中可以有多个模块，模块名自定义，格式就像笔者的rsyncd.conf中的这样。其实模块中的一些参数例如use chroot, max connections, udi, gid, auth users, secrets file以及hosts allow都可以配置成全局的参数。当然笔者给出的参数并不是所有的，你可以通过man rsyncd.conf 获得更多信息。

下面就简单解释一下这些参数的意义：

port：指定在哪个端口启动rsyncd服务，默认是873

log file：指定日志文件

pid file：指定pid文件，这个文件的作用涉及到服务的启动以及停止等进程管理操作

address：指定启动rsyncd服务的IP，假如你的机器有多个IP，就可以指定其中一个启动rsyncd服务，默认是在全部IP上启动

[test]：指定模块名，自定义

path：指定数据存放的路径

use chroot: true|false 默认是true，意思是在传输文件以前首先chroot到path参数所指定的目录下。这样做的原因是实现额外的安全防护，但是缺点是需要以roots权限，并且不能备份指向外部的符号连接所指向的目录文件。默认情况下chroot值为true，如果你的数据当中有软连接文件的话建议设置成false。

max connections：指定最大的连接数，默认是0即没有限制

read only: ture|false 如果为true则不能上传到该模块指定的路径下

list：指定当用户查询该服务器上的可用模块时，该模块是否被列出，设定为true则列出，false则隐藏

uid/gid：指定传输文件时，以哪个用户/组的身份传输

auth users：指定传输时要使用的用户名

secrets file：指定密码文件，该参数连同上面的参数如果不指定则不使用密码验证

hosts allow：指定被允许连接该模块的主机，可以是IP或者网段，如果是多个，之间用空格隔开

B. 编辑secrets file，保存后要赋予600权限

```
[root@Aming-1 ~]# cat /etc/rsyncd.passwd
test:test123
[root@Aming-1 ~]# chmod 600 /etc/rsyncd.passwd
```

图片 15.101 15_223.png.jpg

C. 启动rsyncd服务

```
[root@Aming-1 ~]# rsync --daemon --config=/etc/rsyncd.conf
[root@Aming-1 ~]# cat /var/log/rsyncd.log
2011/05/27 06:25:34 [2062] rsyncd version 2.6.8 starting, listening on port 873
[root@Aming-1 ~]#
```

图片 15.102 15_224.png.jpg

启动后查看日志，看看是否有错误信息，然后再看下端口是否启动。

```
[root@Aming-1 ~]# netstat -ltn |grep 873
tcp        0      0 0.0.0.0:873          0.0.0.0:*
tcp        0      0 :::873              :::*
```

图片 15.103 15_225.png.jpg

如果想开机启动，请把”rsync - daemon - config=/etc/rsyncd.conf” 写入到/etc/rc.d/rc.local文件。

D. 到另一台机器上测试

```
[root@Aming ~]# rsync -avL test@10.0.2.68::test/test1/ /root/test4/
Password:
receiving file list ... done
created directory /root/test4
./
1.sh
1.txt
2
2.txt
3
4

sent 243 bytes  received 525 bytes  170.67 bytes/sec
total size is 77  speedup is 0.10
```


图片 15.104 15_226.png.jpg

记得那个use chroot参数吗，如果设置为true，则/root/test4/1.sh不会被拷贝过来。

```
[root@Aming-1 ~]# sed -i 's/chroot=false/chroot=true/g' /etc/rsyncd.conf
[root@Aming-1 ~]# grep chroot /etc/rsyncd.conf
use chroot=true
```

图片 15.105 15_227.png.jpg

修改rsyncd.conf文件，把use chroot改成true，不用重启服务即可生效。

```
[root@Aming ~]# rm -rf /root/test4
[root@Aming ~]# rsync -avL test@10.0.2.68::test/test1/ /root/test4/
Password:
receiving file list ... symlink has no referent: "/test1/1.sh" (in test)
done
created directory /root/test4
./
1.txt
2
2.txt
3
4

sent 221 bytes  received 441 bytes  147.11 bytes/sec
total size is 0  speedup is 0.00
rsync error: some files could not be transferred (code 23) at main.c(129)
[rator=2.6.8]
```

图片 15.106 15_228.png.jpg

从上例中的详细信息中也可以看到，不能拷贝软连接的。所以请记住，如果涉及到软连接，请设置use chroot=false。另外这种方式也是可以不用手动输入密码的，两种实现方式。

第一种：指定密码文件

```
[root@Aming ~]# vim /etc/rsyncd.passwd
[root@Aming ~]# cat /etc/rsyncd.passwd
test123
[root@Aming ~]# chmod 600 /etc/rsyncd.passwd
```

图片 15.107 15_229.png.jpg

先编辑一个密码文件，并修改600权限。

```
[root@Aming ~]# rsync -avL test@10.0.2.68::test/test1/ /root/test4/ --password-f
ile=/etc/rsyncd.passwd
receiving file list ... symlink has no referent: "/test1/1.sh" (in test)
done

sent 105 bytes  received 225 bytes  660.00 bytes/sec
total size is 0  speedup is 0.00
rsync error: some files could not be transferred (code 23) at main.c(1298) [gene
rator=2.6.8]
```

图片 15.108 15_230.png.jpg

在同步时, 指定密码文件即可(--password-file=/etc/rsyncd.passwd)

第二种: 在rsync服务器端不指定用户

```
[root@Aming-1 ~]# sed -i 's/auth user/#auth user/' /etc/rsyncd.conf
[root@Aming-1 ~]# sed -i 's/secrets/#secrets/' /etc/rsyncd.conf
[root@Aming-1 ~]# egrep 'auth user|secrets' /etc/rsyncd.conf
#auth users=test
#secrets file=/etc/rsyncd.passwd
```

图片 15.109 15_231.png.jpg

把相关的参数都用 '#' 注释掉。然后再到客户端上测试。

```
[root@Aming ~]# rm -rf test*
[root@Aming ~]# rsync -avL 10.0.2.68::test/test1/ /root/test4/
receiving file list ... symlink has no referent: "/test1/1.sh" (in test)
done
created directory /root/test4
./
1.txt
2
2.txt
3
4

sent 193 bytes  received 400 bytes  1186.00 bytes/sec
total size is 0  speedup is 0.00
rsync error: some files could not be transferred (code 23) at main.c(129)
rator=2.6.8]
[root@Aming ~]# ls test4
1.txt 2 2.txt 3 4
```

图片 15.110 15_232.png.jpg

注意, 这里不用再加test@这个用户了, 默认是以root的身份拷贝的, 现在已经不需要输入密码了。

【linux系统日志】

日志重要吗? 必须的, 没有日志你怎么知道你的系统状况? 没有日志你如何排查一个trouble? 日志记录了系统每天发生的各种各样的事情, 你可以通过他来检查错误发生的原因, 或者受到攻击时攻击者留下的痕迹。日志主要的功能有: 审计和监测。他还可以实时的监测系统状态, 监测和追踪侵入者等等。

笔者常查看的日志文件为/var/log/message. 它是核心系统日志文件, 包含了系统启动时的引导消息, 以及系统运行时的其他状态消息。IO 错误、网络错误和其他系统错误都会记录到这个文件中。另外其他信息, 比如某个人的身份切换为 root以及用户自定义安装的软件 (apache) 的日志也会在这里列出。通常, /var/log/messages 是在做故障诊断时首先要查看的文件。那你肯定会说了, 这么多日志都记录到这个文件中, 那如果服务器上有很多服务岂不是这个文件很快就会写的很大, 没有错, 但是系统有一个日志轮询的机制, 每星期切换一个日志, 变

成message.1, message.2, …messages.4 连同message一共有5个这样的日志文件。这是通过logrotate工具的控制来实现的，它的配置文件是/etc/logrotate.conf. 如果没有特殊需求请不要修改这个配置文件。

```
[root@5d6d-web-hh-1fctc ~]# vim /etc/logrotate.conf
## 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

# 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 -- we'll rotate them here
/var/log/wtmp {
    monthly
    create 0664 root utmp
    rotate 1
}
```

图片 15.111 15_233.png.jpg

/var/log/message是由syslogd这个守护进程产生的，如果停掉这个服务则系统不会产生/var/log/message，所以这个服务不要停。Syslogd服务的配置文件为/etc/syslog.conf这个文件定义了日志的级别，具体详细的东西笔者不再阐述，因为若没有特殊需求是不需要修改这个配置文件的，请使用” man syslog.conf” 获得更多关于它的信息。

```
[root@localhost ~]# cat /etc/syslog.conf
# 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

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

# Everybody gets emergency messages
*.emerg                                     *

# 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
```

图片 15.112 15_234.png.jpg

除了关注/var/log/message外，你还应该多关注一下'dmesg'这个命令，它可以显示系统的启动信息，如果你的某个硬件有问题（比如说网卡）用这个命令也是可以看到的。

```
[root@localhost ~]# dmesg |less
Linux version 2.6.18-164.el5 (mockbuild@builder16.centos.org) (gcc version 4.1.2
20080704 (Red Hat 4.1.2-46)) #1 SMP Thu Sep 3 03:33:56 EDT 2009
BIOS-provided physical RAM map:
 BIOS-e820: 0000000000010000 - 000000000009f800 (usable)
 BIOS-e820: 000000000009f800 - 00000000000a0000 (reserved)
 BIOS-e820: 00000000000dc000 - 0000000000100000 (reserved)
 BIOS-e820: 0000000000100000 - 0000000000aaf0000 (usable)
 BIOS-e820: 0000000000aaf0000 - 0000000000aaff000 (ACPI data)
 BIOS-e820: 0000000000aaff000 - 0000000000ab00000 (ACPI NVS)
 BIOS-e820: 0000000000ab00000 - 0000000000ac00000 (usable)
 BIOS-e820: 0000000000fec0000 - 0000000000fec10000 (reserved)
 BIOS-e820: 0000000000fee0000 - 0000000000fee01000 (reserved)
 BIOS-e820: 0000000000ffe0000 - 00000000100000000 (reserved)
0MB HIGHMEM available.
172MB LOWMEM available.
found SMP MP-table at 000f6cd0
```

图片 15.113 15_235.png.jpg

这一小节就介绍这么多，在结束之前，笔者给你一个小小的建议。以后在你日常的管理工总中要养成多看日志的习惯，尤其是一些应用软件的日志，比如apache, mysql, php等常用的软件，看它们的日志（错误日志）可以帮助你排查问题以及监控它们的运行状况是否良好。

【xargs与-exec】

1. xargs

在前面的例子中笔者曾经使用过这个命令，你是否有印象呢？现在就详细介绍一下它，平时笔者使用xargs还是比较多的，很方便。

```
[root@localhost ~]# man xargs
XARGS(1) XARG
S(1)
NAME
    xargs - build and execute command lines from standard input
SYNOPSIS
    xargs [-0prt看] [-E eof-str] [-e[eof-str]] [--eof[=eof-str]]
    [--null] [-d delimiter] [--delimiter delimiter] [-I
    replace-str] [-i[replace-str]] [--replace[=replace-str]]
    [-l[max-lines]] [-L max-lines] [--max-lines[=max-lines]]
    [-n max-args] [--max-args=max-args] [-s max-chars]
    [--max-chars=max-chars] [-P max-procs] [--max-procs=max-
    procs] [--interactive] [--verbose] [--exit]
    [--no-run-if-empty] [--arg-file=file] [--version] [--help]
    [command [initial-arguments]]
```

图片 15.114 15_236.png.jpg

查看xargs的man文档，解释是这样的：build and execute command lines from standard input. 至于翻译成中文理解着有点困难。不妨笔者举个例子说明它的作用。

```
[root@localhost ~]# echo "12121212">123.txt
[root@localhost ~]# ls 123.txt |xargs cat
12121212
```

图片 15.115 15_237.png.jpg

它的作用就是把管道符前面的输出作为xargs 后面的命令的输入。它的好处在于可以把本来两步或者多步才能完成的任务简单一步就能完成。xargs常常和find命令一起使用，比如，查找当前目录创建时间大于10天的文件，然后再删除。

```
[root@localhost ~]# find ./ -mtime +10 |xargs rm
```

图片 15.116 15_238.png.jpg

这种应用是最为常见的，xargs后面的rm 也可以更选项，当是目录时，就需要-r选项了。在笔者看来xargs的这个功能不叫什么，它的另一个功能才叫神奇。现在我有一个这样的需求，查找当前目录下所有.txt的文件，然后把这些.txt的文件变成.txt_bak 。正常情况下，我们不得不写脚本去实现，但是使用xargs就一步。

```
[root@localhost ~]# mkdir test
[root@localhost ~]# cd test
[root@localhost test]# touch 1.txt 2.txt 3.txt
[root@localhost test]# ls
1.txt 2.txt 3.txt
[root@localhost test]# ls *.txt|xargs -n1 -i{} mv {} {}_bak
[root@localhost test]# ls
1.txt_bak 2.txt_bak 3.txt_bak
```

图片 15.117 15_239.png.jpg

xargs -n1 -i{} 类似for循环，-n1意思是一个一个对象的去处理，-i{} 把前面的对象使用{}取代，mv {} {}_bak 相当于 mv 1.txt 1.txt_bak。你刚开始接触这个命令时也许有点难以理解，多练习一下你就会熟悉了，笔者建议你记住这个应用，很实用。

2. -exec

使用find命令时，经常使用一个选项就是这个-exec了，可以达到和xargs同样的效果。比如，查找当前目录创建时间大于10天的文件并删除。

```
[root@localhost ~]# find ./ -mtime +10 -exec rm -rf {} \;
```

图片 15.118 15_240.png.jpg

这个命令中也是把{}作为前面find出来的文件的替代符，后面的”\”为” ;” 的脱意符，不然shell会把分号作为该行命令的结尾。这个-exec有时候也挺实用的。

```
[root@localhost ~]# cd test
[root@localhost test]# ls
1.txt_bak 2.txt_bak 3.txt_bak
[root@localhost test]# find ./ *txt_bak -exec mv {} {}_bak \;
[root@localhost test]# ls
1.txt_bak_bak 2.txt_bak_bak 3.txt_bak_bak
```

图片 15.119 15_241.png.jpg

用-exec同样可以实现前面那个复杂的需求。

【screen工具介绍】

有时候，也许你会有这样的需求，需要执行一个命令或者脚本，但是需要几个小时甚至几天。这就要考虑一个问题，就是中途断网或出现其他意外情况，执行的任务中断了怎么办？你可以把命令或者脚本丢到后台运行，不过也不保险。笔者下面就介绍两种方法来避免这样的问题发生。

1. 使用nohup

```
[root@localhost ~]# cat /usr/local/sbin/test.sh
#!/bin/bash
sleep 1000
[root@localhost ~]# nohup sh /usr/local/sbin/test.sh &
[1] 2036
[root@localhost ~]# nohup: appending output to `nohup.out'
```

图片 15.120 15_242.png.jpg

直接加一个' &' 虽然丢到后台了, 但是当退出该终端时很有可能这个脚本也会退出的, 而在前面加上' nohup p' 就没有问题了。nohup的作用就是不挂断地运行命令。

2. screen工具的使用

简单来说, screen是一个可以在多个进程之间多路复用一个物理终端的窗口管理器。screen中有会话的概念, 用户可以在一个screen会话中创建多个screen窗口, 在每一个screen窗口中就像操作一个真实的SSH连接窗口那样。下面笔者介绍screen的一个简单应用。

1) 打开一个会话, 直接输入screen命令然后回车, 进入screen会话窗口。如果你没有screen命令, 请用' yum install -y screen' 安装。

```
[root@localhost ~]# screen
[root@localhost ~]# █
```

图片 15.121 15_243.png.jpg

2) screen -ls 查看已经打开的screen会话

```
[root@localhost ~]# screen -ls
There is a screen on:
      2082.pts-0.localhost      (Attached)
1 Socket in /var/run/screen/S-root.
```

图片 15.122 15_244.png.jpg

3) Ctrl +a 再按d退出该screen会话, 只是退出, 并没有结束。结束的话输入Ctrl +d 或者输入exit

4) 退出后还想再次登录某个screen会话, 使用screen -r [screen 编号], 这个编号就是上图中那个2082. 当只有一个screen会话时, 后面的编号是可以省略的。

当你有某个需要长时间运行的命令或者脚本时就打开一个screen会话, 然后运行该任务。按ctrl +a 再按d退出会话, 不影响终端窗口上的任何操作。

【linux下同步时间服务器】

时间的准确性在服务器上非常重要，所以要与标准时间保持同步，毕竟服务器的时钟并不一定精准，所以需要我们每隔一段时间去同步一下时间。笔者所管理的服务器每隔6小时就会同步一下时间。如何同步呢？这就要用到ntpdate 这个指令。如果你的服务器上没有这个指令，请使用'yum install -y ntpdate'安装，或者下载源码包安装。

同步时间的命令为：'ntpdate timeserver' 这里的timeserver为时间服务器的IP或者hostname，常用的timeserver有210.72.145.44, time.windows.com(windows的时间服务器)。如果你想每隔6小时同步一次那么请指定一个计划任务。

```
00 */6 * * * /usr/sbin/ntpdate 210.72.145.44 >/dev/null
```

之所以在后面加一个重定向，是因为这个同步的过程是有内容输出的，因为我们的计划任务是在后台执行的，输出的内容会以邮件的形式发送给用户，所以为了避免邮件太多请输出到/dev/null（在linux下这个设备是虚拟的存在，你可以理解为空洞）



T



16

LAMP环境搭建



经过前部分章节的学习，你已经掌握了linux的基础知识了。但是想成为一名系统管理员恐怕还有点难度，因为好多单位招聘这个职位的时候都要求有一定的工作经验。然而真正的经验一天两天是学不来的，是靠长时间积累得来的。不过你也不要灰心，所谓的工作经验无非也就是是一些运行在linux系统上的软件的配置以及应用。就好像是装在windows上的office一样，大部分人都会装，但是十分会用的却不多。是因为office太难吗，当然不是，只是因为只有一小部分人花费了很长很长的时间去使用和研究office而已。

LAMP 是Linux Apache MySQL PHP的简写，其实就是把Apache, MySQL以及PHP安装在Linux系统上，组成一个环境来运行php的脚本语言。至于什么是php脚本语言，笔者不再介绍，请自己查资料吧。Apache是最常用的WEB服务软件，而MySQL是比较小型的数据库软件，这两个软件以及PHP都可以安装到windows的机器上。下面笔者就教你如何构建这个LAMP环境。

安装MySQL

一般我们平时安装MySQL都是源码包安装的，但是由于它的编译需要很长的时间，所以，笔者建议你安装二进制免编译包。你可以到MySQL官方网站去下载：<http://www.mysql.com/downloads/> 具体版本根据你的平台和需求而定，目前比较常用的mysql-5.1.x 和mysql-5.3.x下面是安装步骤：

1. 下载mysql到/usr/local/src/

```
cd /usr/local/src/
```

```
wget http://syslab.comsenz.com/downloads/linux/mysql-5.0.86-linux-i686-icc-glibc23.tar.gz
```

1. 解压

```
tar zxvf /usr/local/src/mysql-5.0.86-linux-i686-icc-glibc23.tar.gz
```

1. 把解压完的数据移动到/usr/local/mysql

```
mv mysql-5.0.86-linux-i686-icc-glibc23 /usr/local/mysql
```

1. 建立mysql用户

```
useradd mysql
```

1. 初始化数据库

```
cd /usr/local/mysql
```

```
mkdir /data/mysql ; chown -R mysql:mysql /data/mysql
```

```
./scripts/mysql_install_db --user=mysql --datadir=/data/mysql
```

--user定义数据库的所属主，--datadir定义数据库安装到哪里，建议放到大空间的分区上，这个目录需要自行创建。

1. 拷贝配置文件

```
cp support-files/my-large.cnf /etc/my.cnf
```

1. 拷贝启动脚本文件并修改其属性

```
cp support-files/mysql.server /etc/init.d/mysqld
```

```
chmod 755 /etc/init.d/mysqld
```

1. 修改启动脚本

```
vim /etc/init.d/mysqld
```

需要修改的地方有datadir=/data/mysql（前面初始化数据库时定义的目录）

1. 把启动脚本加入系统服务项，并设定开机启动，启动mysql

```
chkconfig --add mysqld
```

```
chkconfig mysqld on
```

```
service mysqld start
```

如果启动不了，请到/data/mysql/ 下查看错误日志，这个日志通常是主机名.err。关于mysql的配置文件/etc/my.cnf请参考这篇文章 <http://www.92csz.com/19/603.html>

【安装Apache】

```
cd /usr/local/src/
```

```
wget http://syslab.comsenz.com/downloads/linux/httpd-2.2.11.tar.gz
```

```
useradd www（增加 Apache运行账户）
```

```
tar zvxf httpd-2.2.11.tar.bz2
```

```
cd httpd-2.2.11
```

```
./configure --prefix=/usr/local/apache2 --with-included-apr --enable-so --enable-deflate=shared  
--enable-expires=shared --enable-rewrite=shared --enable-static-support --disable-userdir
```

```
make
```

make install

安装PHP

```
wget http://syslab.comsenz.com/downloads/linux/php-5.2.10.tar.gz

tar zvxf php-5.2.10.tar.gz
cd php-5.2.10
./configure --prefix=/usr/local/php \
--with-apxs2=/usr/local/apache2/bin/apxs \
--with-config-file-path=/usr/local/php/etc \
--with-mysql=/usr/local/mysql \
--with-libxml-dir \
--with-gd \
--with-jpeg-dir \
--with-png-dir \
--with-freetype-dir \
--with-iconv-dir \
--with-zlib-dir \
--with-bz2 \
--with-openssl \
--with-mcrypt \
--enable-soap \
--enable-gd-native-ttf \
--enable-ftp \
--enable-mbstring \
--enable-sockets \
--enable-exif \
--disable-ipv6
make && make install
mkdir /usr/local/php/etc
cp php.ini-dist /usr/local/php/etc/php.ini
```

apache结合php

Apache主配置文件为：/usr/local/apache2/conf/httpd.conf



第 16 章 vim /usr/local/apache2/conf/httpd.conf



找到:

```
AddType application/x-gzip .gz .tgz
```

在该行下面添加

```
AddType application/x-httpd-php .php
```

找到:

```
<IfModule dir_module>
  DirectoryIndex index.html
</IfModule>
```

将该行改为

```
<IfModule dir_module>
  DirectoryIndex index.html index.htm index.php
</IfModule>
```

找到:

```
#Include conf/extra/httpd-mpm.conf
#include conf/extra/httpd-info.conf
#include conf/extra/httpd-vhosts.conf
#include conf/extra/httpd-default.conf
```

去掉前面的“#”号，取消注释。

配置apache的进程管理以及虚拟主机

1. 配置Apache进程管理

配置文件为: /usr/local/apache2/conf/extra/httpd-mpm.conf

将配置文件中下面一段修改为如下:

```
<IfModule mpm_prefork_module>
  ServerLimit      2048 新添加
  StartServers     5
  MinSpareServers  5
  MaxSpareServers  10
  MaxClients       1024 默认最大为256，设置为超过256必须增加有ServerLimit
  MaxRequestsPerChild 0
</IfModule>
```

1. 配置Apache虚拟主机

配置文件为：/usr/local/apache2/conf/extra/httpd-vhosts.conf

将配置文件中下面一段修改为如下：

```
<VirtualHost *:80>
# ServerAdmin webmaster@dummy-host.example.com
DocumentRoot "/data/www"
ServerName www.example.com.cn
ErrorLog "|/usr/local/apache2/bin/rotatelog -l /www/logs/error.log-%Y%m%d 86400"
CustomLog "|/usr/local/apache2/bin/rotatelog -l /www/logs/access.log-%Y%m%d 86400" combined
</VirtualHost>
```

说明：

- ServerAdmin 参数后为管理员email
- DocumentRoot 指的是论坛文件存放的目录
- ServerName 是论坛的域名
- ErrorLog 是论坛错误日志 通过管道使用apache自带的rotatelog工具将日志切割为每天一个文件
- CustomLog 是论坛访问日志，同样切割为每天一个文件

配置Apache缺省httpd设置

配置文件为：/usr/local/apache2/conf/extra/httpd-default.conf

将配置文件中下面一段：

将KeepAlive On 改为KeepAlive Off

配置Apache的访问权限

vim /usr/local/apache2/conf/httpd.conf 找到

```
<Directory />
Options FollowSymlinks
AllowOverride None
Order deny,allow
Deny from all
</Directory>
```

改成：

```
<Directory />
Options FollowSymLinks
AllowOverride None
Order deny,allow
Allow form all
</Directory>
```

配置Apache的运行账户

```
vim /usr/local/apache2/conf/httpd.conf
```

找到

```
User daemon Group daemon
```

改成

```
User www Group www
```

配置完上述内容之后，启动Apache：

```
/usr/local/apache2/bin/apachectl start
```

【测试LAMP是否成功】

```
vim /data/www/1.php
```

写入：

```
<?php
phpinfo();
?>
```

保存后，然后在浏览器中输入 `http://你配置的域名/1.php` 看是否能看到php的相关配置信息。

Zend安装

有时，需要在你的LAMP环境中配置ZEND，因为有些php的应用程序比如Discuz! 或者phpwind等是需要用zend来解密的，不装zend会显示乱码。安装步骤为：

```
cd /usr/local/src
```

```
wget http://syslab.comsenz.com/downloads/linux/ZendOptimizer-3.3.3-linux-glibc23-i386.tar.gz
```

```
tar zxvf ZendOptimizer-3.3.3-linux-glibc23-i386.tar.gz
```



```
cd ZendOptimizer-3.3.3-linux-glibc23-i386
```

```
./install.sh
```

根据提示安装。php.ini文件的路径为：/usr/local/php/etc/ 当提示是否重启apache时，选择不重启。



T



LNMP 环境搭建



和LAMP不同的是LNMP中的N指的是是Nginx（类似于Apache的一种web服务软件）其他都一样。目前这种环境应用的也是非常之多。Nginx设计的初衷是提供一种快速高效多并发的web服务软件。在静态页面的处理上Nginx的确胜Apache一筹，然而在动态页面的处理上Nginx并不比Apache有多少优势。但是，目前还是有很多爱好者对Nginx比较热衷，随着Nginx的技术逐渐成熟，它在web服务软件领域的地位越来越高。

MySQL安装

1. 下载mysql到/usr/local/src/

```
cd /usr/local/src/
```

```
wget http://syslab.comsenz.com/downloads/linux/mysql-5.0.86-linux-i686-icc-glibc23.tar.gz
```

1. 解压

```
tar zxvf /usr/local/src/ mysql-5.0.86-linux-i686-icc-glibc23.tar.gz
```

1. 把解压完的数据移动到/usr/local/mysql

```
mv mysql-5.0.86-linux-i686-ii-glibc23 /usr/local/mysql
```

1. 建立mysql用户

```
useradd mysql
```

1. 初始化数据库

```
cd /usr/local/mysql
```

```
mkdir /data/mysql ; chown -R mysql:mysql /data/mysql
```

```
./scripts/mysql_install_db --user=mysql --datadir=/data/mysql
```

--user定义数据库的所属主，--datadir定义数据库安装到哪里，建议放到大空间的分区上，这个目录需要自行创建。

1. 拷贝配置文件

```
cp support-files/my-large.cnf /etc/my.cnf
```

1. 拷贝启动脚本文件并修改其属性

```
cp support-files/mysql.server /etc/init.d/mysql
```

```
chmod 755 /etc/init.d/mysqld
```

1. 修改启动脚本

```
vim /etc/init.d/mysqld
```

需要修改的地方有datadir=/data/mysql（前面初始化数据库时定义的目录）

1. 把启动脚本加入系统服务项，并设定开机启动，启动mysql

```
chkconfig --add mysqld
```

```
chkconfig mysqld on
```

```
service mysqld start
```

如果启动不了，请到/data/mysql/ 下查看错误日志，该日志格式为主机名.err。

php的安装

这里要先声明一下，针对Nginx的php安装和针对apache的php安装是有区别的，因为Nginx中的php是以fastcgi的方式结合nginx的，可以理解为nginx代理了php的fastcgi，而apache是把php作为自己的模块来调用的。

```
useradd www
```

```
cd /usr/local/src/
```

```
wget http://syslab.comsenz.com/downloads/linux/php-5.2.10.tar.gz
```

```
wget http://syslab.comsenz.com/downloads/linux/php-5.2.10-fpm-0.5.13.diff.gz
```

下载的第二个包php-5.2.10-fpm-0.5.13.diff.gz是用来给php打补丁的，默认情况下，php是无法编译出fastcgi的。

```
tar zxvf php-5.2.10.tar.gz
```

```
gzip -cd php-5.2.10-fpm-0.5.13.diff.gz | patch -d php-5.2.10 -p1
```

```
cd php-5.2.10
```

```
./configure --prefix=/usr/local/php --with-config-file-path=/usr/local/php/etc --with-mysql=/usr/local/mysql --with-mys
```

```
make && make install
```

```
mkdir /usr/local/php/etc

cp php.ini-dist /usr/local/php/etc/php.ini

vim /usr/local/php/etc/php-fpm.conf
```

/tmp/php-fcgi.sock 这一行要改成这样，这里这样修改了以后，在配置nginx的时候就需要注意这个路径了。

修改用户和组的名称为" www"

去掉注释，改成这样：

```
Unix user of processes
    <value name="user">www</value>
Unix group of processes
    <value name="group">www</value>

/usr/local/php/sbin/php-fpm start
```

其他关于php的扩展模块安装请参考：

CentOS 5.5下安装mysql5.1.57+php5.2.17(FastCGI)+nginx1.0.1高性能Web服务器

nginx 安装以及配置

1. nginx源码安装

```
cd /usr/local/src/

wget http://syllab.comsenz.com/downloads/linux/nginx-0.9.6.tar.gz

tar zxvf nginx-0.9.6.tar.gz

cd nginx-0.9.6

./configure --prefix=/usr/local/nginx --sbin-path=/usr/local/nginx/sbin/nginx --conf-path=/usr/local/nginx/conf/nginx.conf

make && make install

mkdir /dev/shm/nginx_temp
```

有的nginx版本编译时会因为pcre编译不过去，需要修改一下 --with-pcre=/usr/local/src/pcre-7.8，前提是已经下载了pcre源码包pcre-7.8.tar.gz，并解压到/usr/local/src/pcre-7.8，不需要编译pcre

1. 编写nginx的启动脚本，并加入系统服务

vi /etc/init.d/nginx 写入以下内容:

```
#!/bin/bash
# chkconfig: - 30 21
# description: http service.
# Source Function Library
. /etc/init.d/functions
# Nginx Settings
NGINX_SBIN="/usr/local/nginx/sbin/nginx"
NGINX_CONF="/usr/local/nginx/conf/nginx.conf"
NGINX_PID="/usr/local/nginx/var/nginx.pid"
RETVAL=0
prog="Nginx"
start() {
    echo -n $"Starting $prog: "
    mkdir -p /dev/shm/nginx_temp
    daemon $NGINX_SBIN -c $NGINX_CONF
    RETVAL=$?
    echo
    return $RETVAL
}
stop() {
    echo -n $"Stopping $prog: "
    killproc -p $NGINX_PID $NGINX_SBIN -TERM
    rm -rf /dev/shm/nginx_temp
    RETVAL=$?
    echo
    return $RETVAL
}
reload(){
    echo -n $"Reloading $prog: "
    killproc -p $NGINX_PID $NGINX_SBIN -HUP
    RETVAL=$?
    echo
    return $RETVAL
}
restart(){
    stop
    start
}
configtest(){
    $NGINX_SBIN -c $NGINX_CONF -t
    return 0
}
case "$1" in
```

```

start)
    start
    ;;
stop)
    stop
    ;;
reload)
    reload
    ;;
restart)
    restart
    ;;
configtest)
    configtest
    ;;
*)
    echo $"Usage: $0 {start|stop|reload|restart|configtest}"
    RETVAL=1
esac

exit $RETVAL

```

保存后，更改/etc/init.d/nginx的权限

```

chmod 755 /etc/init.d/nginx

chkconfig --add nginx

chkconfig nginx on

```

1. nginx的配置

vim /usr/local/nginx/conf/nginx.conf

把原来的文件清空，然后粘贴如下内容：

```

user www www;

worker_processes 2;

error_log /usr/local/nginx/logs/nginx_error.log crit;

pid /usr/local/nginx/var/nginx.pid;

#Specifies the value for maximum file descriptors that can be opened by this process.

```

```
worker_rlimit_nofile 51200;

events
{
    use epoll;

    worker_connections 6000;
}

http
{
    include mime.types;

    default_type application/octet-stream;

    server_names_hash_bucket_size 2048;

    server_names_hash_max_size 4096;

    log_format combined_realip '$remote_addr $http_x_forwarded_for [$time_local] '
'$host "$request_uri" $status '
'"$http_referer" "$http_user_agent"';

    sendfile on;

    tcp_nopush on;

    keepalive_timeout 30;

    client_header_timeout 3m;

    client_body_timeout 3m;

    send_timeout 3m;

    connection_pool_size 256;

    client_header_buffer_size 1k;
```



```
large_client_header_buffers 8 4k;

request_pool_size 4k;

output_buffers 4 32k;

postpone_output 1460;

client_max_body_size 10m;

client_body_buffer_size 256k;

client_body_temp_path /usr/local/nginx/client_body_temp;

proxy_temp_path /usr/local/nginx/proxy_temp;

fastcgi_temp_path /usr/local/nginx/fastcgi_temp;

fastcgi_intercept_errors on;

tcp_nodelay on;

gzip on;

gzip_min_length 1k;

gzip_buffers 4 8k;

gzip_comp_level 5;

gzip_http_version 1.1;

gzip_types text/plain application/x-javascript text/css text/html application/xml;

server

{

listen 80;

server_name www.example.com;

index index.html index.htm index.php;
```

```

root /data/www;

location ~ \.php$ {

include fastcgi_params;

fastcgi_pass unix:/php-fcgi.sock;

fastcgi_index index.php;

fastcgi_param SCRIPT_FILENAME /data/www$fastcgi_script_name;

}

}

```

保存后就可以启动nginx了，在重启之前最好先检查一下是否有问题

`/usr/local/nginx/sbin/nginx -t` 如果显示 "syntax is ok 和 nginx.conf was tested successfully"这样的信息，就说明配置没有问题了，否则就需要根据提示修改了。 `service nginx start`

如果启动不了，请到 `/usr/local/nginx/logs/` 目录下查看 `nginx_error.log` 这个日志文件。若是没有这个日志文件，很有可能是那个目录没有写权限，请执行

```
chmod +w /usr/local/nginx/logs/
```

```
service nginx restart
```

【测试是否解析php文件】

```
vim /data/www/1.php
```

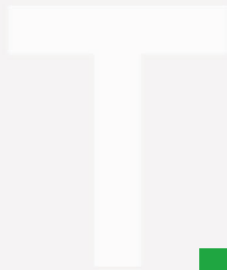
写入如下内容：

```

<?php
phpinfo();
?>

```

然后设定hosts文件，访问 `www.92csz.com/1.php` 看是否能解析出这个页面。



18

学会使用简单的 MySQL 操作



在前面两个章节中已经介绍过MySQL的安装了，但是光会安装还不够，还需要会一些基本的相关操作。当然了，关于MySQL的内容也是非常多的，只不过对于linux系统管理员来讲，一些基本的操作已经可以应付日常的管理工作了，至于更高深的那是DBA（专门管理数据库的技术人员）的事情了。

更改mysql数据库root的密码

首次进入数据库是不用密码的

```
/usr/local/mysql/bin/mysql -u root

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 2

Server version: 5.0.86 MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

现在已经进入到mysql的操作界面了。退出的话，直接输入exit即可。

```
mysql> exit

Bye
```

先解释一下上面的命令的含义，-u 用来指定要登录的用户，root用户是mysql自带的管理员账户，默认没有密码的，那么如何给root用户设定密码？按如下操作：

```
/usr/local/mysql/bin/mysqladmin -u root password '123456'
```

这样就可以设定root用户的密码了。其中mysqladmin就是用来设置密码的工具，-u 指定用户，password 后跟要定义的密码，密码需要用单引号或者双引号括起来。另外你也许发现了，敲命令时总在前面加/usr/local/mysql/bin/ 这样很累。但是直接打mysql 又不能用，这是因为在系统变量\$PATH中没有/usr/local/mysql/bin/这个目录，所以需要这样操作(如果你的linux可以直接打出mysql这个命令，则不要做这个操作)：

```
vim /etc/profile
```

在最后加入一行：

```
export PATH=$PATH:/usr/local/mysql/bin/
```

保存后运行

```
source /etc/profile
```

设定完密码后，再来运行最开始进入mysql数据库操作界面的命令：

```
mysql -u root
```

```
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)
```

就报错了，这是因为root用户有密码。

```
mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.0.86 MySQL Community Server (GPL)
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>
```

需要加-p选项指定密码，这时就会提示你输入密码了。

当设定密码后，如果要想更改密码如何操作呢？

```
mysqladmin -u root -p password "123456789"
Enter password:
```

输入原来root的密码就可以更改密码了。

连接数据库

刚刚讲过通过使用mysql -u root -p 就可以连接数据库了，但这只是连接的本地的数据库'localhost'，然后有很多时候都是去连接网络中的某一个主机上的mysql。

```
mysql -u user1 -p -P 3306 -h 10.0.2.69
```

其中-P（大写）指定远程主机mysql的绑定端口，默认都是3306；-h指定远程主机的IP

一些基本的MySQL操作命令

1. 查询当前所有的库

```
mysql> show databases;

+-----+
| Database |
+-----+
| information_schema |
| mysql |
| test |
+-----+
```

1. 查询某个库的表

```
mysql> use mysql;

Database changed

mysql> show tables;

+-----+
| Tables_in_mysql |
+-----+

| columns_priv |
| db |
| func |
| help_category |
| help_keyword |
```

```

| help_relation |
| help_topic |
| host |
| proc |
| procs_priv |
| tables_priv |
| time_zone |
| time_zone_leap_second |
| time_zone_name |
| time_zone_transition |
| time_zone_transition_type |
| user |
+-----+

```

1. 查看某个表的字段

```

mysql> desc func; //func 是表名
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| name | char(64) | NO | PRI | | |
| ret | tinyint(1) | NO | | 0 | |
| dl | char(128) | NO | | | |
| type | enum('function','aggregate') | NO | | NULL | |
+-----+-----+-----+-----+-----+

```



```
| mysql |
```

```
+-----+
```

1. 创建一个新库

```
mysql> create database db1;
```

```
Query OK, 1 row affected (0.04 sec)
```

1. 创建一个表

```
mysql> create table t1 ( `id` int(4), `name` char(40));
```

```
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> desc t1;
```

```
+-----+-----+-----+-----+-----+-----+
```

```
| Field | Type | Null | Key | Default | Extra |
```

```
+-----+-----+-----+-----+-----+-----+
```

```
| id | int(4) | YES | | NULL | |
```

```
| name | char(40) | YES | | NULL | |
```

```
+-----+-----+-----+-----+-----+-----+
```

1. 查看当前数据库版本

```
mysql> select version();
```

```
+-----+
```

```
| version() |
```

```
+-----+
```

```
| 5.0.86 |
```

```
+-----+
```

1. 查看当前系统时间

```
mysql> select current_date, current_time;
```

```
+-----+-----+
| current_date | current_time |
+-----+-----+
| 2011-05-31 | 08:52:50 |
+-----+-----+
```

1. 查看当前mysql的状态

```
mysql> show status;
```

```
+-----+-----+
| Variable_name | Value |
+-----+-----+
| Aborted_clients | 0 |
| Aborted_connects | 1 |
| Binlog_cache_disk_use | 0 |
| Binlog_cache_use | 0 |
| Bytes_received | 664 |
| Bytes_sent | 6703 |
```

这个命令打出很多东西，显示你的mysql状态。

1. 查看mysql的参数

```
mysql> show variables;
```

很多参数都是可以在/etc/my.cnf中定义的。

1. 创建一个普通用户并授权

```
mysql> grant all on *.* to user1 identified by '123456';
```

```
Query OK, 0 rows affected (0.01 sec)
```

all 表示所有的权限（读、写、查询、删除等等操作），*.* 前面的 * 表示所有的数据库，后面的 * 表示所有的表，identified by 后面跟密码，用单引号括起来。这里的user1指的是localhost上的user1，如果是给网络上的其他机器上的某个用户授权则这样：

```
mysql> grant all on db1.* to 'user2'@'10.0.2.100' identified by '123456';
```

```
Query OK, 0 rows affected (0.00 sec)
```

用户和主机的IP之间有一个@，另外主机IP那里可以用%替代，表示所有主机。例如：

```
mysql> grant all on db1.* to 'user3'@'%' identified by '123456';
```

```
Query OK, 0 rows affected (0.00 sec)
```

一些常用的sql

1. 查询语句

```
mysql> select count(*) from mysql.user;
```

mysql.user表示mysql库的user表；count(*)表示表中共有多少行。

```
mysql> select * from mysql.db;
```

查询mysql库的db表中的所有数据

```
mysql> select db from mysql.db;
```

查询mysql库db表的db段。

```
mysql> select * from mysql.db where host like '10.0.%';
```

查询mysql库db表host字段like 10.0.% 的行，这里的%表示匹配所有，类似于前面介绍的通配符。

1. 插入一行

```
mysql> insert into db1.t1 values (1, 'abc');
```

```
Query OK, 1 row affected (0.00 sec)
```

t1表在前面已经创建过。

```
mysql> select * from db1.t1;
```

```
+-----+-----+
```

```
| id | name |
```

```
+-----+-----+
```

```
| 1 | abc |
```

```
+-----+-----+
```

1. 更改某一行

```
mysql> update db1.t1 set name='aaa' where id=1;
```

```
Query OK, 1 row affected (0.02 sec)
```

```
Rows matched: 1 Changed: 1 Warnings: 0
```

这样就把原来id为1的那行中的name改成'aaa'

1. 删除表

```
mysql> drop table db1.t1;
```

```
Query OK, 0 rows affected (0.01 sec)
```

1. 删除数据库

```
mysql> drop database db1;
```

```
Query OK, 0 rows affected (0.07 sec)
```

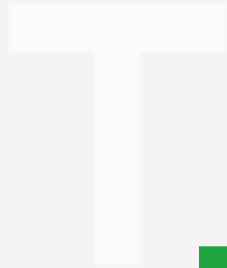
1. 备份与恢复库

```
mysqldump -uroot -p mysql >mysql.sql
```

这里的mysqldump 就是备份的工具了，-p后面的mysql指的是mysql库,把备份的文件重定向到mysql.sql。如果恢复的话，只要：

```
mysql -uroot -p mysql < mysql.sql
```

关于MySQL的基本操作笔者就介绍这么多，当然学会了这些还远远不够，希望你能够在你的工作中学习到更多的知识，如果你对MySQL有很大兴趣，不妨深入研究一下，毕竟多学点总没有坏处。如果想学跟多的东西请去查看MySQL官方中文参考手册（5.1）。



19

NFS 服务配置



【什么是NFS】

NFS会经常用到，用于在网络上共享存储。这样讲，你对NFS可能不太了解，笔者不妨举一个例子来说明一下NFS是用来做什么的。假如有三台机器A、B、C，它们需要访问同一个目录，目录中都是图片，传统的做法是把这些图片分别放到A、B、C。但是使用NFS只需要放到A上，然后A共享给B和C即可。访问的时候，B和C是通过网络的方式去访问A上的那个目录的。

【配置NFS】

NFS配置起来还是蛮简单的，只需要编辑配置文件/etc/exports即可。下面笔者先创建一个简单的NFS服务器。

```
[root@localhost ~]# cat /etc/exports
```

```
/home/ 10.0.2.0/24(rw,sync,all_squash,anonuid=501,anongid=501)
```

这个配置文件就这样简单一行。共分为三部分，第一部分就是本地要共享出去的目录，第二部分为允许访问的主机（可以是一个IP也可以是一个IP段）第三部分就是小括号里面的，为一些权限选项。关于第三部分，笔者简单介绍一下：

rw：读写；

ro：只读；

sync：同步模式，内存中数据时时写入磁盘；

async：不同步，把内存中数据定期写入磁盘中；

no_root_squash：加上这个选项后，root用户就会对共享的目录拥有至高的权限控制，就像是对本机的目录操作一样。不安全，不建议使用；

root_squash：和上面的选项对应，root用户对共享目录的权限不高，只有普通用户的权限，即限制了root；

all_squash：不管使用NFS的用户是谁，他的身份都会被限定成为一个指定的普通用户身份；

anonuid/anongid：要和root_squash 以及 all_squash一同使用，用于指定使用NFS的用户限定后的uid和gid，前提是本机的/etc/passwd中存在这个uid和gid。

介绍了上面的相关的权限选项后，再来分析一下笔者刚刚配置的那个/etc/exports文件。其中要共享的目录为/home，信任的主机为10.0.2.0/24这个网段，权限为读写，同步，限定所有使用者，并且限定的uid和gid都为501。

【使用NFS】

当编辑完配置文件/etc/exports后，就该启动NFS服务了。启动方法为：

```
[root@localhost ~]# service portmap start; service nfs start
```

NFS是依托portmap的，所以首先要启动portmap，然后启动NFS才能是刚才的配置生效。启动完NFS后，就该使用NFS服务了。

```
[root@localhost ~]# showmount -e 127.0.0.1 (用在client上)
```

Export list for 127.0.0.1:

```
/home 10.0.2.0/24
```

用showmount -e 加IP就可以查看NFS的共享情况，上例中，就可以看到127.0.0.1的共享目录为/home，信任主机为10.0.2.0/24这个网段。另外这个showmount 命令还有一个常用的选项就是-a了，它的意思是，把连接本机的NFS的client全部列出。

```
[root@localhost ~]# mount -t nfs 10.0.2.69:/home /mnt (client上)
```

```
[root@localhost ~]# showmount -a (nfs服务器上)
```

All mount points on localhost:

```
10.0.2.69:/home
```

前面的mount 命令为挂载NFS共享目录，相信你能看懂这个格式。showmount -a 命令列出所有的client。

NFS服务中还有一个常用的命令那就是exportfs，它的常用选项为[-aruv]。

-a：全部挂载或者卸载；

-r：重新挂载；

-u：卸载某一个目录；

-v：显示共享的目录；

使用exportfs命令，当改变/etc/exports配置文件后，不用重启nfs服务直接用这个exportfs即可。

```
[root@localhost ~]# cat /etc/exports
```

```
/tmp/ 10.0.2.0/24(rw,sync,no_root_squash)
```

```
[root@localhost ~]# exportfs -arv (nfs服务器上)
```

```
exporting 10.0.2.0/24:/tmp
```

更改目录后，直接exportfs -arv即可生效。

在上面使用到了mount命令来挂载nfs，其实mount这个nfs服务还是有些说法的。首先是用-t nfs 来指定挂载的类型为nfs。另外在使用nfs时，常用一个选项就是nolock了，即在挂载nfs服务时，不加锁。

```
[root@localhost ~]# mount -t nfs -o nolock 10.0.2.69:/tmp /mnt/
```

```
[root@localhost ~]# showmount -a
```

All mount points on localhost:

```
10.0.2.69:/home
```

```
10.0.2.69:/tmp
```

另外我们还可以把要挂载的nfs目录写到client上的/etc/fstab文件中，挂载时只需要mount -a即可。

```
[root@localhost ~]# cat /etc/fstab
```

```
LABEL=/          /              ext3 defaults    1 1
LABEL=/boot      /boot          ext3 defaults    1 2
tmpfs            /dev/shm       tmpfs defaults     0 0
devpts           /dev/pts       devpts gid=5,mode=620 0 0
sysfs            /sys           sysfs defaults     0 0
proc             /proc          proc  defaults     0 0
LABEL=SWAP-hda2  swap           swap  defaults     0 0
10.0.2.69:/tmp   /mnt           nfs  nolock      0 0
```

写完/etc/fstab文件后，只需要mount -a即可挂载nfs服务的共享目录。

```
[root@localhost ~]# umount /mnt/ 首先把刚才挂载的nfs卸载掉
```

```
[root@localhost ~]# mount -a
```

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

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/hda3       7.3G  3.7G  3.3G  53% /
/dev/hda1       99M   12M   83M  12% /boot
```



```
tmpfs          84M  0 84M  0% /dev/shm
```

```
10.0.2.69:/tmp 7.3G 3.7G 3.3G 53% /mnt
```

关于NFS部分就讲这么多，内容并不多，相信你很快就能掌握！



T



20

配置 FTP 服务



【什么是FTP】

也许你对FTP不陌生，但是你是否了解FTP到底是个什么玩意？FTP 是File Transfer Protocol（文件传输协议）的英文简称，而中文简称为“文传协议”。用于Internet上的控制文件的双向传输。同时，它也是一个应用程序（Application）。用户可以通过它把自己的PC机与世界各地所有运行FTP协议的服务器相连，访问服务器上的大量程序和信息。FTP的主要作用，就是让用户连接上一个远程计算机（这些计算机上运行着FTP服务器程序）察看远程计算机有哪些文件，然后把文件从远程计算机上拷到本地计算机，或把本地计算机的文件送到远程计算机去。FTP用的比NFS更多，所以你一定要熟练配置它。

【配置ftp】

安装Redhat/CentOS系统时也许你会连带着把ftp装上，系统默认带的ftp是vsftp，比较常用，配置也很简单。但笔者常使用的ftp软件为pure-ftpd。因为这个软件比vsftp配置起来更加灵活和安全。下面是笔者配置pure-ftpd的过程：

下载最新的pure-ftp源码包pure-ftpd-1.0.21.tar.bz2

```
# wget http://syslab.comsenz.com/downloads/linux/pure-ftpd-1.0.21.tar.bz2
```

```
#tar jxvf pure-ftpd-1.0.21.tar.bz2
```

```
#cd pure-ftpd-1.0.21
```

```
./configure \
```

```
"--prefix=/usr/local/pureftpd" \
```

```
"--without-inetd" \
```

```
"--with-altlog" \
```

```
"--with-puredb" \
```

```
"--with-throttling" \
```

```
"--with-largefile" \
```

```
"--with-peruserlimits" \
```

```
"--with-tls" \
```

```
"--with-language=simplified-chinese"
```

```
#make && make install
```

启动

用配置文件

```
#mkdir /usr/local/pureftpd/etc
#cd configuration-file
#cp pure-ftp.conf /usr/local/pureftpd/etc/pure-ftp.conf
#cp pure-config.pl /usr/local/pureftpd/sbin/pure-config.pl
#chmod 755 /usr/local/pureftpd/sbin/pure-config.pl
```

在启动pure-ftp之前需要先修改配置文件，配置文件为/usr/local/pureftpd/etc/pure-ftp.conf,你可以打开看一下，里面内容很多，如果你英文好，可以好好研究一番，下面是我的配置文件，如果你嫌麻烦，直接拷贝过去即可。

```
ChrootEveryone      yes
BrokenClientsCompatibility no
MaxClientsNumber    50
Daemonize           yes
MaxClientsPerIP     8
VerboseLog          no
DisplayDotFiles     yes
AnonymousOnly       no
NoAnonymous         no
SyslogFacility      ftp
DontResolve         yes
MaxIdleTime         15
PureDB              /usr/local/pureftpd/etc/pureftpd.pdb
```

```

LimitRecursion      2000 8
AnonymousCanCreateDirs  no
MaxLoad             4
AntiWarez           yes
Umask               133:022
MinUID              100
AllowUserFXP        no
AllowAnonymousFXP   no
ProhibitDotFilesWrite  no
ProhibitDotFilesRead  no
AutoRename          no
AnonymousCantUpload  no
PIDFile             /usr/local/pureftpd/var/run/pure-ftpd.pid
MaxDiskUsage        99
CustomerProof       yes

```

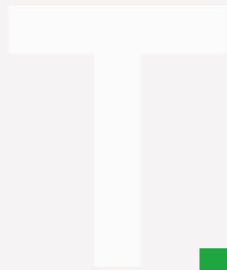
#####到此结束，保存即可#####

启动命令： /usr/local/pureftpd/sbin/pure-config.pl /usr/local/pureftpd/etc/pure-ftpd.conf

#####接下来该建立用户了#####

/usr/local/pureftpd/bin/pure-pw useradd ftp_test -u www -d /data/wwwroot其中，-u 将虚拟用户 ftp_test 与系统用户 www 关联在一起。-d 参数使 ftp_test 只能访问其主目录。执行完上述命令后，会提示输入密码。

/usr/local/pureftpd/bin/pure-pw mkdb



21

配置 squid 服务



什么是squid

Squid是比较知名的代理软件，它不仅可以在linux上还可以跑在windows以及Unix上，它的技术已经非常成熟。目前使用Squid的用户也是十分广泛的。Squid与Linux下其它的代理软件如Apache、Socks、TIS FWTK和delegate相比，下载安装简单，配置简单灵活，支持缓存和多种协议。Squid的缓存功能相当好用，不仅可以减少带宽的占用，同样也大大降低了后台的WEB服务器的磁盘I/O的压力。Squid接收用户的下载申请，并自动处理所下载的数据。也就是说，当一个用户象要下载一个主页时，它向Squid发出一个申请，要Squid替它下载，然后Squid 连接所申请网站并请求该主页，接着把该主页传给用户同时保留一个备份，当别的用户申请同样的页面时，Squid把保存的备份立即传给用户，使用户觉得速度相当快。Squid将数据元缓存在内存中，同时也缓存DNS查寻的结果，除此之外，它还支持非模块化的DNS查询，对失败的请求进行消极缓存。Squid支持SSL，支持访问控制。由于使用了ICP，Squid能够实现重叠的代理阵列，从而最大限度的节约带宽。Squid对硬件的要求是内存一定要大，不应小于128M，硬盘转速越快越好，最好使用服务器专用SCSI 硬盘，处理器要求不高，400MH以上既可。

安装squid

```
wget http://www.squid-cache.org/Versions/v2/2.6/squid-2.6.STABLE20.tar.gz

tar zxvf squid-2.6.STABLE20.tar.gz

cd squid-2.6.STABLE20
ulimit -HSn 65535

useradd squid
```

编译参数

```
./configure --prefix=/usr/local/squid \
--disable-dependency-tracking \
--enable-dlmalloc \
--enable-gnuregex \
--disable-carp \
--enable-async-io=240 \
--with-pthreads \
--enable-storeio=ufs,aufs,diskd,null \
--disable-wccp \
--disable-wccpv2 \
--enable-kill-parent-hack \
--enable-cachemgr-hostname=localhost \
```

```

--enable-default-err-language=Simplify_Chinese \
--with-build-environment=POSIX_V6_ILP32_OFFBIG \
--with-maxfd=65535 \
--with-aio \
--disable-poll \
--enable-epoll \
--enable-linux-netfilter \
--enable-large-cache-files \
--disable-ident-lookups \
--enable-default-hostsfile=/etc/hosts \
--with-dl \
--with-large-files \
--enable-removal-policies=heap,lru \
--enable-delay-pools \
--enable-snmp \
--disable-internal-dns

make && make install

```

关于squid的版本，有必要提一下，目前squid最新版本已经到了3.1了，但是笔者认为2.6版本比较好用，如果你有兴趣可以研究一下3.1。

squid配置

编辑配置文件 /usr/local/squid/etc/squid.conf

把原来配置文件删除，替换成：

```
http_port 80 transparent
```

```
cache_replacement_policy lru #如果有多个（下面两行）缓存目录，则需要写这个参数
```

```
cache_dir aufs /cache1 8192 16 256 #缓存目录1 /cache1 大小为8G
```

```
cache_dir aufs /cache2 4096 16 256 #缓存目录2 /cache2 大小为4G
```

上面两行定义了缓存目录，这个缓存目录可以只有一个，也可以定义很多个。

```

cache_mem 2048 MB #分配多少内存给squid，建议留至少512M给系统，如果你是虚拟机内存很小，只作为试验用的话，那就分一
maximum_object_size 2048 KB #缓存的文件最大不能超过2M
maximum_object_size_in_memory 512 KB #缓存在内存中的文件最大不超过512k
visible_hostname cache.example.com #显示给用户的主机名
client_persistent_connections off #client端关闭长连接
server_persistent_connections on #server端打开长连接

```



```
memory_pools on
memory_pools_limit 1024 MB
forwarded_for on
log_icp_queries off
cache_mgr cache@example.com #定义管理员的mail为cache@example.com
via on
httpd_suppress_version_string off
cache_effective_user squid #定义以squid用户的身份运行squid
cache_effective_group squid
error_directory /usr/local/squid/share/errors/Simplify_Chinese
icon_directory /usr/local/squid/share/icons
mime_table /usr/local/squid/etc/mime.conf
ie_refresh off
tcp_recv_bufsize 32 KB

acl all src 0.0.0.0/0.0.0.0
acl localhost src 127.0.0.0/8
acl Mgr_ip src 127.0.0.0/8
acl allow_ip dst 127.0.0.0/8 192.168.0.0/16 #定义允许代理的web的IP或者IP段
acl PURGE method PURGE
acl Safe_ports port 80 8080
acl CONNECT method CONNECT
acl manager proto cache_object
acl HTTP proto HTTP

http_access allow allow_ip
http_access allow manager Mgr_ip
http_access deny manager
http_access deny PURGE
http_access deny !Safe_ports
http_access deny all
icp_access deny all
ipcache_size 1024
ipcache_low 90
ipcache_high 95
memory_replacement_policy lru
hosts_file /etc/hosts
request_header_max_size 128 KB
hierarchy_stoplist cgi-bin ? \.php \.html
acl QUERY urlpath_regex cgi-bin \? \.php \.html
cache deny QUERY
quick_abort_min -1 KB
quick_abort_max 32 KB
quick_abort_pct 95
# error page
```

```

#error_map http://www.92csz.com/404.html 403
#deny_info http://www.92csz.com/error.html cctv_Domain
# timeout
peer_connect_timeout 20 seconds
connect_timeout 20 seconds
read_timeout 60 seconds
request_timeout 20 seconds
pconn_timeout 20 seconds
shutdown_lifetime 5 seconds
strip_query_terms off
icp_port 0
# logfile
emulate_httpd_log on
logformat combined %>a %ui %un [%t] "%rm %ru HTTP/%rv" %Hs %<st "%{Referer}>h" "%{User-Agent}>h" %Ss:%Sh
#access_log /log/squid-log/access.log combined
cache_store_log /dev/null
cache_log /var/log/squid/cache.log
logfile_rotate 12
# MISCELLANEOUS
store_objects_per_bucket 15
client_db off

```

修改完配置文件后保存，然后初始化squid

```

mkdir /cache1 /cache2 /var/log/squid

chown -R squid:squid /cache1 /cache2 /var/log/squid

/usr/local/squid/sbin/squid -z

```

用来生成cache目录，如果你的配置文件配置出错，往往会在初始化的时候报错，错误信息会直接显示在屏幕上。初始化成功后，就可以启动squid了，启动命令为：

```
nohup /usr/local/squid/bin/RunCache &
```

启动后，可以去看看cache.log 在这个日志中，你可以看到很多关于squid的信息，当然也包括一些错误日志。

如果想开机启动则需要要在/etc/rc.d/rc.local中最后加入一行

```
/usr/local/bin/RunCache &
```

到这里算是配置完成了，但是还有一个问题，就是如何定义被代理的web以及域名？单单看配置文件并没有说代理的web是哪一个。确实，这个配置文件其实可以代理多台web，只要你在/etc/hosts中定义要代理的域名以及IP即可，hosts格式在前面已经介绍过。笔者要提醒你的是，如果是一台web上的多个域名，请不要写一行，虽然hosts是允许的，但是如果写成一个IP对应多个域名，squid代理时就会出错。所以有几个域名就要写几行。

更改/etc/hosts后要重启squid才能生效:

```
/usr/local/squid/sbin/squid -krec
```

在重启前可以先检测一下, 是否有错, 命令为:

```
/usr/local/squid/sbin/squid -kcheck
```

如果没有错, 则不会显示任何信息, 否则会显示一些信息出来。



T



22

配置 Tomcat




```
vim /etc/profile
```

在末尾输入以下内容

```
#set java environment
JAVA_HOME=/usr/local/jdk1.6.0_23/
JAVA_BIN=/usr/local/jdk1.6.0_23/bin
JRE_HOME=/usr/local/jdk1.6.0_23/jre
PATH=$PATH:/usr/local/jdk1.6.0_23/bin:/usr/local/jdk1.6.0_23/jre/bin
CLASSPATH=/usr/local/jdk1.6.0_23/jre/lib:/usr/local/jdk1.6.0_23/lib:/usr/local/jdk1.6.0_23/jre/lib/charsets.jar
export JAVA_HOME JAVA_BIN JRE_HOME PATH CLASSPATH
```

执行命令source /etc/profile，使配置立即生效

```
source /etc/profile
```

检测是否设置正确：

```
java -version
```

如果显示如下内容，则配置正确。

```
java version "1.4.2"
gij (GNU libgcj) version 4.1.2 20080704 (Red Hat 4.1.2-46)

Copyright (C) 2006 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or F
```

3. 安装Tomcat

```
cd /usr/local/src/
```

```
wget http://archive.apache.org/dist/tomcat/tomcat-7/v7.0.14/bin/apache-tomcat-7.0.14.tar.gz
```

如果觉得这个版本不适合你，请到tomcat官方网站下载适合你的版本。

```
tar zxvf apache-tomcat-7.0.14.tar.gz
mv apache-tomcat-7.0.14 /usr/local/tomcat
cp -p /usr/local/tomcat/bin/catalina.sh /etc/init.d/tomcat
vim /etc/init.d/tomcat
```

在第二行加入以下内容：

```
# chkconfig: 2345 63 37
# description: tomcat server init script
```

```
JAVA_HOME=/usr/local/jdk1.6.0_23/
CATALINA_HOME=/usr/local/tomcat
```

```
chmod 755 /etc/init.d/tomcat
chkconfig --add tomcat
chkconfig tomcat on
...
```

启动tomcat:

```
service tomcat start
```

查看是否启动成功:

```
ps aux |grep tomcat
```

如果有进程的话，请在浏览器中输入http://IP:8080/ 你会看到tomcat的主界面。

配置tomcat

在配置tomcat前，先来看看tomcat的几个目录：

```
find /usr/local/tomcat/ -maxdepth 1 -type d ( -maxdepth的作用指定目录级数，后边跟1代表只查找1级目录 )
/usr/local/tomcat/ /usr/local/tomcat/lib # tomcat的库文件目录 /usr/local/tomcat/temp # 临时文件存放目录
/usr/local/tomcat/webapps # web应用目录，也就是我们访问的web程序文件所在目录 /usr/local/tomcat/conf # 配置文件目录
/usr/local/tomcat/logs # 日志文件所在目录 /usr/local/tomcat/work # 存放JSP编译后产生的class文件 /usr/local/tomcat/bin # tomcat的脚本文件
Tomcat的主配置文件为/usr/local/tomcat/conf/server.xml
```

1. 配置tomcat服务的访问端口。

默认是8080，如果你想修改为80，则需要修改server.xml文件。

找到 <Connector port="8080" protocol="HTTP/1.1"

修改为: <Connector port="80" protocol="HTTP/1.1"

2. 配置新的虚拟主机

```
cd /usr/local/tomcat/conf/ vim server.xml
```

找到</Host>，下一行插入新的<Host>，内容如下：

```
<Host name="www.example.cn" appBase="/data/tomcatweb"  
  unpackWARs="false" autoDeploy="true"  
  xmlValidation="false" xmlNamespaceAware="false">  
<Context path="" docBase="." debug="0" reloadable="true" crossContext="true"/>
```

...

完成后，重启tomcat

```
service tomcat stop; service tomcat start
```

测试新建的虚拟主机，首先需要修改你电脑的hosts文件

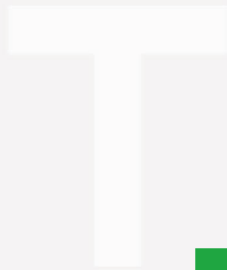
vim /data/tomcatweb/test.jsp 加入以下内容：

```
<html><body><center>  
Now time is: <%=new java.util.Date()%>  
</center></body></html>
```

保存后，在你的浏览器里输入 `http://www.example.cn/test.jsp` 看是否访问到如下内容：

```
Now time is: Thu Jun 02 14:32:34 CST 2011
```

上面的test.jsp就是要显示当前系统的时间。



配置 samba 服务器



以前我们在 Windows 上共享文件的话，只需右击要共享的文件夹然后选择共享相关的选项设置即可。然而如何实现 Windows 和 linux 的文件共享呢？这就涉及到了 samba 服务了，这个软件配置起来也不难，使用也非常简单。

samba 配置文件 smb.conf

一般你装系统的时候会默认安装 samba，如果没有安装，只需要运行这个命令安装(CentOS)：“yum install -y samba samba-client” Samba 的配置文件为/etc/samba/smb.conf，通过修改这个配置文件来完成我们的各种需求。打开这个配置文件，你会发现很多内容都用“#”或者“;”注释掉了。先看一下未被注释掉的部分：

```
[global]
    workgroup = MYGROUP
    server string = Samba Server Version %v
    security = user
    passdb backend = tdbsam
    load printers = yes
    cups options = raw

[homes]
    comment = Home Directories
    browseable = no
    writable = yes

[printers]
    comment = All Printers
    path = /var/spool/samba
    browseable = no
    guest ok = no
    writable = no
    printable = yes
```

主要有以上三个部分：[global], [homes], [printers]。

[global] 定义全局的配置，“workgroup”用来定义工作组，相信如果你安装过 windows 的系统，你会对这个 workgroup 不陌生。一般情况下，需要我们把这里的“MYGROUP”改成“WORKGROUP”（windows 默认的工作组名字）。

security = user #这里指定 samba 的安全等级。关于安全等级有四种：

- share: 用户不需要账户及密码即可登录 samba 服务器
- user: 由提供服务的 samba 服务器负责检查账户及密码（默认）
- server: 检查账户及密码的工作由另一台 windows 或 samba 服务器负责
- domain: 指定 windows 域控制服务器来验证用户的账户及密码。

passwd backend = tdb sam # passwd backend (用户后台), samba有三种用户后台: smbpasswd, tdb sam和ldapsam。

smbpasswd: 该方式是使用smb工具smbpasswd给系统用户(真实用户或者虚拟用户)设置一个Samba密码,客户端就用此密码访问Samba资源。smbpasswd在/etc/samba中,有时需要手工创建该文件。

tdbsam: 使用数据库文件创建用户数据库。数据库文件叫passwd.tdb,在/etc/samba中。passwd.tdb用户数据库可使用smbpasswd -a创建Samba用户,要创建的Samba用户必须先是在系统用户。也可使用pdbedit创建Samba账户。

pdbedit参数很多,列出几个主要的:

-pdbedit -a username: 新建Samba账户。-pdbedit -x username: 删除Samba账户。-pdbedit -L: 列出Samba用户列表,读取passwd.tdb数据库文件。-pdbedit -Lv: 列出Samba用户列表详细信息。-pdbedit -c "[D]" -u username: 暂停该Samba用户账号。-pdbedit -c "[]" -u username: 恢复该Samba用户账号。

ldapsam: 基于LDAP账户管理方式验证用户。首先要建立LDAP服务,设置“passwd backend = ldapsam:ldap://LDAP Server” load printers 和 cups options 两个参数用来设置打印机相关。

除了这些参数外,还有几个参数需要你了解:

- netbios name = MYSERVER # 设置出现在“网上邻居”中的主机名
- hosts allow = 127. 192.168.12. 192.168.13. # 用来设置允许的主机,如果在前面加“;”则表示允许所有主机
- log file = /var/log/samba/%m.log # 定义samba的日志,这里的%m是上面的netbios name
- max log size = 50 # 指定日志的最大容量,单位是K

[homes] 该部分内容共享用户自己的家目录,也就是说,当用户登录到samba服务器上时实际上是进入到该用户的家目录,用户登陆后,共享名不是homes而是用户自己的标识符,对于单纯的文件共享的环境来说,这部分可以无视掉。

[printers] 该部分内容设置打印机共享。

samba实践

注意: 在试验之前,请先检测selinux是否关闭,否则可能会试验不成功。

1. 共享一个目录,任何人都可以访问,即不用输入密码即可访问,要求只读。

打开samba的配置文件/etc/samba/smb.conf

[global]部分

把” MY GROUP” 改成” WORKGROUP”

把” security = user” 修改为 “security = share”

然后在文件的最末尾处加入以下内容：

```
[share]
    comment = share all
    path = /tmp/samba
    browseable = yes
    public = yes
    writable = no

mkdir /tmp/samba
chmod 777 /tmp/samba
```

启动samba服务

```
/etc/init.d/smb start
```

测试：

首先测试你配置的smb.conf是否正确，用下面的命令

```
testparm
```

如果没有错误，则在你的windows机器上的浏览器中输入 file://IP/share 看是否能访问

1. 共享一个目录，使用用户名和密码登录后才可以访问，要求可以读写

打开samba的配置文件/etc/samba/smb.conf

[global] 部分内容如下：

```
[global]
    workgroup = WORKGROUP
    server string = Samba Server Version %v
    security = user
    passdb backend = tdbsam
    load printers = yes
    cups options = raw
```

然后加入以下内容：

```
[myshare]
comment = share for users
path = /samba
browseable = yes
writable = yes
public = no
```

保存配置文件，创建目录：

```
mkdir /samba
```

```
chmod 777 /samba
```

然后添加用户。因为在[global]中” passdb backend = tdbsam”，所以要使用” pdbedit” 来增加用户，注意添加的用户必须在系统中存在。

```
useradd user1 user2
pdbedit -a user1 # 添加user1账号，并定义其密码
pdbedit -a user2
pdbedit -L # 列出所有的账号
```

测试：

打开IE浏览器输入file://IP/myshare/ 然后输入用户名和密码

1. 使用linux访问samba服务器

Samba服务在linux下同样可以访问。前提是你的linux安装了samba-client软件包。安装完后就可以使用smbclient命令了。

```
smbclient //IP/共享名 -U 用户名
```

```
如： [root@localhost]# smbclient //10.0.4.67/myshare/ -U user1
```

Password:

```
Domain=[LOCALHOST] OS=[Unix] Server=[Samba 3.0.33-3.29.el5_6.2]
```

```
smb: >
```

出现如上所示的界面。可以打一个”？” 列出所有可以使用的命令。常用的有cd, ls, rm, pwd, tar, mkdir, chown, get, put等等，使用help + 命令可以打印该命令如何使用，其中get是下载，put是上传。

另外的方式就是通过mount挂载了：

如：

```
mount -t cifs //10.0.4.67/myshare /mnt -o username=user1,password=123456
```

格式就是这样，要指定-t cifs //IP/共享名 本地挂载点 -o后面跟username 和 password

挂载完后就可以像使用本地的目录一样使用共享的目录了。



24

使用 Nagios 搭建监控服务器



关于Nagios

Nagios是一款用于监控系统和服务网络的开源应用程序，它的模式是服务器—客户端，也就是说首先要在一台服务器上（server）部署相应的主要套件，然后在要监控的服务器上部署客户端程序，这样server会和client通信，从而监控client端的各项资源。Nagios功能十分强大几乎所有的项目都可以监控，大到服务器的存活状态，小到服务器上的某一个服务（web）。这些功能都是通过自定义插件（或者叫做脚本）来实现。

当Nagios监控到某项资源发生异常会通知到用户，你可以接入手机短信接口也可以接入邮件接口。我们可以通过web页面来查看Nagios所监控的各项资源，默认搭建的Nagios服务器只能监控简单的几个项目，而其他服务之类的监控项目都是由我们自己开发的插件来实现的。

需要下载的软件

```
nagios-3.0.5
```

```
nagios-plugins-1.4.13
```

```
nrpe-2.12.tar.gz
```

```
apache-2.2.11
```

```
// 以上软件版本可以不一样
```

监控中心Server端的配置

1. 安装apache（略，请参考第16章中相关内容，只需安装，到后边再配置）
2. 建立nagios账户

```
useradd nagios
```

1. 下载软件

```
cd /usr/local/src/
```

```
wget http://syslab.comsenz.com/downloads/linux/nagios-3.0.5.tar.gz
```

```
wget http://syslab.comsenz.com/downloads/linux/nagios-plugins-1.4.13.tar.gz
```

```
wget http://syslab.comsenz.com/downloads/linux/nrpe-2.12.tar.gz
```


1. 编译安装nagios

```
cd /usr/local/src/

tar zxvf nagios-3.0.5.tar.gz

cd nagios-3.0.5

./configure --prefix=/usr/local/nagios

make all

make install

make install-init # 把nagios做成一个运行脚本，使nagios随系统开机启动

make install-config # 把配置文件样例复制到nagios的安装目录

make install-commandmode # 给外部命令访问nagios配置文件的权限

chown -R nagios:nagios /usr/local/nagios
```

1. 编译安装nagios-plugins

```
cd /usr/local/src/

tar zxvf nagios-plugins-1.4.13.tar.gz

cd nagios-plugins-1.4.13

./configure --prefix=/usr/local/nagios --with-nagios-user=nagios --with-nagios-group=nagios

make && make install
```

查看是否安装成功的方法是：

```
ls /usr/local/nagios/libexec/ 看这个目录下是否有插件文件
```

1. 安装nrpe

```
cd /usr/local/src/

tar zxvf nrpe-2.12.tar.gz

cd nrpe-2.12

./configure --enable-ssl --enable-command-args
```

```
make all
```

```
make install-plugin
```

```
make install-daemon
```

```
make install-daemon-config
```

1. 配置web接口

```
vim /usr/local/apache2/conf/httpd.conf
```

在最后加入以下内容：

```
ScriptAlias /nagios/cgi-bin /usr/local/nagios/sbin
<Directory "/usr/local/nagios/sbin/">
    AllowOverride AuthConfig
    Options ExecCGI
    Order allow,deny
    Allow from all
</Directory>
Alias /nagios/ /usr/local/nagios/share/
<Directory "/usr/local/nagios/share">
    Options None
    AllowOverride AuthConfig
    Order allow,deny
    Allow from all
</Directory>
```

1. 配置nagios

```
cd /usr/local/nagios/etc/
```

```
vim cgi.cfg
```

把 `use_authentication=1` 改成 `use_authentication=0` 意思是不用用户验证

1. 启动nagios

在启动前先检测一下：

```
/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
```

如果最后显示如下，则说明配置没有问题了。

```
Total Warnings: 0
```

Total Errors: 0

启动命令:

```
/etc/init.d/nagios start
```

或者:

```
/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
```

此时, 就可以访问web页面的nagios了, 在浏览器中输入:

```
http://IP/nagios/ 看看吧。
```

在要监控的机器上client部署nagios

如果你打开了web页面, 点击左栏的Host Detail 会在右栏看到一行数据, 其中Host 名为 “localhost”, Status显示为” up”, 并且显示为绿色, 如果是其他颜色就说明你的localhost出了问题。目前只有一行数据, 也就是说只监控了监控中心 (localhost) 一台主机, 那么如何添加其他机器被它监控呢? 这就需要在要被监控的机器上也部署nagios软件。

1. 添加账户

```
useradd nagios
```

1. 安装nrpe

```
cd /usr/local/src/
```

```
wget http://syslab.comsenz.com/downloads/linux/nrpe-2.12.tar.gz
```

```
tar zxvf nrpe-2.12.tar.gz
```

```
cd nrpe-2.12
```

```
./configure --enable-ssl --enable-command-args
```

```
make all
```

```
make install-plugin
```

```
make install-daemon
```

```
make install-daemon-config
```

1. 安装nagios-plugin

```
cd /usr/local/src/
```

```
wget http://syslab.comsenz.com/downloads/linux/nagios-plugins-1.4.13.tar.gz
```

```
tar zxvf nagios-plugins-1.4.13.tar.gz
```

```
cd nagios-plugins-1.4.13
```

```
./configure --prefix=/usr/local/nagios --with-nagios-user=nagios --with-nagios-group=nagios
```

```
make && make install
```

到此就算安装完成了，请查看/usr/local/nagios/目录下是否有四个目录分别为：bin etc libexec share 另外在libexec目录下会有很多check_开头的文件。如果你的机器上没有，就请重新安装吧。

1. 配置

```
vim /usr/local/nagios/etc/nrpe.cfg
```

找到” allowed_hosts=127.0.0.1” 改成 “allowed_hosts=127.0.0.1,10.0.4.67”

// 后边的IP是server的IP

找到” dont_blame_nrpe=0” 改成 “dont_blame_nrpe=1”

1. 启动nrpe

```
/usr/local/nagios/bin/nrpe -c /usr/local/nagios/etc/nrpe.cfg -d
```

在监控中心添加被监控主机

添加主机当然是要到server端（监控中心）修改配置文件了。

1. 修改主配置文件

```
cd /usr/local/nagios/etc/
```

```
vim nagios.cfg
```

增加内容：

```
cfg_dir=/usr/local/nagios/etc/services ##定义一个目录，以后把新增加的主机信息文件全部放到这里
```

1. 添加被监控主机信息

```
mkdir /usr/local/nagios/etc/services
```

```
cd /usr/local/nagios/etc/services
```

vim 10.0.4.56.cfg 加入如下内容:

```
define host{
    use linux-server
    host_name 10.0.4.56
    alias 10.0.4.56
    address 10.0.4.56
}
define service{
    use generic-service
    host_name 10.0.4.56
    service_description check_ping
    check_command check_ping!100.0,20%!200.0,50%
    max_check_attempts 5
    normal_check_interval 1
}
define service{
    use generic-service
    host_name 10.0.4.56
    service_description check_ssh
    check_command check_ssh
    max_check_attempts 5
    normal_check_interval 1
}
define service{
    use generic-service
    host_name 10.0.4.56
    service_description check_http
    check_command check_http
    max_check_attempts 5
    normal_check_interval 1
}
```

// 注意，这里的IP是client端的IP，监控的项目有三个ping, ssh, http。其实这三个项目使用的脚本都为本地脚本，也就是说，即使远程主机没有安装nagios和nrpe同样可以监控这些项目。但是如果监控load, disk, 等等就需要通过nrpe服务来搞定了，道理很简单，load和disk都需要登录到远程主机上去获得信息，而ping, ssh, http都不需要的。这个到远程主机获取相关的信息的过程是由nrpe完成的。如果你的client上没有启动nrpe服

务那么我们是无法获取远程主机的load和disk等信息的。下面笔者配置一下使用nrpe来监控远程主机的相关项目。

在server端编辑/usr/local/nagios/etc/objects/commands.cfg

vim /usr/local/nagios/etc/objects/commands.cfg # 在最后面添加如下内容

```
define command{
    command_name    check_nrpe
    command_line    $USER1$/check_nrpe -H $HOSTADDRESS$ -c $ARG1$
}
```

然后编辑10.0.4.56.cfg（还是server上）

cd /usr/local/nagios/etc/services

vim 10.0.4.56.cfg # 加入如下内容:

```
define service{
    use    generic-service
    host_name    10.0.4.56
    service_description    check_load
    check_command    check_nrpe!check_load
    max_check_attempts    5
    normal_check_interval    1
}

define service{
    use    generic-service
    host_name    10.0.4.56
    service_description    check_disk_hda1
    check_command    check_nrpe!check_hda1
    max_check_attempts    5
    normal_check_interval    1
}

define service{
    use    generic-service
    host_name    10.0.4.56
    service_description    check_disk_hda2
    check_command    check_nrpe!check_hda2
    max_check_attempts    5
    normal_check_interval    1
}
```

这里需要解释一下相关的” check_command”，先看这个” check_nrpe!check_load” 这里的check_nrpe就是上面/usr/local/nagios/etc/objects/commands.cfg中刚刚定义的，后面的check_load是在远程主机上定义的一个命令脚本。具体在哪里定义稍后介绍。为什么中间加一个” !”，这个是nagios特有的形式，无需关心。下面需要到远程主机上去定义上面用到的脚本了。

在远程主机上编辑/usr/local/nagios/etc/nrpe.cfg 文件

```
vim /usr/local/nagios/etc/nrpe.cfg (client上)
```

把” command[check_hda1]” 那行改成：

```
command[check_hda1]=/usr/local/nagios/libexec/check_disk -w 20% -c 10% -p /dev/hda1
```

然后再增加一行：

```
command[check_hda2]=/usr/local/nagios/libexec/check_disk -w 20% -c 10% -p /dev/hda2
```

这里的check_hda1 和 check_hda2 都是自定义的，和server端的定义的service中的check_command对应。也就是说，如果在server端定义了一个service（通过nrpe方式）那么必须要在客户端上的nrpe.cfg中定义相应的脚本。保存这个文件后，需要重新启动一下nrpe服务。

```
killall nrpe ; /usr/local/nagios/bin/nrpe -c /usr/local/nagios/etc/nrpe.cfg -d (client上)
```

1. 重启nagios服务

修改了配置需要重启服务才能使添加的监控主机生效。

```
/etc/init.d/nagios restart (server上)
```

此时再到web页面去观察是否多了一台10.0.4.56

在nagios客户端上自定义监控脚本

在开始，笔者就讲过，我们可以自定义写监控脚步，从上面的例子中也可以看到监控磁盘状态时，是根据磁盘分区来监控的。这样不免有些麻烦，因为每台主机的磁盘分区状况都不一样（一样还好），而且有多少个分区就需要定义多少个命令。所以笔者就自定义写一个shell脚本来监控所有的磁盘分区：

1. 在客户端上创建脚本/usr/local/nagios/libexec/check_disk.sh

```
vim /usr/local/nagios/libexec/check_disk.sh 写入如下内容：(client上)
```

```
#!/bin/bash
row=`df -h -P|wc -l`
```

```

status=0
for i in `seq 2 $row`
do
    spare=`df -h -P|sed -n "$i"|awk '{print $4}'`
    use_percentage=`df -h -P|sed -n "$i"|sed -n "s/^\///"|awk '{print $5}'`
    spare_percentage=`expr 100 - $use_percentage`
    partition_name=`df -h -P|sed -n "$i"|awk '{print $6}'`
    if [ "$spare_percentage" -lt "3" ];then
        echo -n "$partition_name CRITICAL ${spare_percentage}% $spare "
        status[$i]=2
    elif [ "$spare_percentage" -lt "5" ];then
        echo -n "$partition_name WARNING! ${spare_percentage}% $spare "
        status[$i]=1
    else
        echo -n "$partition_name OK ${spare_percentage}% $spare "
        status[$i]=0
    fi
done
zhuangtai=0
for j in `seq 2 $row`
do
    if [ "${status[$j]}" -gt "$zhuangtai" ];then
        zhuangtai=${status[$j]}
    fi
done
exit $zhuangtai

```

1. 保存后，修改该脚本的权限

```
chmod +x /usr/local/nagios/libexec/check_disk.sh ( client上 )
```

1. 然后编辑/usr/local/nagios/etc/nrpe.cfg文件

```
vim /usr/local/nagios/etc/nrpe.cfg # 加入一行: ( client上 )
```

```
command[check_disk]=/usr/local/nagios/libexec/check_disk.sh
```

保存，重启nrpe服务

```
killall nrpe ; /usr/local/nagios/bin/nrpe -c /usr/local/nagios/etc/nrpe.cfg -d ( client上 )
```

1. 检测刚才的脚本是否正常运行的方法是，到server端执行如下命令：

```
/usr/local/nagios/libexec/check_nrpe -H 10.0.4.56 -c check_disk ( server上 )
```

如果正常的话，会输出一行磁盘检测的数据，否则可能会报错。

1. 到server上添加相应的service

```
cd /usr/local/nagios/etc/services ( server上 )
```

```
vim 10.0.4.56.cfg # 加入如下内容:
```

```
define service{
    use    generic-service
    host_name    10.0.4.56
    service_description    check_disk
    check_command    check_nrpe!check_disk
    max_check_attempts 5
    normal_check_interval 1
}
```

1. 重启nagios服务

```
/etc/init.d/nagios restart ( server上 )
```

配置nagios报警邮件

现在139邮箱有顺便发短信的功能，所以当有报警时，只需发送到你的139邮箱你就同样会收到一条报警短信。这样做的优势就是不用再去买短信网关了，节省了很大一笔钱。

```
vim /usr/local/nagios/etc/objects/contacts.cfg
```

把” email nagios@localhost” 修改成 “email 你的139邮箱”

```
vim /usr/local/nagios/etc/objects/templates.cfg
```

找到:

```
define service{
    name                generic-service
```

之所以看这一段，是因为在上面添加的10.0.4.56.cfg 定义了很多generic-service所以要关注这段的配置。

```
`` define service{ name generic-service
active_checks_enabled 1
passive_checks_enabled 1
parallelize_check 1
obsess_over_service 1
check_freshness 0
```

```

notifications_enabled 1
event_handler_enabled 1
flap_detection_enabled 1
failure_prediction_enabled 1
process_perf_data 1
retain_status_information 1
retain_nonstatus_information 1
is_volatile 0
check_period 24x7
max_check_attempts 3
normal_check_interval 10
retry_check_interval 2
contact_groups admins
notification_options w,u,c,r
notification_interval 60
notification_period 24x7
register 0
} ```` 其中有几个参数需要你注意:

```

- `notifications_enabled` : 是否开启提醒功能。1为开启, 0为禁用。一般, 这个选项会在主配置文件 (`nagios.cfg`) 中定义, 效果相同。
- `notification_interval`: 重复发送提醒信息的最短间隔时间。默认间隔时间是60分钟。如果这个值设置为 0, 将不会发送重复提醒。
- `notification_period`: 发送提醒的时间段。非常重要的主机 (服务) 我定义为 7×24 , 一般的主机 (服务) 就定义为上班时间。如果不在定义的时间段内, 无论什么问题发生, 都不会发送提醒。
- `notification_options`: 这个参数定义了发送提醒包括的情况: `d` = 状态为DOWN, `u` = 状态为UNREACHABLE, `r` = 状态恢复为OK, `f` = flapping., `n`=不发送提醒。

要想正确发送邮件, 上面的参数得配置合理才行。



小李的博客

www.6o06.com
