

绪论 Linux 系统简介和安装

1.1 Linux 系统简介

Linux 是一个完全免费的类 Unix 操作系统，它的标志是一个名为 Tux 的可爱的企鹅，如图 1-1 所示。Linux 诞生以来，凭借稳定、安全、高性能和高扩展等优点，得到广大用户的欢迎。成为目前最流行的操作系统之一。

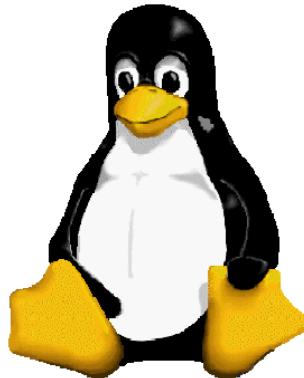


图 1-1

1.1.1 Linux 的发展历史

Linux 的诞生可以追溯到 1991 年，当 Linus 还是芬兰赫尔辛基大学的一名学生时，他对当时为教学而设计的 Minix 操作系统提供的功能不满意，于是他决定自己写比 Minix 更强大的类 UNIX 操作系统来取代 Minix，后来这个类 UNIX 操作系统就是 Linux。

Linus 从一开始就决定自由扩散 Linux，他把源代码发布在 Internet 上，随即就引起爱好者的注意，他们通过 Internet 加入了 Linux 的内核开发工作，一大批高水平程序员的加入，使得 Linux 得到迅猛发展，他们为 Linux 修复错误、增加新功能，不断尽其所能地改进它。现在，Linux 凭借优秀的设计，不凡的性能，加上 IBM、Intel、AMD、DELL、Oracle、Sybase 等国际知名企业的大力支持，市场份额逐步扩大，逐渐成为主流操作系统之一。

1.1.2 Linux 的版权问题

GNU 名字它是“GNU's Not UNIX”的缩写形式。

Linux 是基于 Copyleft（版权所无）的软件模式进行发布的，其实 Copyleft 是与 Copyright（版权所有）相对立的新名称，它是 GNU 项目制定的通用公共许可证 GPL（General Public License）。GNU 项目是由 Richard Stallman 于 1984 年提出的，他建立了自由软件基金会（FSF）并提出 GNU 计划的目的是开发一个完全自由的，与 UNIX 类似但功能更强大的操作系统，以便为所有的计算机使用者提供一个功能齐全，性能良好的基本系统。它的标志是角马，如图 1-2 所示。



图 1-2 GNU 的标志角马

GPL 是由自由软件基金会发行的用于计算机软件的协议证书，使用该证书的软件被称为自由软件（后来改名为开放源代码软件（Open Source Software））。大多数的 GNU 程序和超过半数的自由软件使用它。GPL 保证任何人有权使用、拷贝和修改该软件。任何人有权取得、修改和重新发布自由软件的源代码，并且规定在不增加附加费用的条件下可以得到自由软件的源代码。同时还规定自由软件的衍生作品必须以 GPL 作为它重新发布的许可协议。Copyleft 软件的组成更加透明化，这样当出现问题时，就可以准确地查明故障原因，及时采取相应回避策略，同时用户不用再担心有“后门”的威胁。

1. 1. 3 linux 的特点

1. 开放性

开放性是指系统遵循世界标准规范，特别是遵循开放系统互连（OSI）国际标准。凡遵循国际标准所开发的硬件和软件，都能彼此兼容，可方便地实现互连。

2. 多用户

多用户是指系统资源可以被不同用户各自拥有使用，即每个用户对自己的资源（例如：文件、设备）有特定的权限，互不影响。Linux 和 Unix 都具有多用户的特性。

3. 多任务

多任务是现代计算机的最主要的一个特点。它是指计算机同时执行多个程序，而且各个程序的运行互相独立。

4. 良好的用户界面

Linux 向用户提供了两种界面：用户界面和系统调用。Linux 的传统用户界面是基于文本的命令行界面。

Linux 还为用户提供了图形用户界面。它利用鼠标、菜单、窗口、滚动条等设施，给用户呈现一个直观、易操作、交互性强的友好的图形化界面。

5. 设备独立性

设备独立性是指操作系统把所有外部设备统一当作成文件来看待，只要安装它们的驱动

程序，任何用户都可以象使用文件一样，操纵、使用这些设备，而不必知道它们的具体存在形式。

6. 提供了丰富的网络功能

完善的内置网络是 Linux 的一大特点。Linux 在通信和网络功能方面优于其他操作系统。其他操作系统不包含如此紧密地和内核结合在一起的连接网络的能力，也没有内置这些联网特性的灵活性。

7. 可靠的系统安全

Linux 采取了许多安全技术措施，包括对读、写进行权限控制、带保护的子系统、审计跟踪、核心授权等，这为网络多用户环境中的用户提供了必要的安全保障。

8. 良好的可移植性

Linux 是一套遵从 POSIX（可移植性操作系统）规范的操作系统，可移植性是指将操作系统从一个平台转移到另一个平台使它仍然能按其自身的方式运行的能力。

1.1.4 Linux 内核版本和发行版本

1、内核版本

内核就好象人的心脏一样，在整个系统中占有举足的地位。Linux 内核之所受到人们的如此重视，是因为它是构成整个 Linux 操作系统最关键的组成部分，可以不夸张地说，没有 Linux 内核出现就没有今天 Linux 操作系统。

Linux 内核是运行程序和管理硬件设备的核心程序，主要包括文件管理，设备管理，内存管理，模块管理，网络管理，进程管理方面的模块。一般接受从运行期库和系统程序中传递过来的用户命令，执行后向用户返回结果。

Linux 内核的官方网站是 <http://www.kernel.org>，从该站点可下载已发布的每一个版本的内核文件。

Linux 内核版本的命名格式为 X.Y.Z，现在最新的版本为 3.14.8。

- X 为主版本号：“3”为主版本号。
- Y 为次版本号：用奇数表示开发版，用偶数表示稳定版。
- Z 为修订版本号：表示对同一内核版本的不断修订和升级。

2、发行版本

仅有内核而没有应用软件的操作系统是无法使用的，所以许多公司或社团将内核、源代码及相关的应用程序组织构成一个完整的操作系统，让一般的用户可以简便地安装和使用 Linux，这就是所谓的发行版本（distribution）。下面介绍目前比较著名的发行版本。

(1) RedHat Linux

RedHat 是最成功的 Linux 发行版本之一，也是全球最流行的 Linux 版本，RedHat 已成 Linux 代名词，许多人一提到 Linux 就毫不犹豫地想到 RedHat。

- RedHat Linux 系列发行版

RedHat Linux 系列发行版是 RedHat 早期面向普通用户使用的 Linux 发行版本，提供免费下载和使用服务，深受广大 Linux 用户的喜爱。而然遗憾的是 RedHat Linux 系列发行版已经停止了开发，最后一个版本是 9.0 版。

➤ RedHat Linux 企业版

RedHat Linux 企业版（RedHat Enterprise Linux）简称 RHEL，用 RedHat 公司面向企业级应用推出的发行版。RHEL 系列产品是商业化的 Linux 发行版本，不提供免费下载和使用，但是仍然遵循开源软件的授权许可，免费提供所有源代码下载。目前系列产品的软新版本为 7.x 系列。本教材使用 RHEL7.0 版本。

➤ CentOS

CentOS（Community Enterprise Operating System，中文意思是：社区企业操作系统）是 Linux 发行版之一，它是来自于 Red Hat Enterprise Linux 依照开放源代码规定释出的源代码所编译而成。由于出自同样的源代码，因此有些要求高度稳定性的服务器以 CentOS 替代商业版的 Red Hat Enterprise Linux 使用。两者的不同，在于 CentOS 并不包含封闭源代码软件，2014 年加入 RedHat.

官方网站：<http://www.redhat.com>



(2) Ubuntu Linux

Ubuntu 首个版本发布于 2004 年 9 月，相对大多数 Linux 发行版来说，它的起步较晚，但是在朝着“易用”和“免费”方面不断发展的路途中，Ubuntu 赢得了大家的喜爱，并成为数一数二的 Linux 发行版本。

官网：<http://www.ubuntu.com/>



(3) Fedora Core 社区版

Fedora 是 RedHat 公司停止 RedHat Linux 系列开发后建立的 Linux 社区版本，可以理解为了 Fedora 是 RedHat Linux 系列产品的后继者。

Fedora Core 主要定位桌面用户，追求绚丽的桌面效果。

官方网站：<http://www.fedoraproject.org>



(4) Debian

Debian 计划是一个致力于创建一个自由操作系统的合作组织。Debian 可以算是迄今为止最遵循 GNU 规范的 Linux 系统，它使用 Debian 特有的软件包管理工具 `dpkg`，使得安装、升级、删除和管理软件变得非常简单。Debian 是完全由网络上 Linux 爱好者负责维护和发行套件。

官方网站: <http://www.debian.org>



(5) SuSE Linux

Novell SUSE Linux Enterprise Server (SLES) 是一个以 Linux 内核为基础的类 Unix 企业服务器操作系统，能运行在从 X86 PC 到小型机乃至超级计算机等硬件平台上，可以稳定高效的运行企业数据中心的所有主流应用业务。

Novell SLES 现已发展成为最受业内瞩目的、发展最为迅猛的主流服务器操作系统之一，成为 RedHat 一个强大竞争对手。

官方网站: <http://www.novell.com/linux/suse>



(6) 红旗 Linux

中国每一个土生土长的 Linux 发行版，对中文支持最好，界面操作的设计符合中国人的习惯。

官方网站: <http://www.redflag-linux.com>



(7) Anroid

Android 就是基于 Linux 的开放源码操作系统，目前在最火热的智能手机和平板电脑中得到了使用。Android 系统属 Google 所有，目前属于全球市场份额最大的智能手机操作系统。



1.1.5 Linux 系统在企业中应用

随着 Linux 系统逐步趋于稳定与完善，它能够提供的功能也越来越丰富、强大。尤其在企业级应用更显现出巨大的技术优势，主要体现在以下几个方面。

1、作为 Internet 网络服务器、数据库服务器的应用

Linux 是一种类 Unix 系统，因此很容易成为 Unix 替代品，原来在 Unix 上跑的网络服务，可以方便移植到 Linux 上。而相对 Windows 系统来说，Linux 系低廉的费用、高稳定性、可靠性、安全性等特点，则更受企业青睐。

结合不同功能的软件，Linux 操作系统可以为企业大多数常用用 Internet 网络服务器的应用，以下列举其中几个方面。

- 使用 Apache 构建 Web 站点服务器。
- 使用 BIND 构建 DNS 服务器。
- 使用 vsftpd 构建 FTP 文件服务器。
- 使用 Qmail 或 Postfix 构建电子邮件服务器。
- 使用 Oracle 构建数据库服务器。
- 使用 Linux 构建高可用性的服务器群集。

如 google、网易、腾讯等众多企业的 Web 站点和数据服务都使用 Linux 操作系统平台。

2、作为企业内部服务器的应用

Linux 操作系统同样适应于架设内部的服务器。现在越来越多企业选择用 Linux 作为服务器，用 Windows 作为桌面办公用机。Linux 在企业内部典型应用包括以下几个方面。

- 使用 DHCP 软件为局域网中的主机动态分配 IP 地址等参数。
- 使用 Samba 服务器软构建企业内部的文件与打印机共享服务器。
- 使用 Squid 服务器软件构建代理服务器。
- 使用 Linux 中的 iptables 构建网关和防火墙服务器。
- 使用 Linux 构建目录服务器。

3、作为软件开发环境的应用

4、云计算和虚拟化应用

1.2 安装 RedHat Linux 操作系统必备知识

安装 RedHat Linux 过程比安装 Windows 稍复杂，在安装之前要了解一些安装相关知识。

1.2.1 安装前准备

安装 Red Hat Linux 的硬件要求如下。

CPU: Pentium 以上处理器。

内存: 至少 128MB, 推荐使用 256MB 以上的内存。

硬盘: 至少需要 1GB 以上的硬盘空间，完全安装需大约 5GB 的硬盘空间。

显卡: VGA 兼容显卡。

光驱: CD-ROM/DVD-ROM。

其他设备: 如声卡、网卡和 Modem 等。

软驱: 可选

Red Hat 网站提供了经过兼容性测试和认证的“硬件兼容性列表”，在得到系统硬件设备的具体型号后，最好访问 <http://bugzilla.redhat.com/hwcert/> 来查看用户的配置是否在清单之中。

中星的服务器远高于以上的配置，自然不用担心硬件问题。

1.2.2 硬盘分区

硬盘分区，这是一个操作系统规划的重中之重，系统以后的扩展性，和安全性都与这步很有关系。Linux 下设备都是以文件形式来表示的，读取硬盘、光驱等资源时均采用“设备文件”形式进行，因些在 Linux 中将硬盘和分区表示为不同的文件。

每个硬盘上分区分为主分区（Primary Partition）、扩展分区（Extension Partition）和逻辑分区（Logical Partition）3 种。主分区最多只能有四个，而扩展分区（占一个主分区）中可以建立多个逻辑分区。在 Windows 系统中查看分区如图 1-3 所示



图 1-3 磁盘分区

1、硬盘接口

(1) IDE 接口

Linux 对连接到 IDE 接口的硬盘使用 /dev/hdx 的方式命名，x 的值对应于硬盘安装位置。X 的值可以是 a、b、c、d。以 hd 代表 IDE 硬盘，a b c d，是表示的硬盘号，表示第几个硬盘，第一个硬盘就是 a，第二个硬盘就是 b，依次推算。

硬盘	名称
IDE1 口的主盘	/dev/hda
IDE1 口的从盘	/dev/hdb
IDE2 口的主盘	/dev/hdc
IDE2 口的从盘	/dev/hdd

(2) SCSI 接口

SD 代表 SCSI 硬盘，USB 是属于 SCSI 设备的。

连接到 SCSI 接口的设备使用 ID 号进行区别，SCSI 设备 ID 号为 0~15，SCSI 接口卡本身的 ID 号是 7。Linux 对连接到 SCSI 接口卡的硬盘使用 /dev/sdx 的方式命名，x 的值可以是 a、b、c、d 等，即 ID 号为 0 的 SCSI 硬盘名为 /dev/sda，ID 号为 1 的 SCSI 硬盘名为 /dev/sdb，以此类推。

2、分区

在 Windows 系统中，使用盘符（如 C、D）来表示分区，那么在 Linux 下如何来表示这些分区呢？

Linux 表示分区时以硬盘的文件名做为基础，在后面添加对应的数字（从 1 开始编号）。

例 1：

那么我电脑的第一个 IDE 硬盘的第 1 个主分区完整表示文件名为 /dev/had1。

例 2：

在图 1-3 中 D 盘（第一个逻辑分区）在 Linux 下怎表示呢？

表示为 /dev/hda2，答案是错误。

一个硬盘最多可以有 4 个（主+扩展）分区，其中，扩展分区只能有一个。

因 1~4 号已被保留，所以第 1 个逻辑分区的代号由 5 号开始，以此顺序增加到磁盘号。

hda1 主分区	-----》 第一个可用分区	(对应图 1-3 中 C 盘)
hda2 主分区	-----》 第二个可用分区	
hda3 主分区	-----》 第三个可用分区	
hda4 扩展分区		
hda5 逻辑分区	-----》 第四个可用分区	(对应图 1-3 中 D 盘)
hda6 逻辑分区	-----》 第五个可用分区	(对应图 1-3 中 E 盘)

注意啦：

硬盘编号是从 0 开始的，分区编号是从 1 开始的。

1.2.3 Linux 文件系统

文件系统是操作系统最为重要的一部分，它定义了磁盘上储存、读取文件的方法和数据结构。文件系统是操作系统组织、存取和保存信息的重要手段，每种操作系统都有自己的文件系统，如 Windows 所用的文件系统主要有 FAT16、FAT32 和 NTFS，Linux 所用的文件系统主要有以下两种格式。

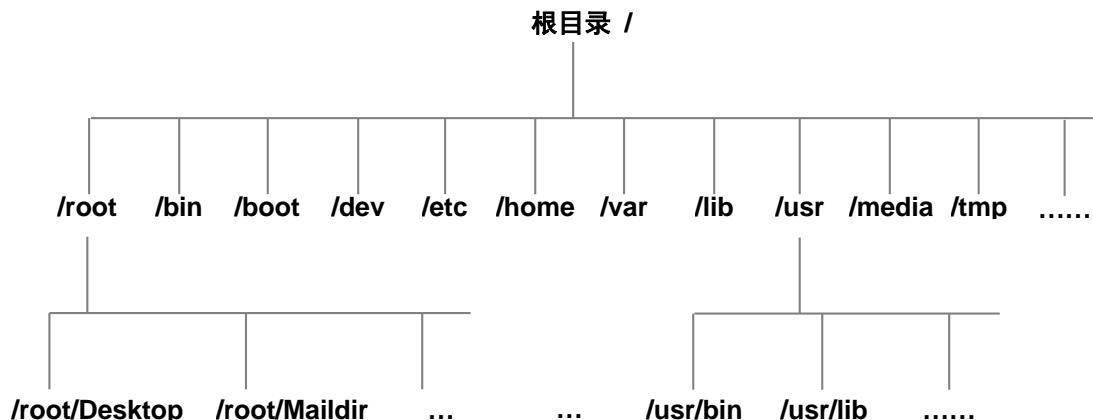
- Xfs：是 SGI 开发的高级日志文件系统，XFS 极具伸缩性，非常健壮。XFS 是一个全 64-bit 的文件系统，它可以支持上百万 T 字节的存储空间。对特大文件及小尺寸文件的支持都表现出众，支持特大数量的目录。最大可支持的文件大小为 $2^{63} = 9 \times 10^{18} = 9$ exabytes，最大文件系统尺寸为 18 exabytes
- EXT4：Ext4 是专门为 Linux 开发的原始的扩展文件系统（ext 或 extfs）的第四版。Linux kernel 自 2.6.28 开始正式支持新的文件系统 Ext4。Ext4 是 Ext3 的改进版，修改了 Ext3 中部分重要的数据结构，而不仅仅像 Ext3 对 Ext2 那样，只是增加了一个日志功能而已。Ext4 可以提供更佳的性能和可靠性，还有更为丰富的功能
- EXT3：是一种日志型的文件系统，特点是保存有对磁盘读写的日志，便于恢复，性能和稳定性更加出色。
- SWAP：交换文件系统，主要用于为 Linux 建立交换分区（相当于 Windows 下的虚拟内存），能够在一定程度缓解物理内存不足的问题，不能直接用于存储用户的文件。一般设为物理内存的 1.5~2 倍，当然现在服务器内存一般都比较大，不一定按此设定，例如你的内存足够大（如 8G 以上），你甚至可以不设交换分区。

1.2.3 Linux 目录结构及和分区关系

LINUX 分区就和 WINDOWS 有很多区别了，WINDOWS 是在分区里创建文件或者目录。而 LINUX 是把分区挂载到目录上。WINDOWS 是以每个分区为根，生成的目录树是以每个盘符为根往下延伸的，而 LINUX 只有一个根，他就是 /，它是 Linux 文件系统的起点。

在根目录下，Liunx 默认会建一些特殊的子目录，分别用于不同的用途，下面简单介绍其中常见子目录及分区建议。

- /bin：主要是存放普通用户的可执行命令。建议和 / 放在一起，不单独分出来。
- /dev：是系统设备文件存放位置，比如刚才的分区，建议和 / 放在一起，不单独分出来。
- /home：是普通用户的家目录，很多文件服务器都会用到用户的家目录存放资料，所以建议单独分区，而且还有一个好处，如果你系统坏了，实在不能用了，需要从新安装系统。你单独分出了 home 分区，里面的东西可以在安装好系统后，从新挂载进系统。就不会出现重要数据丢失的问题



- /boot: 存放 GRUB (启动装载程序) 和内核的文件 强烈建议单独分成第一个主分区，这样系统启动不了，比较容易排除故障。
- /etc: 大部分配置文件的存放目录。虽然重要但是不大，但是一定要和/放一起，因为启动的时候需要读取里面的配置文件，这个是不能单独分出去的。
- /root 超级用户的家目录，可以和/放一起。
- /var 是很多服务器文件使用的目录，如用户邮箱目录，建议单独分区。
- /usr: 存放其他用户的应用程序，通常被划分很多子目录，用于存不同类型的应用程序。
- lost+found : 分区的文件碎片。
- misc: 自动挂载服务需要的的目录，建议和 /放一起
- proc: 目录里的东西就是现在内存中的东西。不会占用硬盘空间，不须要单独分区
- sbin : 是超级管理员的可执行文件存放目录
- tmp : 临时文件目录，不需要单独分出来
- lib : 系统和可执行程序的库文件。 和/放一起
- media 和 mnt 也是挂载光驱和其他设备用的，和/放一起
- selinux 增强性安全 LINUX，和 usr 部分用户安装文件存放目录。也和/放一起

出于安全性和扩展性考虑我们会分出 4 个分区：/boot, /, /home, /var

var 和 home 目录是看你服务器性质而定。一般把剩余空间全部分配给他们

安装 Linux 时，需要在硬盘建立 Linux 使用的分区，在大多情况下，至少需要为 Linux 建立以下 3 个分区。

/boot 分区：/boot 分区用于引导系统，它包含了操作系统的内核和在启动系统过程中所要用到的文件，该分区的大小一般为 100MB。

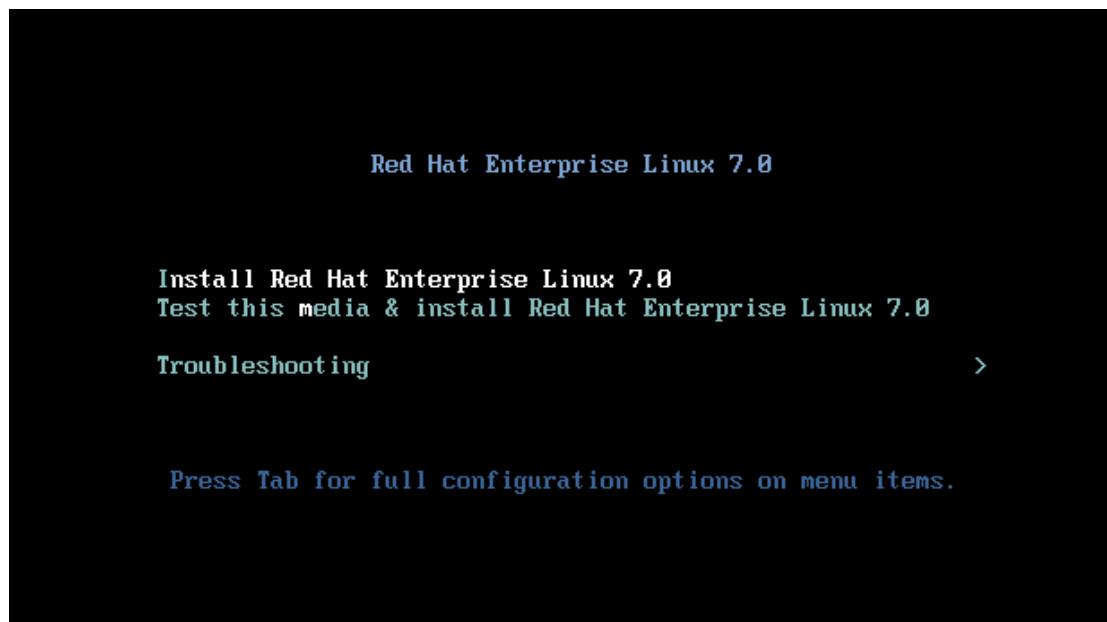
swap 分区：swap 分区的作用是充当虚拟内存，其大小通常是物理内存的两倍左右（当物理内存大于 512MB 时，swap 分区为 512MB 即可）。例如物理内存是 128MB，那么 swap 分区的大小应该是 256MB。

/ (根) 分区：Linux 将大部分的系统文件和用户文件都保存在/ (根) 分区上，所以该分区一定要足够大，一般要求大于 5GB。

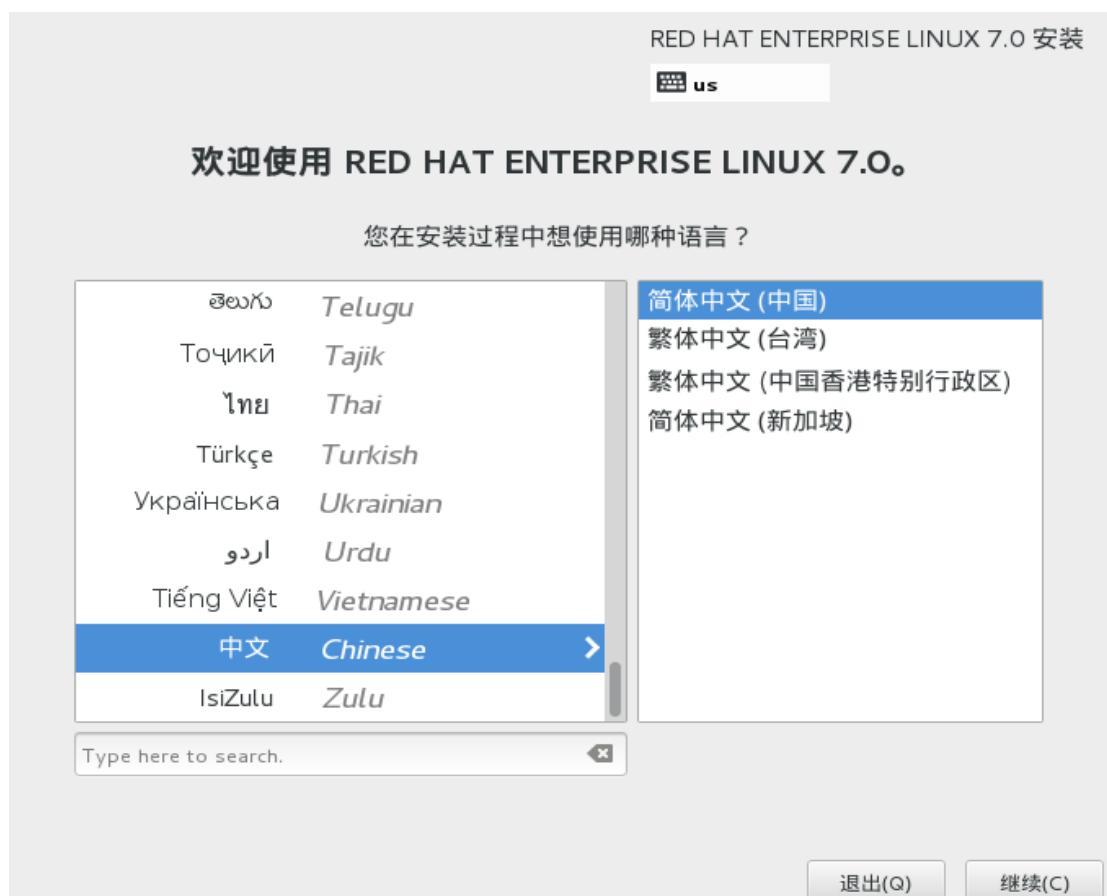
如果以上知识你都会了，那么安装 LINUX 对您来说就象切菜一样简单了。

1.3 安装 RedHat Enterprise Linux 7.0

- 从光盘启动之如下画面，点击“Install Red Hat Enterprise Linux 7.0”



- 选择安装过程使用的语言。



- 安装信息摘要



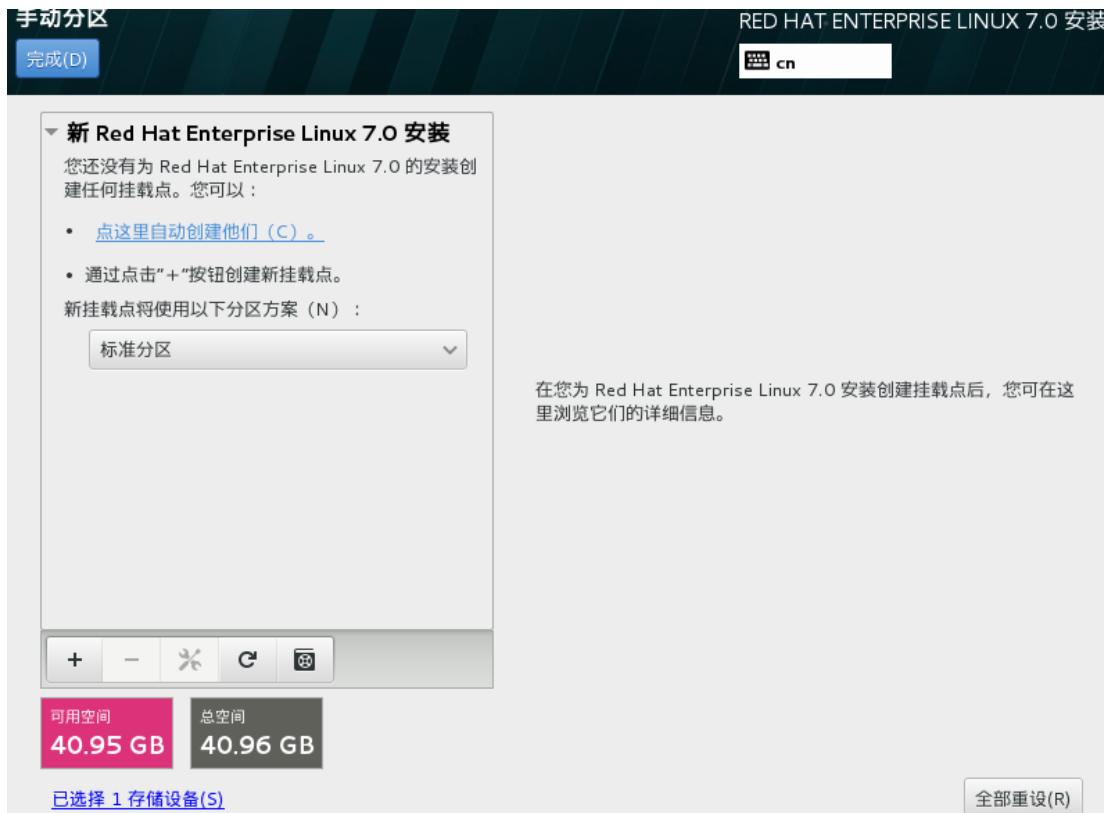
4. 单击软件选择带 GUI 服务器



5. 点击安装位置，会出现如下对话框：

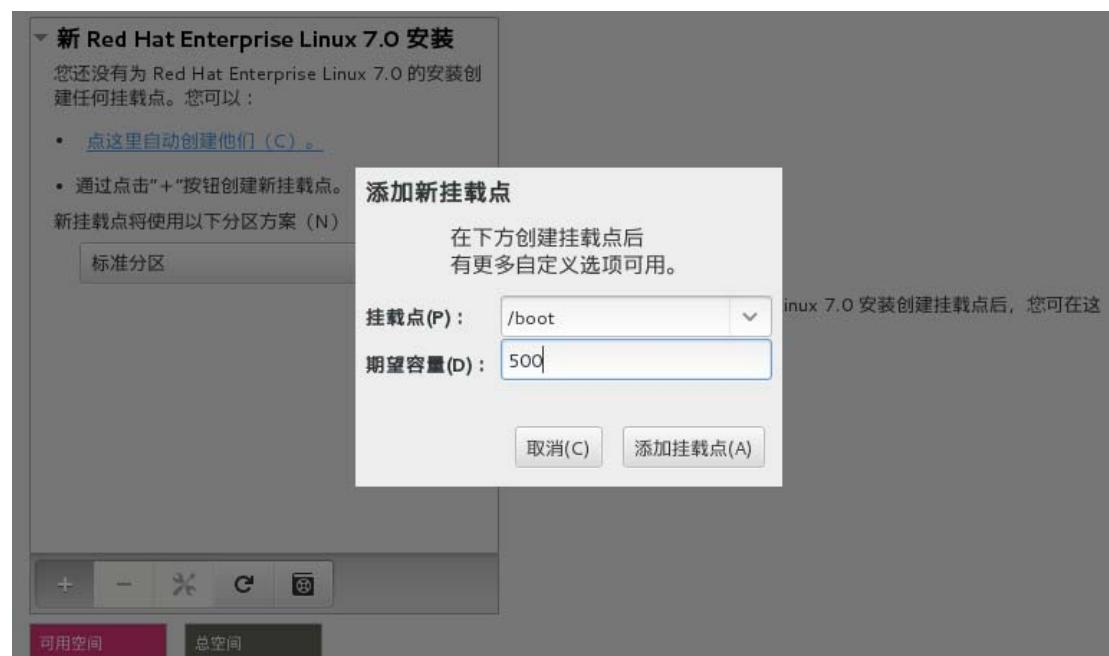


6. 单击“完成”，手动分区，选择标准分区

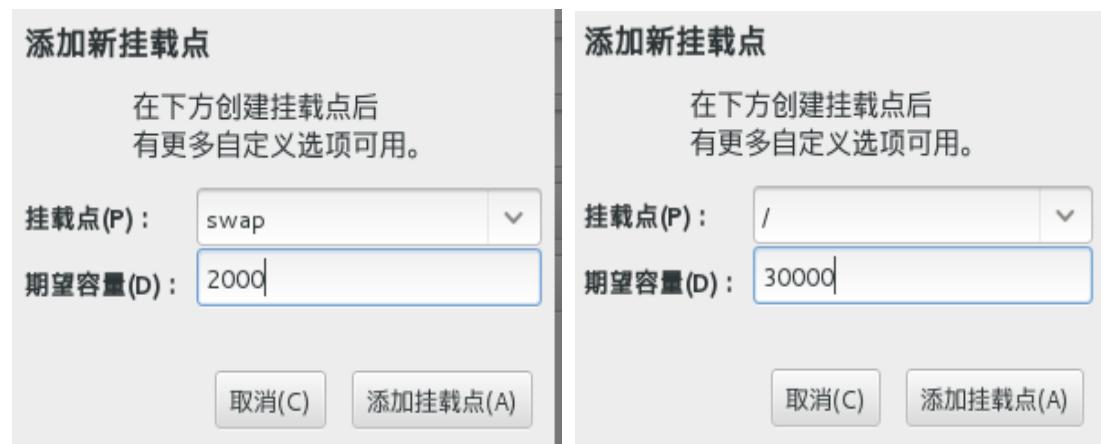


7. 单击“+”，创建自定义分区。

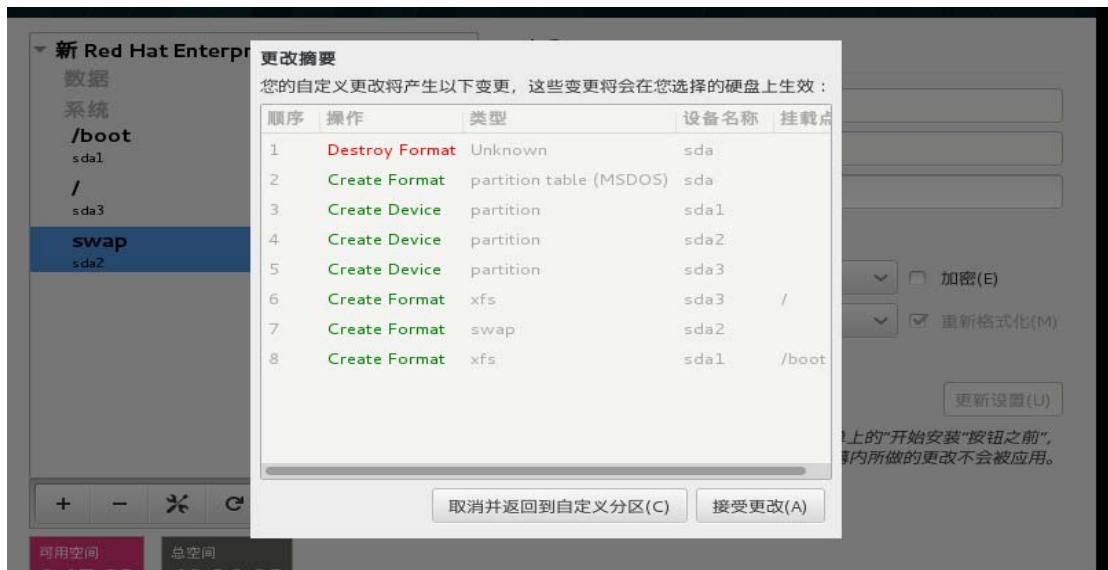
8. 首先创建一个 boot 分区，大小为 500MB。



9. 再创建根分区以及交换分区。文件系统为 xfs。



10. 创建完成后，单击“完成”，弹出“更改摘要”，选择“接受更改”。



11. 紧接着就是安装画面。安装的同时也可以设置 root 密码以及创建用户。





13. 安装过程结束后，点击“重启”。同意许可协议后出现登录界面。



第一章 访问命令行

1.1 使用控制台访问命令行

1、Shell

这是一类程序的总称，这类程序充当命令解释器，它提供了命令的输入行，提示符，我们的操作环境等。Red Hat Linux 默认的 shell 是 Bash，当你输入完用户名及密码后，login 程序将为你打开以下提示符，这是 bash 在运行。你可以在提示符下输入命令，bash 将为你解释执行。



root 用户的提示符是 # 其他非 root 用户默认为 \$

如果默认安装将出现图形登陆窗口，这时可以按 Ctrl+Alt+F2 – F6 来切换至六个虚拟控制台中的一个，你将看到控制台的 login 登陆文字，提示你输入用户名和密码，这时你可以输入安装过程中的 root 的密码来进行登陆，将出现命令提示符：

```
Red Hat Enterprise Linux Server release 6.1 (Santiago)
Kernel 2.6.32-131.0.15.el6.i686 on an i686

localhost login: root
Password:
Last login: Tue Jan  3 01:54:08 on tty2
[root@localhost ~]#
```

从虚拟控制台返回图型界面可以按 Alt+F1

从虚拟控制台之间的切换可以按 Alt+Fn

Ctrl+D 或 exit 命令关闭当前的 terminal 窗口

Ctrl+Shift+t 新建 terminal 窗口

2 Linux 命令概述

命令是代表实现某一特定功能的指令或程序。

Shell 是操作系统平台和用户进行交互的界面。Linux 用户在这个界面中输入 Linux 命令，然后由 Shell 对命令进行分析后，将命令请求交给适当的后台程序处理，等待处理结果，并通过 Shell 界面将处理结果返回给用户。

1) 命令规则

在 Shell 中，命令有一定格式和规则，其通常的命令格式是“命令+选项+参数”，例如：把 /etc 目录下所有打包成 bk.tar 文件。



- (1) 命令选项和参数都区分大小，这和 Windows 中不同。
- (2) 选项是调节命令的具体功能，以减号 (-) 开始，多个选项能够用一个减号 (-) 连起来，如 ls -l -a 和 ls -la 相同。
- (3) 参数是命令执行处理对象，可能是命令执行过程中所使用的文件、目录或用户

名等内容。

- (4) 使用分号 ; 能够将两个命令隔开, 这样能够实现一行中输入多个命令。命令的执行顺序和输入的顺序相同。
- (5) Tab 键: 命令补全, 在路由器配置中很用得很熟了。可以按两次 Tab 键系统可以列出可用名称列表。

2) Linux 命令帮助

因为 Linux 下命令众多, 没有人记得住所有的命令和选项, 所以对于 linux 的学习最重要的是先学会使用帮助,

(1) help 命令

查看内部命令的帮助信息。、

例如: 用 help 查看内部命 cd 帮助信息

```
[root@localhost ~]#
[root@localhost ~]# help cd
cd: cd [-L|-P] [dir]
      Change the shell working directory.

      Change the current directory to DIR. The default DIR is the value of the
      HOME shell variable.

      The variable CDPATH defines the search path for the directory containing
      DIR. Alternative directory names in CDPATH are separated by a colon (:).
      A null directory name is the same as the current directory. If DIR begins
      with a slash (/), then CDPATH is not used.
```

(2) 使用 “ --help ” 选项

例如: 用 help 查看命令 ls 帮助信息

```
[root@localhost ~]#
[root@localhost ~]# ls --help
用法 : ls [选项1...][文件1...]
```

列出 FILE 的信息(默认为当前目录)。

如果不指定 -cftuvSUX 或 --sort 选项, 则根据字母大小排序。

长选项必须使用的参数对于短选项时也是必需使用的。

-a, --all	不隐藏任何以 . 开始的项目
-A, --almost-all	列出除 . 及 .. 以外的任何项目
--author	与 -l 同时使用时列出每个文件的作者
-b, --escape	以八进制溢出序列表示不可打印的字
--block-size=大小	块以指定大小的字节为单位
-B, --ignore-backups	不列出任何以 "~" 字符结束的项目
-c	配合 -lt: 根据 ctime 排序并显示 cti 状态最后更改的时间)

[] : 表示可选项

... : 表示可以有多个项

| : 表示只能有一个项

<> : 表示可变的数据, 有是简写成大写字符。例: <filename> 表示插入文件名

1.2 使用桌面访问命令行

GNOME 桌面环境

桌面环境是 Linux 系统的图形用户界面。默认的桌面环境在红帽企业 Linux 7 由 GNOME 3 提供，它为用户和统一的开发平台上的 X Window 系统提供的图形框架顶部的一体台式机。

在 GNOME Shell 提供了核心的用户界面功能的 GNOME 桌面环境。在 gnome-外壳应用程序是高度可定制的。默认情况下，RHEL 7 的用户使用“GNOME 经典”为主题的 gnome-shell，它类似于 GNOME 2 桌面环境。另一个可用的选项是“现代”的 GNOME 3 主题所使用的上游 GNOME 项目。主题可以通过输入用户的密码时，选择旁边的标志的齿轮图标按钮中选择。

第一次新用户登录时，初始安装程序运行，帮助他们配置基本帐户设置。GNOME 的帮助应用程序，然后开始在入门 GNOME 屏幕。此屏幕包含视频和文档，帮助新用户到 GNOME 3 的环境。GNOME 的帮助可以在 GNOME-shell 中键入 F1 可以很快上手，通过选择应用程序→文件→帮助，或通过运行 yelp 命令。

1.3 使用 bash shell 执行命令

```
[student@desktopX ~]$ date
Sat Apr  5 08:13:50 PDT 2014
[student@desktopX ~]$ date +%R
08:13
[student@desktopX ~]$ date +%x
04/05/2014
```

```
[student@desktopX ~]$ passwd
Changing password for user student.
Changing password for student.
(current) UNIX password: old_password
New password: new_password
Retype new password: new_password
passwd: all authentication tokens updated successfully.
```

Tab 键

Tab 键使用户能够快速完成命令输入，一旦他们输入了足够的提示完整的命令或文件名，使其独一无二。如果输入的字符不唯一，按两次 Tab 键显示开始已经输入的字符的所有命令。

```
[student@desktopX ~]$ pas<Tab><Tab>
passwd      paste      pasuspender
[student@desktopX ~]$ pass<Tab>
[student@desktopX ~]$ passwd
Changing password for user student.
Changing password for student.
(current) UNIX password:
```

```
[student@desktopX ~]$ ls /etc/pas<Tab>
[student@desktopX ~]$ ls /etc/passwd<Tab>
passwd  passwd-
```

```
[root@desktopX ~]# useradd --<Tab><Tab>
--base-dir      --groups          --no-log-init    --shell
--comment       --help            --non-unique     --skel
--create-home   --home-dir       --no-user-group  --system
--defaults      --inactive       --password      --uid
--expiredate   --key            --root          --user-group
--gid           --no-create-home --selinux-user
[root@desktopX ~]# useradd --
```

history 命令

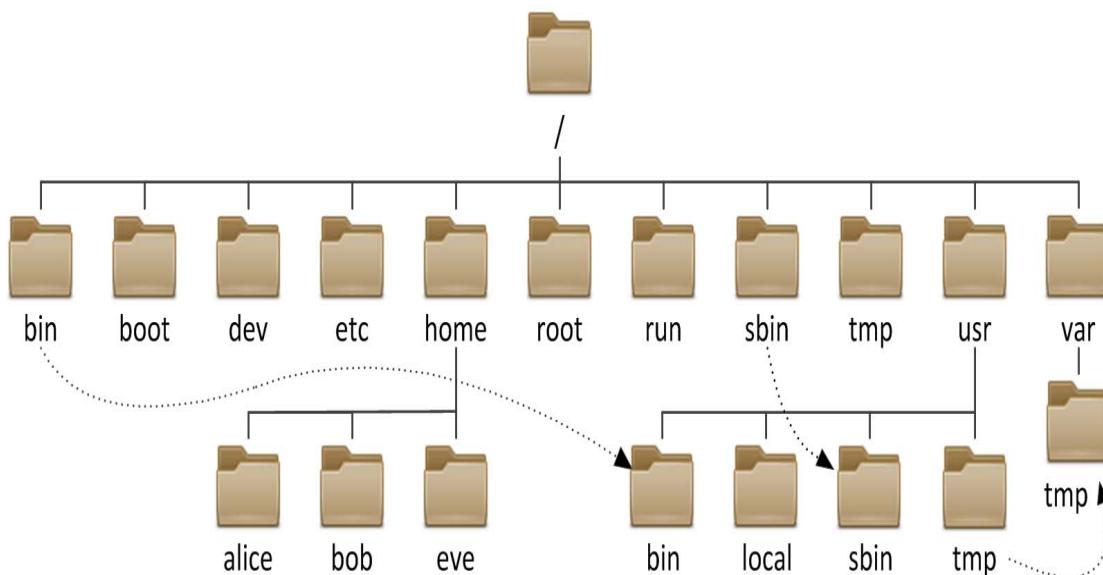
```
[student@desktopX ~]$ history
...Output omitted...
23 clear
24 who
25 pwd
26 ls /etc
27 uptime
28 ls -l
29 date
30 history
[student@desktopX ~]$ !ls
ls -l
total 0
drwxr-xr-x. 2 student student 6 Mar 29 21:16 Desktop
...Output omitted...
[student@desktopX ~]$ !26
ls /etc
abrt                  hosts          pulse
adjtime               hosts.allow    purple
aliases               hosts.deny    qemu-ga
...Output omitted...
```

命令行快捷键

- Ctrl+a 跳到命令行开始位置
- Ctrl+e 跳到命令行结束位置
- Ctrl+u 删除光标前字符
- Ctrl+k 删除光标后字符
- Ctrl+左方向键 跳到前一个单词
- Ctrl+右方向键 跳到后一单词
- Ctrl+r 搜索历史记录
- Ctrl+l 清屏

第二章 从命令行管理文件

2.1 Linux 文件系统结构



/usr 安装软件，共享库，包括文件，和静态只读数据的程序。重要的子目录，包括：

- /usr/bin: 用户命令
- /usr/sbin: 系统管理命令
- /usr/local: 局部定制软件

/etc 配置文件

/var 如数据库缓存目录日志文件，打印假脱机文件，和网站内容会根据应用发生变化
 /run 进程的运行时数据开始自上次启动。这包括过程 ID 文件和文件锁，在其他的事情。

(新增目录，解决 dev 目录使用混乱问题)

/home 普通用户存储他们的个人数据和配置文件

/root 超级用户的家目录

/tmp 临时文件目录，具备 10 天以上的文件将被删除的特性

/boot 启动所需要的文件目录

/dev 设备文件与文件系统

/proc Kernel 进程与配置交互目录

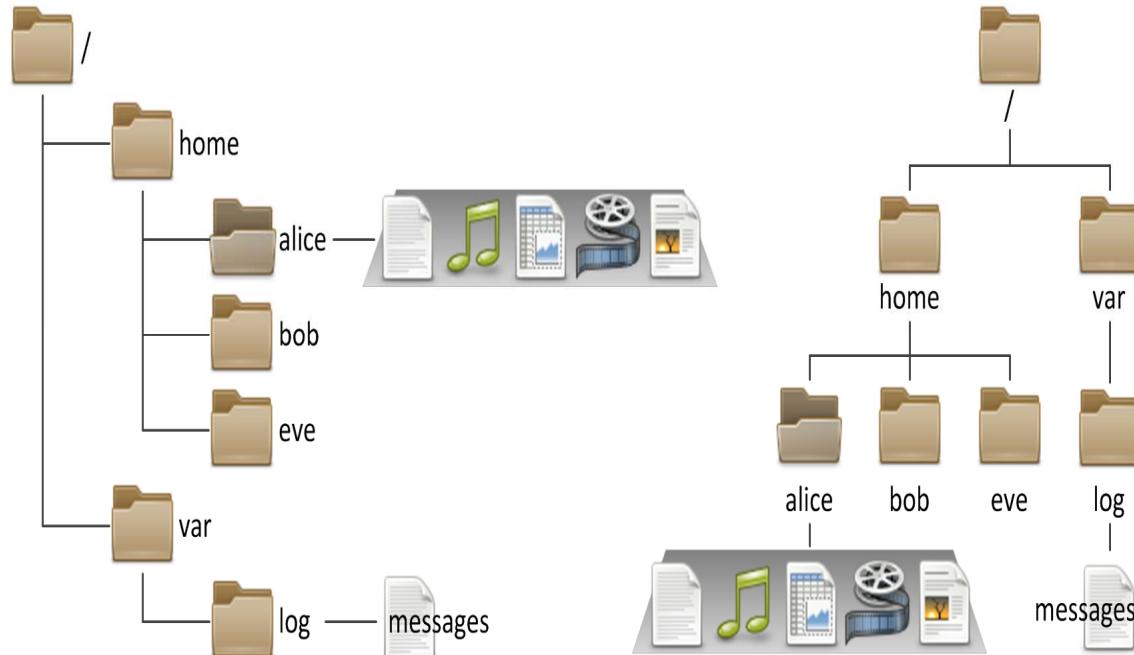
以下目录是通过软链接建立相同目录

/bin	/usr/bin
/sbin	/usr/sbin
/lib	/usr/lib
/lib64	/usr/lib64

2.2 通过文件名定位文件

1、路径

引入路径概念目的最终是找到我们所需要的目录或文件。比如我们想要编辑 file.txt 文件，我们首先要知道他存放在哪里，也就是说我们要指出他所在的位置，这时就要用到路径了。



路径分为绝对路径和相对路径；

➤ 绝对路径

在 Linux 中，绝对路径是从/（也被称为根目录）开始的，比如/usr、/etc/passwd。如果一个路径是从/开始的，它一定是绝对路径，这样就好理解了；

➤ 相对路径

相对路径是以 . 或 .. 开始的，. 表示用户当前操作所处的位置，而.. 表示上级目录；在路径中，. 表示用户当前所处的目录，而.. 上级目录，要把. 和.. 当做目录来看。

2、导航路径

1) pwd —— 查看当前的工作目录 (Print Working Directory)

用于显示当前用户所在工作目录（绝对路径）。

```
[root@localhost ~]# pwd
/root
[root@localhost ~]# █
```

2) cd —— 切换工作目录 (Change Directory)

cd [绝对路径 / 相对路径]

可以用来在不同目录中切换。相当于 dos 中的 cd。

例如：

# cd /home	进入/home 目录
# cd /var/www	进入/var/www 目录, 绝对路径
# cd ..	(在 cd 和.. 之间有一个空格) 可以回到上一层目录。
# cd ../home	进入上一级目录下的 home 目录
# cd	返回当前用户的家目录
# cd ~lipi	到 lipi 的主目录
# cd -	回到刚才的目录

3) ls —— (List) 显示目录内容

ls 可以用来查看一个目录内有什么文件(子目录)。相当于 dos 中的 dir。

ls 命令的常用选项说明：

-a	列出目录下的所有文件, 包括以 . 开头的隐含文件。
-l	列出文件的详细信息。
-R	(递归) 同时列出所有子目录内存
-d	只列出目录名; 不列出它的内容。

ls 命令使用举例：

例 1：显示/root 目录中中包含的子目录和文件系息。

```
[root@zx 桌面]#
[root@zx 桌面]# ls /root
anaconda-ks.cfg  install.log.syslog  模板  图片  下载  桌面
install.log      公共的                  视频  文档  音乐
[root@zx 桌面]# cd /root
[root@zx ~]# ls
anaconda-ks.cfg  install.log.syslog  模板  图片  下载  桌面
install.log      公共的                  视频  文档  音乐
[root@zx ~]# ■
```

例 2：以长格式形式显示当前目录的下所有内容 (包含隐藏文件)。

```
[root@zx ~]#
[root@zx ~]# ls -la
总用量 228
dr-xr-x---. 24 root root 4096 1月 4 01:04 .
dr-xr-xr-x. 25 root root 4096 1月 4 00:59 ..
-rw-----. 1 root root 2036 1月 4 2012 anaconda-ks.cfg
-rw-----. 1 root root 268 1月 4 00:29 .bash_history
-rw-r--r--. 1 root root 18 5月 20 2009 .bash_logout
-rw-r--r--. 1 root root 176 5月 20 2009 .bash_profile
-rw-r--r--. 1 root root 176 9月 23 2004 .bashrc
drwxr-xr-x. 3 root root 4096 1月 5 2012 .cache
drwxr-xr-x. 5 root root 4096 1月 4 2012 .config
-rw-r--r--. 1 root root 100 9月 23 2004 .cshrc
drwx-----. 3 root root 4096 1月 4 2012 .dbus
```

例 3：显示/etc 目录下所有以 “.conf” 结尾的文件。

```
[root@zx ~]# ls /etc/*.conf
/etc/ant.conf           /etc/nscd.conf
/etc/asound.conf         /etc/nslcd.conf
/etc/autofs_ldap_auth.conf /etc/nsswitch.conf
/etc/cas.conf            /etc/ntp.conf
/etc/cgconfig.conf       /etc/oddjobd.conf
/etc/cgrules.conf        /etc/pam_ldap.conf
/etc/dnsmasq.conf        /etc/pbm2ppa.conf
```

2.3 通过命令行工具管理文件

1、touch——新建空文件

touch 可以用来创建一个空文件，但当 touch 的文件已存在时，touch 会将当前的系统时钟赋予该文件。

例 1：在当前目录下创建文件名为 file1 的文件。

2、file——查看文件类型

```
[root@zx ~]# pwd
/root
[root@zx ~]# touch file1
[root@zx ~]# touch file2
[root@zx ~]# ls
anaconda-ks.cfg  install.log      zx      视频  下载
file1             install.log.syslog 公共的  图片  音乐
file2             soft              模板   文档  桌面
[root@zx ~]#
```

在 Windows 系统，文件扩展名代表文件类型，在 Linux 系统，文件扩展名没有实际意义，用 file 命令可以查看文件类型。

例 1：查看 ls 命令程序的文件类型。(32 位的可执行文件)

```
[root@zx ~]# file /bin/ls
/bin/ls: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically
inked (uses shared libs), for GNU/Linux 2.6.18, stripped
[root@zx ~]#
```

例 2：查看/root/install.log 文件类型。(文本文件)

```
[root@zx ~]#
[root@zx ~]# file /root/install.log
/root/install.log: UTF-8 Unicode text
[root@zx ~]#
```

3、cp——复制 (copy) 文件或目录

cp [选项] [源文件] [目标文件]。

常用选项：

- a 尽可能将文件状态、权限等资料都照原状予以复制。
- r 递归复制所有文件及子目录，复制目录时必须用些选项。
- f 若目的地已经有相同档名的档案存在，则在复制前先予以删除再行复制。

例 1：将文件 file1 复制到同一目录下，名字为 mydoc

```
[root@zx ~]# ls
anaconda-ks.cfg  install.log      zx       视频  下载
file1            install.log.syslog 公共的   图片  音乐
file2            soft             模板   文档  桌面
[root@zx ~]# cp file1 mydoc
[root@zx ~]# ls
anaconda-ks.cfg  install.log      soft     模板  文档  桌面
file1            install.log.syslog zx      视频  下载
file2            mydoc           公共的   图片  音乐
[root@zx ~]#
```

例 2：将文件 file1 复制到目录 /root/soft 目录下，文件名称 readme.txt

```
[root@zx ~]# cp file1 /root/soft/readme.txt
[root@zx ~]# ls /root/soft
readme.txt
[root@zx ~]#
```

例 3：复制 /root/soft 目录及目录内文件到 /root/zx/network/c1 目录中。

```
[root@zx ~]#
[root@zx ~]# cp /root/soft/ /root/zx/network/c1 ← 不成功
cp: 略过目录 "/root/soft/"
[root@zx ~]# cp /root/soft/ -r /root/zx/network/c1
[root@zx ~]# ls /root/zx/network/c1
readme.txt  soft ← 成功了
[root@zx ~]#
```

4、rm——删除（Remove）文件或目录

常用选项：

- i 删之前逐一询问确认（默认）。
- f 即使原文件属性设为只读，亦直接删除，无需逐一确认。
- r 递归删除整个目录树，删除目录时用到此选项。

使用 rm 命令要小心。因为一旦文件被删除，它是不能被恢复的。为了防止这种情况的发生，可以使用 i 选项来逐个确认要删除的文件。

例 1：删除当前目录下文件名为 file1 的文件

```
[root@zx ~]#
[root@zx ~]# rm file1
rm: 是否删除普通空文件 "file1" ? y
[root@zx ~]#
```

例 2：删除 /root/soft 目录中文件名以 r 开头的文件。

```
[root@zx ~]# rm /root/soft/r*
rm: 是否删除普通空文件 "/root/soft/readme.txt" ? y
[root@zx ~]#
```

例行 3、直接删除 zx 目录及其子目录和文件，不需要提示。

```
[root@zx ~]# ls /root/
anaconda-ks.cfg  install.log      mydoc  公共的  视频  文档  音乐
file2            install.log.syslog zx      模板  图片  下载  桌面
[root@zx ~]# rm -rf /root/zx
[root@zx ~]# ls
anaconda-ks.cfg  install.log      mydoc  模板  图片  下载  桌面
file2            install.log.syslog 公共的  视频  文档  音乐
[root@zx ~]# 
```

5、mv——移动（Move）文件目录或改名

用于将指定的文件或目录转移位置，如果目标位和源位置相同，同起到改名的作用。

例 1：将上例中 mydoc 文件改名为 a.txt。

```
Mv mydoc a.txt
```

例 2：将上例中在/root 目录中创建的目录 soft 移动到/tmp 目录中。

```
Mv soft /tmp/
```

6、rmdir——删除空目录

rmdir 只能删除空目。

文件内容操作命令

1、echo—— shell 中输出内容的命令

```
[root@zx ~]# echo zxccie
zxccie
[root@zx ~]#
echo $PATH    显示变量 PATH 中的内容
[root@zx ~]# echo $PATH
/usr/lib/qt-3.3/bin:/usr/local/sbin:/usr/sbin:/sbin:,
:/root/bin:/root/bin
[root@zx ~]# 
```

2、cat——显示或连接（concatenate）文件内容

它的作用其实是连接文件。但默认情况下它会将连接文件的结果送到标准输出。所以我们常用来显示文件内容。类似于 dos 中的 type。

例 1：查看/etc/sysconfig/network-scripts/ifcfg-eth0 网卡配置文件信息。

```
[root@zx ~]#
[root@zx ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
HWADDR="00:0C:29:D9:7E:E9"
NM_CONTROLLED="yes"
ONBOOT="no"
[root@zx ~]# 
```

3、more 和 less——分页查看文件内容

如果文件内容比较长，在一个屏幕内无法显示完，那么使用 cat 命令就可能无法看

全, 这里我们可以使用 more 或 less 命令即可。

- More 命令一次显示一屏内容, 用回车可以逐行查看, 用空格可以往下翻页, 按 b 往前翻。按 Q 或 q 退出。

例 1: 用 more 命令查看/root/install.log 文件内容。

- Less 命令的功能和 more 类似, 也是按页显示文件, 不同之处是 less 允许用户即可向前翻 (按 b 键) 又可以向后 (按空格或 d 键) 翻页。另外 less 还支持文件内容快速查找。先再按 “/”, 再输入要查找的字符, 直接跳转到查找到第一个相匹配的内容, 按 Q 或 q 退出。

例 2: 用 less 命令查看/root/install.log 文件内容。

4、head 和 tail——查看文件头或末尾部分内容

- a) head 命令可以仅查看文件前几行, 默认查看 10 行。

例 1: 查看 install.log 文件的前三行。

```
head -3 /root/install.log
```

- b) tail 命令与 head 相对应, tail 命令可以仅查看文件后几行, 默认查看 10 行。多用于查看系统的日志文件。

5、wc——统计文件内容中单词数量 (Word Count) 等信息。

Wc 命令用于统计文件内容中包含的行数、单词数、字节数等信息, 以文件名做为参数。

```
[root@zx 桌面]# wc /root/install.log
1518 3061 59017 /root/install.log
[rot@zx 桌面]#
    行数      单词数      字节数
```

6、grep——检索、过滤文件内容

Grep 命令用于在文件中查找并显示包含指定的字符串行。

命令格式:

```
Grep [选项] 查找条件 目标文件
```

常用选项如下:

- -i 查找内容时忽略大小写
- -v 查找不符合条件的行

grep 命令举例。

例 1: 在/etc/passwd 文件中查找包含 “root” 字符串的行。

```
[root@zx ~]# grep root /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
[root@zx ~]#
```

例 2: 在/etc/passwd 文件中查找 li 开头的行。

```
[root@zx ~]# grep '^li' /etc/passwd
lipi:x:500:500:lipi296:/home/lipi:/bin/bash
[root@zx ~]#
```

例 3：在/etc/passwd 文件中查找除了包含“nologin”的行。

```
[root@zx ~]# grep -v 'nologin' /etc/passwd
root:x:0:0:root:/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
lipi:x:500:500:lipi296:/home/lipi:/bin/bash
[root@zx ~]#
```

例 4：用 cat 命令查看/etc/passwd 文件内容，并过滤掉包含“nologin”的行。

```
[root@zx ~]# cat /etc/passwd | grep -v nologin
root:x:0:0:root:/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
lipi:x:500:500:lipi296:/home/lipi:/bin/bash
[root@zx ~]#
```

7、du——统计目录及文件的空间占用情况

Du 命令可用天统计指定目录（或文件）所占磁盘空间的大小。使用目录或文件名作为参数。常用选项如下：

- -a 统计所有目录及文件
- -h 以更人性化方式显示结果
- -s 只统计每个参数，而不统计每个子目录和文件，不能和-a一起使用。

例 1：统计/home 目录中各文件占用磁盘空间。

```
[root@zx ~]# du -ah /home
4.0K  /home/lipi/.emacs
4.0K  /home/lipi/.gnome2
4.0K  /home/lipi/.mozilla/plugins
4.0K  /home/lipi/.mozilla/extensions
12K   /home/lipi/.mozilla
4.0K  /home/lipi/.bash_logout
4.0K  /home/lipi/.bashrc
4.0K  /home/lipi/.bash_profile
36K   /home/lipi
40K   /home
[root@zx ~]#
```

例 2：统计/home 目录总计占用磁盘空间。

```
[root@zx ~]# du -sh /home
40K   /home
[root@zx ~]#
```

2.4 通过通配符匹配文件

*	0个或多个字符
?	任意单个字符
~	当前用户的家目录
~username	特定用户的家目录
~+	当前工作目录
~-	前一个工作目录
[abc]	括号任意一个字符
[!abc]	不包含括中任一字符
[[:alpah:]]	任一字母
[[:lower:]]	任一小写字母
[[:upper:]]	任一大写字母
[[:alnum:]]	任一字母或数字
[[:punct:]]	除空格、字母、数字以外任一可打印字符
[[:digit:]]	任一数字，0-9

```
[student@desktopX ~]$ mkdir glob; cd glob
[student@desktopX glob]$ touch alfa bravo charlie delta echo able baker cast dog easy
[student@desktopX glob]$ ls
able alfa baker bravo cast charlie delta dog easy echo
[student@desktopX glob]$
```

```
[student@desktopX glob]$ ls a*
able alfa
[student@desktopX glob]$ ls *a*
able alfa baker bravo cast charlie delta easy
[student@desktopX glob]$ ls [ac]*
able alfa cast charlie
[student@desktopX glob]$ ls ???
able alfa cast easy echo
[student@desktopX glob]$ ls ??????
baker bravo delta
[student@desktopX glob]$
```

```
[student@desktopX glob]$ ls ~/glob
able alfa baker bravo cast charlie delta dog easy echo
[student@desktopX glob]$ echo ~/glob
/home/student/glob
[student@desktopX glob]$
```

```
[student@desktopX glob]$ echo {Sunday,Monday,Tuesday,Wednesday}.log
Sunday.log Monday.log Tuesday.log Wednesday.log
[student@desktopX glob]$ echo file{1..3}.txt
file1.txt file2.txt file3.txt
[student@desktopX glob]$ echo file{a..c}.txt
filea.txt fileb.txt filec.txt
[student@desktopX glob]$ echo file{a,b}{1,2}.txt
filea1.txt filea2.txt fileb1.txt fileb2.txt
[student@desktopX glob]$ echo file{a{1,2},b,c}.txt
filea1.txt filea2.txt fileb.txt filec.txt
[student@desktopX glob]$
```

```
[student@desktopX glob]$ echo Today is `date +%A`.
Today is Wednesday.
[student@desktopX glob]$ echo The time is $(date +%M) minutes past $(date +%l%p).
The time is 26 minutes past 11AM.
[student@desktopX glob]$
```

```
[student@desktopX glob]$ host=$(hostname); echo $host
desktopX
[student@desktopX glob]$ echo "***** hostname is ${host} *****"
***** hostname is desktopX *****
[student@desktopX glob]$ echo Your username variable is \$USER.
Your username variable is $USER.
[student@desktopX glob]$
```

```
[student@desktopX glob]$ echo "Will variable $host evaluate to $(hostname)?"
Will variable desktopX evaluate to desktopX?
[student@desktopX glob]$ echo 'Will variable $host evaluate to $(hostname)?'
Will variable $host evaluate to $(hostname)?
[student@desktopX glob]$
```

第三章 红帽企业 Linux 获得帮助

3.1 man

1、man man

- 1 Executable programs or shell commands
- 2 System calls (functions provided by the kernel)
- 3 Library calls (functions within program libraries)
- 4 Special files (usually found in /dev)
- 5 File formats and conventions eg /etc/passwd
- 6 Games
- 7 Miscellaneous (including macro packages and conventions), e.g.
 man(7), groff(7)
- 8 System administration commands (usually only for root)
- 9 Kernel routines [Non standard]

例：查看 passwd 命令帮助

```
# man passwd
```

例：查看/etc/passwd 文件帮助

```
# man 5 passwd
```

2、Man 命令操作：

/string	搜索
n	向下搜索
N	向上搜索

G	man page 开始位置
g	man page 结束位置
q	退出

2、根据一个关键字搜索 man page

```
# man -k passwd
```

3.2 PINFO

使用 PINFO 命令阅读帮助文档。Man 是一个有用的命令参考正式的格式，但不可作为通用的文件。对于这样的文件，GNU 项目开发了一个不同的网络文件系统，称为 GNU info。

```
# man tar  
# pinfo tar
```

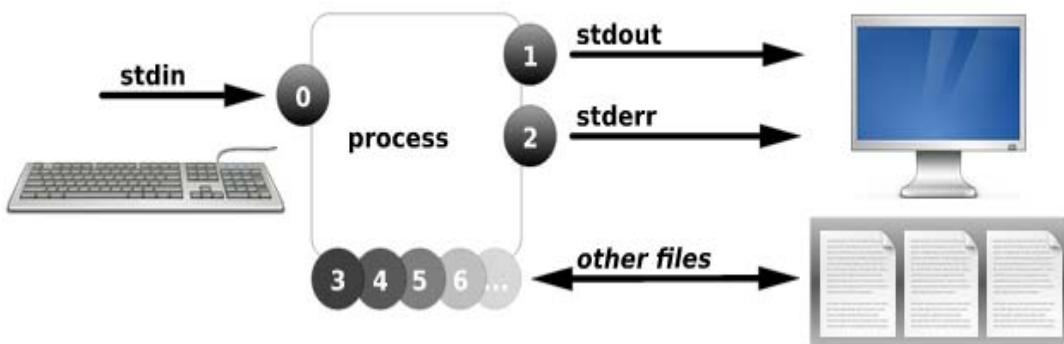
Pinfo 提供可以有菜单，可以选择跳转。

3.3 阅读/ usr /share/ doc 文件

```
[root@localhost ~]# firefox file:///usr/share/doc
```

第四章 建立查看编辑文本文件

4.1 输入输出重定向



Linux 系统使用文件来描述系统的硬件、设备资源，例如已经学习的硬盘分区设备文件、光盘设备文件等。在用户通过操作系统处理信息的过程中，包括以下几类交互式的设备文件。

- 标准输入：默认设备是键盘，文件编号为 0，命令从标准输入文件中读取在执行过程中

需要输入的数据。

- 标准输出：默认设备是显示器，文件编号为 1，命令将执行结果发送到标准输出文件。
- 标准错误：默认设备是显示器，文件编号为 2，命令将执行时错误信息发送到标准错误文件。

在标准输入、输出和标准错误默认使用了键盘和显示器作为关联设备，因此在执行命令时会从键盘接收用户输入的字符，并将结果显示在屏幕上，如果命令执行错误，也将错误信息显示在屏幕上反馈给用户。这样通过最普通的终端设备（键盘和显示器）用户就可以执行 Linux 命令，并完成最基本的输入输出操作。

在实际的 Linux 操用过程中，也可以变更输入输出内容的方向，而不使用默认的标准输入输出设备（键盘和显示器），这种操作称为重定向。

1) 标准输出重定向

标准输出重定向是将命令的输出结果定向（保存）到指定的文件中，而不是直接显示在显示器屏幕上。输出重定向使用 > 或 >> 操作符号，分别用于覆盖、追加文件。

> 重定向符后面指定的文件如果不存在，则会新建该文件，并将结果保存到该文件中；若该文件存在，则会清空文件内容并保存命令的执行结果到文件中。

例：查看 2012 年 2 月日历，并将输出结果保存到文件 cal.txt 中

```
[root@zx ~]# cal 2 2012 > cal.txt
[root@zx ~]# cat cal.txt
    二月 2012
日 一 二 三 四 五 六
      1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29

[root@zx ~]#
```

>> 重定向符可以将命令执行的结果追加到指定文件的末尾进行保存，而不覆盖文件原有内容。

例：查看 2012 年 3 月日历，，并将输出结果追加到上例中的 cal.txt 文件中。

```
[root@zx ~]# cal 3 2012 >> cal.txt
[root@zx ~]# cat cal.txt
    二月 2012
日 一 二 三 四 五 六
      1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29

    三月 2012
日 一 二 三 四 五 六
      1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31

[root@zx ~]#
```

2) 标准输入重定向

标准输入重定向就是将命令中接收输入的途径由默认键盘更改（重定向）为指定的文件。使用输入重定向将使一些交互式的操作可以通过读取文件来完成。标准输入重定向需要使用 < 操作符。

例：用 wc 统计/etc/passwd 文件中用户数，通过输入重定向把/etc/passwd 文件传给 wc 命令。

```
[root@zx ~]# wc -l < /etc/passwd
43
[root@zx ~]#
```

3) 标准错误重定向

标准错误重定向就是将执行命令过程中出现的错误信息重定向保存到指定的文件，而不是直接在屏幕上显示。**错误重定向需要使用 2> 操作符**，其中 2 是错误文件的编号（在标准输出、标准输入重定向时实际上省略了 0、1 编号）。**2>>表示追加到文件。**

例：使用 help 命令查看 ls 命令的帮助信息时，由于 ls 并非内部命令，系统会报错，通过 2> 操作符可以将报错信息重定向到指定文件 err.log 中。

```
[root@zx ~]# help ls
bash: help: no help topics match `ls'. Try `help help' or `man -ls'.
[root@zx ~]# help ls 2> err.log
[root@zx ~]# cat err.log
bash: help: no help topics match `ls'. Try `help help' or `man -ls'.
[root@zx ~]#
```

在实际应用中，错误重定向主要提供以下两个用途。

- 在调试程序过程中，收集程序的错误信息，为程序排错提供依据。
- 使用 Shell 脚本程序时，将错误重定向到指定文件中，以保持用户显示界面的整洁。

当命令执行结果中同时包含标准输出（正常执行）和错误输出的内容时，可以同时使用操作符 `>`、`2>` 将信息分别重定向到不同的文件中，也可以使用 `&>` 操作符将两种输出内容重定向到同一文件中。

例：使用 `ls` 命令查看 `/etc/passwd` 和 `/etc/password`（并不存在）文件属性时，将输出结果中标准输出重定向到 `std.txt` 文件中，将输出结果中错误输出重定向到 `err.log` 文件中。

```
[root@zx ~]# ls -l /etc/passwd /etc/password > std.txt 2> err.log
[root@zx ~]# cat std.txt
-rw-r--r--. 1 root root 2121 1月 31 01:18 /etc/passwd
[root@zx ~]# cat err.log
ls: 无法访问 /etc/password: 没有那个文件或目录
[root@zx ~]#
```

例：将上例的所有输出（标准和错误）重定向到 `out.log` 文件中。

```
[root@zx ~]# ls -l /etc/passwd /etc/password &> out.log
[root@zx ~]# cat out.log
ls: 无法访问 /etc/password: 没有那个文件或目录
-rw-r--r--. 1 root root 2121 1月 31 01:18 /etc/passwd
[root@zx ~]#
```

3、管道操作

管道可以把一系列命令连接起来，这意味着第一个命令的输出会作为第二个命令的输入通过管道传给第二个命令，第二个命令的输出又会作为第三个命令的输入，以此类推。显示在屏幕上的是管道行中最后一个命令的输出（如果命令行中未使用输出重定向）。通过使用管道符“|”来建立一个管道行。

4.2 文本处理工具结合重定向和管道应用

1、grep 命令

选项	说 明：
<code>-i</code>	不区分大小写搜索
<code>-n</code>	返回包含行号
<code>-v</code>	返回不包含模式的行
<code>-Ax</code>	匹配关键字的后 x 行 如： <code># grep -A 5 ftp /etc/passwd</code>
<code>-Bx</code>	匹配关键字的前 x 行
<code>-r</code>	递归式搜索，从当前目录开始

-c	匹配行的统计
--color	--color=auto 如: # date --help grep --color year
-l	列出至少有一行包含模式的文件的名称
^abc	以 abc 开头的行
abc\$	以 abc 结尾的行

例：过滤出/etc/inittab 文件中不以“#”开头行，并再次过滤出非空行，将结果重定向保存为 inittab.txt 文件。

```
[root@zx ~]# grep -v "^#" /etc/inittab | grep -v "^\$" > inittab.txt
[root@zx ~]# cat inittab.txt
id:5:initdefault:
[root@zx ~]#
```

2、cut 命令----用于“剪切”文件中的列

选项	说 明:
-d	指定分隔符 (Tab 是默认值)
-f	与-d 一起使用，指定显示哪个区域
-c	以字符为单位进行分割
-b	以字节为单位进行分割

例：找出系统中使作 Bash 作为登录 Shell、名称以“li”开头的用户信息，并只显示用户名、登录 Shell 两个字段的内容。

```
[root@zx ~]# grep "bash" /etc/passwd | grep "^li"
lipi:x:500:500:lipi296:/home/lipi:/bin/bash
[root@zx ~]# grep "bash" /etc/passwd | grep "^li" | cut -d ":" -f 1,7
lipi:/bin/bash
[root@zx ~]#
```

上例中，cut 命令用于分割通过管道输入的每一行内容，结合“-d”选项使用分号“：“作分隔符，结合“-f”选项指定只显示出分割的第 1、7 两个字段的区域。

3、awk 命令----用于“剪切”文件中的列

例：查看当前系数的总存空间、剩余可用的内存空间，去除其他无关信息。

```
[root@zx ~]# free -m
      total        used        free      shared      buffers      cached
Mem:       659         613         46          0         61        340
-/+ buffers/cache:     210        448
Swap:      499          0        499
[root@zx ~]# free -m | grep "Mem"
Mem:       659         613         46          0         61        340
[root@zx ~]# free -m | grep "Mem" | awk '{print $2,$4}'
659 45
[root@zx ~]#
```

上例中，free 命令的“-m”选项用于以 MB 为单位显示信息。awk 命令用于以空格或制表位作为分隔，输出指定区域的字段数据，例如，'{print \$2,\$4}'表示只输出第 2、4 两个字段的数据内容。

4、tee 命令

功能：从标准输入设备读取数据，将其内容输出到标准输出设备，同时输出保存成文件。

语法：tee [选项] [文件]
-a 或--append 附加到既有文件的后面，而非覆盖它。

例：查看 eth0 配置信息，并把结果写到一个文件中。

```
[root@desktop36 tmp]# ifconfig eth0 | tee eth0.txt
eth0      Link encap:Ethernet HWaddr 00:0C:29:99:25:54
          inet6 addr: fe80::20c:29ff:fe99:2554/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:269 errors:0 dropped:0 overruns:0 frame:0
            TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:33252 (32.4 KiB)   TX bytes:468 (468.0 b)

[root@desktop36 tmp]# ll
total 4
-rw-r--r--. 1 root root 407 Mar 12 16:14 eth0.txt
srwxrwxrwx. 1 root root 0 Mar 12 16:14 fcoemon.dcbd.1283
[root@desktop36 tmp]# cat eth0.txt
eth0      Link encap:Ethernet HWaddr 00:0C:29:99:25:54
          inet6 addr: fe80::20c:29ff:fe99:2554/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:269 errors:0 dropped:0 overruns:0 frame:0
            TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:33252 (32.4 KiB)   TX bytes:468 (468.0 b)
```

5、tr 命令

tr 用来从标准输入中通过替换或删除操作进行字符转换。使用 tr 时要转换两个字符串：字符串 1 用于查询，字符串 2 用于处理各种转换。tr 刚执行时，字符串 1 中的字符被映射到字符串 2 中的字符，然后转换操作开始。

1、最常用选项的命令格式为：

```
[root@zx]# tr -cds ["原字符串 1"] ["改新的字符串 2"] < o_file
-c 用字符串 1 中字符集的补集替换此字符集，要求字符集为 ASCII。
-d 删除字符串 1 中所有输入字符。
-s 删除所有重复出现字符序列，只保留第一个；
o_file 是来源文件。
```

2、字符范围：只能使用单字符或字符串范围或列表。

[a-z] a-z 内的字符组成的字符串。[A-Z] A-Z 内的字符组成的字符串。
[0-9] 数字串。

例：将/root/a.txt 文件小写字母全部替换成大写字母

```
[root@zx~]# tr 'a-z' 'A-Z' < /root/a.txt
```

6、sort 命令----排序

类似数据库中 order by

默认按 ascii 码排序，升序（第一个字符）。通常与管道一起使用。

选项	说 明：
-r	反向排序
-n	按数值而非字符排序
-f	忽略大小写。默认小 a 小于大写 A
-u	除去重复的行。同： sort file.txt uniq -c
-k x	设置排序字段(列)。X 表示列。
-t	指定分隔符 (默认空格)

```
[root@desktop5 tmp]# sort -t: -k3 -n passwd |more
[root@desktop5 tmp]# ps aux |cut -d ' ' -f6 |sort -r |more
[root@desktop5 tmp]# ps aux |cut -d ' ' -f6 |sort -ru |more
```

7、sed 命令----查找并替换。强大的流编辑器。主要是行。

选项	说 明：
s/old/new/	执行字符串替换，将“old”替换为“new”。第一个匹配
s/old/new/g	g 全局
1, 50s/	old/new/g 1-50 行
d	删除匹配的行
i	插入到匹配行的前面

实例

➤ 删除：d 命令

```
$ sed '2d' example-----删除 example 文件的第二行。
```

```
$ sed '2,$d' example-----删除 example 文件的第二行到末尾所有行。
```

```
$ sed '$d' example-----删除 example 文件的最后一行。
```

```
$ sed '/test/d' example-----删除 example 文件所有包含 test 的行。
```

➤ 替换：s 命令

```
$ sed 's/test/mytest/g' example-----在整行范围内把 test 替换为 mytest。如果没有 g 标记，则只有每行第一个匹配的 test 被替换成 mytest。
```

➤ 从文件读入: r 命令

\$ sed '/test/r file' example----file 里的内容被读进来，显示在与 test 匹配的行后面，如果匹配多行，则 file 的内容将显示在所有匹配行的下面。

➤ 写入文件: w 命令

\$ sed -n '/test/w file' example----在 example 中所有包含 test 的行都被写入 file 里。

➤ 插入: i 命令

\$ sed '/test/i\\new line-----', example

如果 test 被匹配，则把反斜杠后面的文本插入到匹配行的前面。

➤ 下一个: n 命令

\$ sed '/test/{ n; s/aa/bb/; }' example

如果 test 被匹配，则移动到匹配行的下一行，替换这一行的 aa，变为 bb，并打印该行，然后继续。

8、mail 命令----收发邮件。

例：常用格式发信

➤ 你可以把当前 shell 当成编辑器来用，编辑完内容后 Ctrl-D 或”。“结束

mail -s test root@instructor.example.com

➤ 管道

echo “This is test mail” | mail -s test liweijie@163.com

➤ 以 file 的内容为邮件内容发信

mail -s test liweijie@163.com < file

➤ 把多行发送给 STDIN

使用“<<终止词”命令从键盘把多行重导向给 STDIN

直到 终止词 位置的所有文本都发送给 STDIN

有时被称为就地文本 (heretext)

\$ mail -s "Please Call" jane@example.com << END

> Hi dear,

> Please give me a call

> lwj

> END

9. diff: 比较 2 个文件的不同。它还可以用于创建补丁文件。

选项	说 明:
----	------

-c	显示上下文周围的行
-u	使用统一输出格式（对于生成补丁文件很有用）
-r	从指定的目录开始对文件执行递归式比较

Linux shell 提示符下文本编辑器

4.1 Linux 的 VI 编辑器

1、关于文本编辑器

文本编辑器有很多，比如图形模式的 gedit、kwrite、OpenOffice，文本模式下的编辑器有 vi、vim (vi 的增强版本) 和 nano vi 和 vim 是我们在 Linux 中最常用的编辑器。我们有必要介绍一下 vi (vim) 最简单的用法，以让 Linux 入门级用户在最短的时间内学会使用它。

2、为什么要学会应用 vi

vi 或 vim 是 Linux 最基本的文本编辑工具，vi 或 vim 虽然没有图形界面编辑器那样点鼠标的简单操作，但 vi 编辑器在系统管理、服务器管理中，永远不是图形界面的编辑器能比的。当您没有安装 X-windows 桌面环境或桌面环境崩溃时，我们仍需要字符模式下的编辑器 vi；vi 或 vim 编辑器在创建和编辑简单文档最高效的工具；

4.2 vi 编辑器的使用方法

4.2.1 vi 编辑器的工作模式

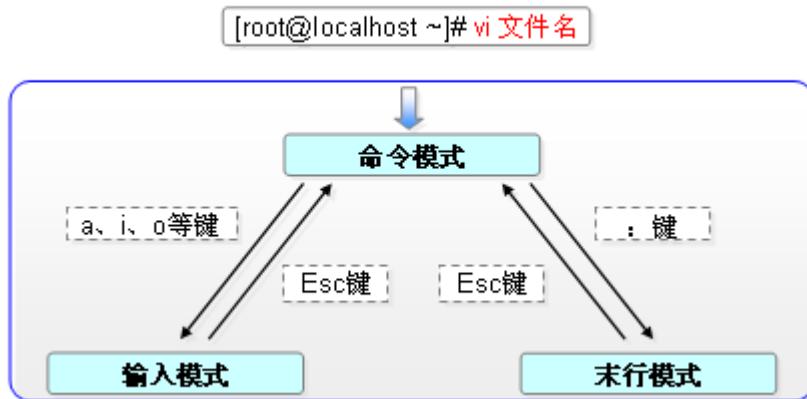
Vi 有三种基本的工作模式：命令模式；文本输入模式；末行模式。

vi 将命令模式和文本输入模式区分开来，这经常被认为是 vi 的一个大问题，但往往这也被认为是 vi 的优势所在。

理解其中的区别是掌握 vi 的关键，vi 启动时，开始处于命令模式。

- 命令模式：我们可以在文件中到处移动，改变文本的某个特定区域、剪切、复制和粘贴文本，还有更多。
- 输入模式：是指用户可以真正输入文本。
- 末行模式：设置 vi 编辑环境、保存文件、退出以及对文件内容进行查找、替换操作。
- 可视模式：便于选取文本

4.2.2 vi 编辑器模式的切换



1、从命令模式与文本输入模式切换：

- i: 光标在当前位置进入文本输入模式
 - I: 光标跳到行首并进入文本输入模式
 - a: 光标后退一格并进入文本输入模式
 - A: 光标退到行尾并进入文本输入模式
 - o: 在光标所在行下新起一行并进入文本输入模式
 - O: 在光标所在行上新起一行并进入文本输入模式
 - s: 删除光标所在字符并进入文本输入模式
 - S: 删除光标所在行并进入文本输入模式
- 按 ESC 键可以返回命令模式。

2、命令行模式与末行模式的切换

在命令模式中，用户按住“：“键可以进入末行模式，等待用户输入命令。多数文件管理命令都是在此模式中之行的。命令执行完毕后，Vi 自动回到命令模式，按 ESC 键可以返回命令模式。

2、命令行模式与可视模式的切换

在命令模式中，用户按住“v”键可以进入可视模式

4.2.3 vi 编辑器命令行模式的基本操作

1、光标移动

- 当我们命令模式后，我们可以用键盘上方向键（h、j、k、l）来移动光标。
- 使用 Page Down 或 Ctrl+F 向下翻页。
- 使用 Page Up 或 Ctrl+B 向上翻页。
- 使用 Home 或“^”、数字 0 可以将光标移到本行的行首。
- 使用 End 或“\$”可以将光标移到本行的行尾。
- 使用 1G 或 gg 可以跳转到文件内容的第一行。
- 使用 G 可以跳转到文件内容的最后一行。
- 使用#G 可以跳转到文件内容的第#行（其中#用具体的数字替换）。

2、文本内容的复制、粘贴和删除操作

Del 或 x	删除一个字符;
#x	删除几个字符, #表示数字, 比如 3x;
dd	删除一行;
#dd	删除多个行, #代表数字, 比如 3dd 表示删除光标行及光标的下两行;
dw	删除一个单词;
#dw	删除几个单词, #用数字表示, 比如 3dw 表示删除三个单词;
d\$	删除光标到行尾的内容;
yy	复制光标所在行
#yy	复制从光标开始的#行
yw	选定光标所在词复制
#yw	复制光标所在位置到之后 #个单词
y\$	复制光标所在位置到行尾的部分
p	贴在光标所在位置之右
P	贴在光标所在位置之左

3、恢复修改及恢复删除操作

- u 撤消最近一次的操作, 可按多次;
- U 撤销当前行所有操作。

4、文件内容查找

在命令模式中按下 “/” 后可输入要查找的字符串, 从光标所在的位置向后查找, 按 n 可以移到下一个查找结果。:set ic 可以忽略大小写, set noic 检查大小写。

4.2.4 vi 编辑器可视模式的基本操作

为了便于选取文本, VIM 引入了可视(Visual)模式。要选取一段文本, 首先将光标移到段首, 在普通模式下按 v 进入可视模式, 然后把光标移到段末。需要注意, 光标所在字符是包含在选区中的。这时可以对所选的文本进行一些操作, 常用的(可视模式)命令有:

x 或 d 剪切(即删除, 同时所选的文本进入剪贴板)

y 复制

当输入了命令以后, VIM 将回到普通模式, 这时可以按 p 或 P 进行粘贴。

4.2.4 vi 编辑器末行模式的基本操作

Vi 编好文件后, 可以在末行模式下按自己的需求键入末行模式的参数。

先键入 “ : ” 进入末行模式,

: q	使用 q 指令退出 vi 编辑器。
: q!	不保存退出 vi 编辑器。
: wq	保存退出。
: w	保存文件。
: w 文件名	另存为。
: e 文件名	打开新文件。
: r 文件名	在当前文件中读入其他文件的内容。

```

: set nu          显示行号
: set nonu        取消显示行号
: sub /i/I        将当前行中的第 1 个字母      “i” 替换为 “I” 。
: 10,20s/default/DEFAULT/g  将 10~20 行中 default 全部替换为 DEFAULT.
: %s/default/DEFAULT/g    将文档中 default 全部替换为 DEFAULT.

```

4.2.5 vi 编辑器高级操作

1、Vim 可以再多分隔窗口环境下编辑多个文件。有两种方法：

在启动 vi 时候使用-o 或者-O 选项，并加上需要同时编辑的文件名。

```
#vi -o file1.txt file2.txt 水平分割窗口编辑 file1.txt 和 file2.txt
#vi -O file1.txt file2.txt 垂直分割窗口编辑 file1.txt 和 file2.txt
```

在不同的窗口间移动，使用 **ctrl+w** 命令。

组合键的列表如下：

ctrl+w+扩大窗口

ctrl+w-缩小窗口

ctrl+w h 移动到窗口左边

ctrl+w j 移动到窗口下边

ctrl+w k 移动到窗口上边

ctrl+w l 移动到窗口右边

ctrl+w ctrl+w 在窗口之间循环移动

2、在 vim 中执行外部命令

输入 :! 然后紧随著输入一个外部命令可以执行该外部命令。

我们以 ls 命令为例。输入 !ls <回车>。该命令就会列举出您当前目录的内容。

例：把系统的日期时间导入光标所在位置

```
:r !date
```

3、在 vim 中定义快捷键

格式： map 快捷键 触发命令

例： 定义 **ctrl+p** 在一行最前面加上#号，不管光标在行中什么位置。

```
:map ^p I#<ESC> -----^p 为 ctrl+v+p 或 ctrl+v ctrl+p
```

例：定义 `ctrl+x` 在一中删除最前面的#号，不管光标在行中什么位置。

```
: map ^x 0x      ----^b 为 ctrl+v+x
```

例：定义 `ctrl+e` 在光标位置插入我的邮箱 `liweijie@163.com`。

```
: map ^e iliweijie@163.com      ----^e 为 ctrl+v+e
```

4、在 vim 中连续行注释

格式：`n1, n2s/^#/g` --[^]代表行首

例：在 10–20 行的行首加上#号注释掉

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

例：在 10–20 行的行首去掉#号

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

5、替换

格式：`ab`

例：输入 `mymail` 替换 `liweijie@sina.com`

```
:ab mymail liweijie@sina.com
```

```
:unab mymail
```

5、保存 VIM 的一些设置，如自动设置行号、快捷键等

```
vim ~/.vimrc
set nu
ab mymail liweijie@sina.com
```

Linux 图型界面下文本编辑器

`gedit`

第五章 管理本地用户和组

5.1 用户和组账号概述

Linux 是一个多用户多任务的分时操作系统，要想进入系统，必须有一个账号。用户的账号一方面可以帮助系统管理员对使用系统的用户进行跟踪，并控制他们对系统资源的访问；另一方面也可以帮助用户组织文件，并为用户提供安全性保护。每个用户账号都拥有一个惟一的用户名和各自的口令。用户在登录时键入正确的用户名和口令后，就能够进入系统。

和自己的主目录。本小节将介绍系统中用户账户和组账户的相关概念。

1、用户账户

在 Linux 系统中，根据系统管理的需要将用户账户分为不同的类型，其拥有的权限、担任的角色也各有不同。主要包括超级用户、普通用户和程序用户

- 超级用户：root 是 Linux 系统中的默认的超级用户，类似于 Windows 系统中 Administrator，对系统管理拥有完全权限。
- 普通用户：是管理员用户创建的，只有特定权限的用户。
- 程序用户：仅用于维护系统或应用程序运行的用户，这些用户一般不允许登录系统。如 ftp、mail 等账号。

2、组账号

组是基于某种联系（如同一部门）的多个用户的集合，对组设置的权限对组内所有用户有效。一个用户至少属于一个组（基本组），还可以加入其他组（附加组）。

3、UID 和 GID

- UID (User Identity, 用户标识号)：作为区分用户的基本依据，原则上用户的 UID 是唯一的。Root 用户的 UID 为 0，程序用户 UID 为 1~499 之间，普通用户 UID 为 500~60000。
- GID (Group Identity, 组标识号)：作为区分组的基本依据，Root 用户的 GID 为 0，程序用户 GID 为 1~499 之间，普通用户 GID 为 500~60000。

5.2 用户账户的管理

Linux 系统中的用户账户、密码等信息均保存在相应的配置文件中，使用用户管理命令直接修改这些配置文件都可以对用户账户进行管理。

实现用户账号的管理，要完成的工作主要有以下几个方面：

- 用户账号的添加、删除与修改。
- 用户口令的管理。

5.2.1 与用户管理有关的系统文件

完成用户管理的工作有许多种方法，但是每一种方法实际上都是对有关的系统文件进行修改。与用户相关的信息都存放在一些系统文件中，这些文件包括 /etc/passwd, /etc/shadow。下面分别介绍这两个文件的内容。

1、passwd 文件

/etc/passwd 文件是用户管理工作涉及的最重要的一个文件。Linux 系统中的每个用户都在 /etc/passwd 文件中有一个对应的记录行，它记录了这个用户的一些基本属性。这个文件对所有用户都是可读的。它的内容类似下面的例子：

```
[root@zx 桌面]# cat /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
```

用户名:口令:UID:GID:注释性描述:主目录:登录 Shell

- “用户名”是代表用户账号的字符串。通常长度不超过8个字符，并且由大小写字母和/或数字组成。登录名中不能有冒号，冒号在这里是分隔符。为了兼容起见，登录名中最好不要包含点字符(.)，并且不使用连字符(-)和加号(+)打头。
- “口令”一些系统中，存放着加密后的用户口令字。。虽然这个字段存放的只是用户口令的加密串，不是明文，但是由于 /etc/passwd 文件对所有用户都可读，所以这仍是一个安全隐患。因此，**现在许多 Linux 系统使用了 shadow 技术，把真正的加密后的用户口令字存放到/etc/shadow 文件中，而在/etc/passwd 文件的口令字段中只存放一个特殊的字符，例如“x”或者“*”。**
- UID，系统内部用它来标识用户。
- GID 字段记录的是用户所属的用户组。它**对应着/etc/group 文件中的一条记录。**
- “注释性描述”字段记录着用户的一些个人情况，例如用户的真实姓名、电话、地址等，这个字段并没有什么实际的用途。在不同的 Linux 系统中，这个字段的格式并没有统一。在许多 Linux 系统中，这个字段存放的是一段任意的注释性描述文字，用做 finger 命令的输出。
- “主目录”，也就是用户的起始工作目录，它是用户在登录到系统之后所处的目录。在大多数系统中，各用户的主目录都被组织在同一个特定的目录下，而用户主目录的名称就是该用户的登录名。各用户对自己的主目录有读、写、执行（搜索）权限，其他用户对此目录的访问权限则根据具体情况设置。
- 用户登录后，要启动一个进程，负责将用户的操作传给内核，这个进程是用户登录到系统后运行的命令解释器或某个特定的程序，即 Shell。。系统管理员可以根据系统情况和用户习惯为用户指定某个 Shell。如果不指定 Shell，那么系统使用 sh 为默认的登录 Shell，即这个字段的值为/bin/bash。

系统中有一类用户称为伪用户 (pseudo users)，这些用户在/etc/passwd 文件中也占有一条记录，但是不能登录。它们的存在主要是方便系统管理，满足相应的系统进程对文件属主的要求。常见的伪用户如下所示。

伪 用户	含 义
bin	拥有可执行的用户命令文件
sys	拥有系统文件
adm	拥有帐户文件
uucp	UUCP 使用
lp	lp 或 lpd 子系统使用

nobody

NFS 使用

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

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

1、shadow 文件

/etc/shadow 中的记录行与/etc/passwd 中的一一对应，它由根据/etc/passwd 中的数据自动产生。它的文件格式与/etc/passwd 类似。

```
[root@zx 桌面]# tail -3 /etc/shadow
js01:!!:15357:0:99999:7:::15704:
ftp01:!!:15357:0:99999:7:::
zxstu01:!!:15357:0:99999:7:::
[root@zx 桌面]#
```

由若干个字段组成，字段之间用“：“隔开。这些字段是：

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

- “登录名”是与/etc/passwd 文件中的登录名相一致的用户账号
- “口令”字段存放的是 sha 加密后的用户口令，当为“*”或“!!”表示用户无法登录系统。若该字段内容为空，则用户无需口令就可登录系统。
- “最后一次修改时间”表示的是从 1970 年 1 月 1 日起到用户最后一次修改口令时的天数。
- “最小时时间间隔”指的是两次修改口令之间所需的最小天数，0 表示不限制。
- “最大时间间隔”指的是口令保持有效的最大天数，默认值 99999，表示不限制。
- “警告时间”字段表示的是从系统开始警告用户到用户密码正式失效之间的天数。
- “不活动时间”表示的是用户没有登录活动但账号仍能保持有效的最大天数。
- “失效时间”字段指定用户失效的天数(从 1970-01-01 起计算)，默认值为空，表示账号永远有效。

5.2.2 使用用户管理命令进行用户管理

1、useradd——添加用户账户

格式 useradd 选项 用户名

常用和选项：

- u 指定用户的 UID 号。
- d 指定用户主目录，如果此目录不存在，则同时使用-m 选项，可以创建主目录。

- g 指定用户所属的用户组。
- G 指定用户所属的附加组。
- s 指定用户的登录 Shell。
- e 指定用户的账户失效的时间，可以使用 YYYY-MM-DD 的日期格式。
- M 不建立用户主目录。

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

例：创建名为 zxstu01 的用户账号，查看 passwd、shadow 文件的变化和用户主目录。

```
[root@zx 桌面]# useradd zxstu01

[root@zx 桌面]# tail -1 /etc/passwd
zxstu01:x:508:508::/home/zxstu01:/bin/bash
[root@zx 桌面]# tail -1 /etc/shadow
zxstu01:!:15357:0:99999:7:::
[root@zx 桌面]# tail -3 /etc/shadow
js01:!:15357:0:99999:7::15704:
ftp01:!:15357:0:99999:7:::
zxstu01:!:15357:0:99999:7:::
[root@zx 桌面]# ll /home
总用量 20
drwx----- 4 js01    users   4096  1月 18 14:20 js01
drwx----- 4 lipi    lipi   4096  1月  4 07:53 lipi
drwx----- 4 zxstu01 zxstu01 4096  1月 18 15:32 zxstu01
drwx----- 4 zxstu02 zxstu02 4096  1月 18 14:11 zxstu02
drwx----- 4      501    501   4096  1月 18 14:02 zxuser01
[root@zx 桌面]#
```

例：创建名为 zxstu02 的用户账号，产将其 UID 指定为 505。

```
[root@zx 桌面]# useradd -u 505 zxstu02
[root@zx 桌面]#
```

例：创建一个管理员账号 admin，将其基本组指定为“wheel”、附加组指定为“root”。宿主目录为“/admin”

```
[root@zx 桌面]# useradd -d /admin -g wheel -G root admin
[root@zx 桌面]#
```

例：创建一个机试账号 js01，指定属于 users 组，该账号于 2012-12-30 失效。

```
[root@zx 桌面]# useradd -g users -e 2012-12-30 js01
[root@zx 桌面]#
```

例：创建一个用于 FTP 访问的账户 ftp01，**禁止此用户登录系统（指定 shell 为 /sbin/nologin）**，且不为其创建主目录。

```
[root@zx 桌面]# useradd -M -s /sbin/nologin ftp01
[root@zx 桌面]# █ 不建主目录 登录shell 用户名
```

在 Linux 系统中，大部分程序用户都是禁止登录到系统的。

2、passwd——用户口令管理

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

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

格式 passwd 选项 用户名

常用和选项：

- l 锁定口令，即禁用账号。
- u 口令解锁。
- d 清空口令，仅使用用户名即可登录系统。
- f 强迫用户下次登录时修改口令。
- S 查看状态

如果默认用户名，则修改当前用户的口令。

例：以 root 为 zxstu01 设置口令，并查看 shadow 文件的变化。

```
[root@zx 桌面]# tail -1 /etc/shadow
zxstu01:!!:15357:0:99999:7::: //在设置密码之前，此字段为!!
[root@zx 桌面]# passwd zxstu01
更改用户 zxstu01 的密码。
新的密码：
无效的密码：过于简单化/系统化
重新输入新的密码：
passwd: 所有的身份验证令牌已经成功更新。
[root@zx 桌面]# tail -1 /etc/shadow
zxstu01:$6$T2W0wcT0$3KM3Lygv8lrjTfot0L.i70XAFG1sVgoeKCgYhyrE8xUpZSH0oY7M
Lu9MQFK/pR1brp7bruPUng6i.W.:15357:0:99999:7::: //设置密码后，此字段为MD5加密密文
[root@zx 桌面]#
```

例：将用户 zxstu01 的口令锁定，查看 shadow 文件的变化和用户口令状态，解锁后再次查看。

```
[root@zx 桌面]# passwd -l zxstu01
锁定用户 zxstu01 的密码 。 锁定后状态
passwd: 操作成功
[root@zx 桌面]# tail -1 /etc/shadow
zxstu01:!!$6$T2W0wcT0$3KM3Lygv8lrjTfot0L.i70XAFG1s\
erLu9MQFK/pR1brp7bruPUn6i.W.:15357:0:99999:7:::
[root@zx 桌面]# passwd -S zxstu01
zxstu01 LK 2012-01-18 0 99999 7 -1 (密码已被锁定。)
[root@zx 桌面]# 
```

```
[root@zx 桌面]# passwd -u zxstu01
解锁用户 zxstu01 的密码 。 解锁后状态
passwd: 操作成功
[root@zx 桌面]# passwd -S zxstu01
zxstu01 PS 2012-01-18 0 99999 7 -1 (密码已设置，使用 SHA512 加密。)
[root@zx 桌面]# 
```

例：以 zxstu01 用户登录终端后，自己修改自己的口令，需要原口令验证。

```
Red Hat Enterprise Linux Server release 6.0 (Santiago)
Kernel 2.6.32-71.el6.i686 on an i686

zx login: zxstu01
Password: 
Last login: Wed Jan 18 16:04:33 on tty2
[zxstu01@zx ~]$ passwd
Changing password for user zxstu01.
Changing password for zxstu01.
(current) UNIX password:
New password: 
```

3、usermod——修改用户账号属性

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

格式 usermod 选项 用户名

常用和选项：

- l 改变用户账户的登录名称。
- u 修改用户的 UID 号。
- d 修改用户主目录。
- g 修正用户所属的用户组。
- G 修改指定用户所属的附加组。
- s 指定用户的登录 Shell。
- e 修改用户的账户失效的时间，可以使用 YYYY-MM-DD 的日期格式。
- M 不建立用户主目录。
- L 锁定用户账户。

-U 解锁用户账户。

例：将 admin 用户的宿主目移动到/home 目录下，并使用 usermod 命令修改主目录。

```
[root@zx 桌面]# mv /admin /home/
[root@zx 桌面]# usermod -d /home/admin admin
[root@zx 桌面]#
```

4、userdel——删除用户账号

如果一个用户的账号不再使用，可以从系统中删除。删除用户账号就是要将/etc/passwd 等系统文件中的该用户记录删除，必要时还删除用户的主目录。删除一个已有的用户账号使用 userdel 命令。

格式 userdel 选项 用户名

常用的选项是-r，它的作用是把用户的主目录一起删除。

```
[root@zx 桌面]# userdel -r zxstu02
[root@zx 桌面]# ls -ld /home/zxstu02
ls: 无法访问 /home/zxstu02: 没有那个文件或目录
[root@zx 桌面]#
```

此命令删除用户 zxstu02 在系统文件中（主要是 /etc/passwd, /etc/shadow, /etc/group 等）的记录，同时删除用户的主目录。

5.3 组账户的管理

每个用户都有一个用户组，系统可以对一个用户组中的所有用户进行集中管理。不同 Linux 系统对用户组的规定有所不同，如 Linux 下的用户属于与它同名的用户组，这个用户组在创建用户时同时创建。

用户组的管理涉及用户组的添加、删除和修改。组的增加、删除和修改实际上就是对 /etc/group 文件的更新。

5.3.1 与用户管理有关的系统文件

用户组的所有信息都存放在/etc/group 文件中。此文件的格式也类似于/etc/passwd 文件。

```
[root@zx 桌面]# cat /etc/group
root:x:0:root,admin
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon
```

由冒号隔开若干个字段，这些字段有：

组名:口令:组标识号:组内用户列表

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

5.3.2 使用组管理命令来管理组

1、groupadd——添加组账号

格式 groupadd 选项 组名

常用的选项是-g，它的作用是指定 GID。

例：创建组 zxkj，并查看/etc/group 文件。

```
[root@zx 桌面]# groupadd zxkj
[root@zx 桌面]# tail -1 /etc/group
zxkj:x:509:
[root@zx 桌面]#
```

2、gpasswd——添加、删除组成员

Gpasswd 命令本来是用来设置组的口令，但极少使用，**实际上更多是用该命令来为组添加、删除用户成员。**

格式 gpasswd 选项 用户名 组名

常用的选项

- a 添加一个成员
- d 删除成员
- M 同时添加多个成员，多个用户用“，”隔开。

例：将用户 zxstu01 加入组 zxkj。

```
[root@zx 桌面]# tail -1 /etc/group
zxkj:x:509:
[root@zx 桌面]# grep "zxkj" /etc/group
zxkj:x:509:
[root@zx 桌面]# gpasswd -a zxstu01 zxkj
Adding user zxstu01 to group zxkj
[root@zx 桌面]# grep "zxkj" /etc/group
zxkj:x:509:zxstu01
[root@zx 桌面]#
```

3、groupdel——删除组账号

例：将组 zxkj 删除。

```
[root@zx 桌面]# groupdel zxkj
[root@zx 桌面]#
```

5.4 用户和组账号查询

在用户管理工作中，虽然可以通过查看配置文件可以查看用户、组相关信息，但不直观。在 Linux 系统中，还有几个常用的查询命令，以下逐一介绍。

1、id——查看用户身份标识

`Id` 命令可以查看当前用户的 UID 和所对应的基本组、附加组信息。

例：查看 root 用户的身份标识信息。

```
[root@zx 桌面]# id
uid=0(root) gid=0(root) 组=0(root),1(bin),
el) 环境=unconfined_u:unconfined_r:unconfi
[root@zx 桌面]#
```

2、groups——查看用户所属的组

例：分别查看当前 root 用户和 zxstu01 所属的组。

```
[root@zx 桌面]# groups
root bin daemon sys adm disk wheel
[root@zx 桌面]# groups zxstu01
zxstu01 : zxstu01
[root@zx 桌面]#
```

3、users、w、who——查看当前登录到主机用户信息

例：比较这三个命令的输出不同。

```
[root@zx 桌面]# users //仅列出用户名
root root root zxstu01
[root@zx 桌面]# who //列出用户名、终端、登录时间
root    tty1      2012-01-04 01:00 (:0)
root    pts/0      2012-01-17 23:16 (:0.0)
root    pts/1      2012-01-18 14:02 (:0.0)
zxstu01  tty2      2012-01-18 16:06
[root@zx 桌面]# w //列出用户名、终端、执行命令的统计信息
17:26:29 up 1 day, 3:17, 4 users, load average: 0.08, 0.02, 0.01
USER   TTY     FROM           LOGIN@  IDLE   JCPU   PCPU WHAT
root    tty1    :0            04Jan12 14days 6:35   6:35  /usr/bin/Xorg
root    pts/0   :0.0          Tue23   17:54m  0.05s  0.05s /bin/bash -l
root    pts/1   :0.0          14:02   0.00s  0.23s  0.06s w
zxstu01  tty2    -            16:06   1:03m  0.03s  0.03s -bash
[root@zx 桌面]#
```

6.5 用户切换与提权

大多数 Linux 服务器并不建议用户直接以 root 用户进行登录。一方面可以大大减少因误操作而导致的破坏，另一方面也降低了特权密码在不安全的网络中被泄露的风险。鉴于这些原因，需要为普通用户提供一种身份切换或权限提升机制，以便在必要的时候执行管理任务。

Linux 系统为我们提供了 su、sudo 两种机制，其中 su 主要用来切换用户，而 sudo 用来提升执行权限。下面分别进行讲解。

1、su 命令——切换用户

使用 su 命令，可以切换为指定的另一个用户，从而具有该用户的所有权限，当然，切换时需要目标用户的密码进行验证（从 root 切换为其他用户时例外），例如，当前登录的用户为 jerry，若要切换为 root 用户，可以执行以下操作。

```
[jerry@zx~]$ su - root
口令:                                              //输入 root 用户口令
[root@zx ~]#                                         //验证成功后获得 root 权限
```

上述命令操作中，选项“-”等同于“--login”或“-l”，表示切换后进入目标用户的登录 Shell 环境，若缺少此选项仅切换身份，不切换用户环境。对于切换为 root 用户的情况，“root”可以省略。

2、sudo 命令——提升权限

通过 su 命令可以非常方便地切换为另一个用户，但前提条件是必须知道目标用户的登录密码。例如，若要从 jerry 用户切换为 root，必须知道 root 用户的密码。对生产环境中的 Linux 服务器来说，多一个人知道特权密码，其安全风险也就增加一分。

那么，有没有一种折衷的办法，即可以让普通用户拥有一部分管理权限，又不需要将 root 用户的密码告诉他呢？

1) 在配置文件/etc/sudoers 中添加授权

Sudo 机制的配置文件位于 /etc/sudoers，文件的默认权限为 440，需使用专门的 visudo 工具进行编辑，虽然也可以用 vi 进行编辑，但保存时必须执行 “:w!” 来强执操作，否则系统将提示为只读文件而拒绝保存。

配置文件 /etc/sudoers 中，授权记录的基本配置格式如下所示。

```
user  MACHINE = COMMANDS
```

主要包括用户、主机、命令三个部分，即授权哪些人在哪些主机上执行哪些命令。各部分的具体含义如下所述。

- 用户(USER): 授权的用户名，或采用“%组名”的形式（授权一个组的所有用户）。
- 主机(MACHNE): 使用此配置文件的主机名称。此选项主要是方便在多个主机间共用同一份 sudoers 文件，一般设为 localhost 或者实际的主机名即可。
- 命令(COMMANDS): 允许授权的用户通过 sudo 方式执行的特权命令，需填写命令程序的完整路径，多个命令之间以逗号“,”进行分隔。

典型的 sudo 配置记录中，每一行对应一个用户或组的 sudo 授权配置。例如，若要授权用户 jerry 能够执行 ifconfig 命令来修改 IP 地址，wheel 组的用户不需要验证密码即可执行任何命令，可以执行以下操作。

```
[root@zx~]# visudo
.....
jerry    instructor = /sbin/ifconfig
```

```
%wheel  All = NOPASSWD: ALL
```

当使用相同授权的用户较多，或者授权的命令较多时，可以采用集中定义的别名。用户、主机、命令部分都可以定义为别名（必须为大写），分别通过关键 User_Alias 、 Host_Alias 、 Cmnd_Alias 来进行设置。例如，以下操作通过别名方式来添加授权记录，允许用户 jerry 、 tom 在主机 smtp 、 pop 中执行 rpm 、 yum 命令。

```
[root@zx~]# visudo
.....
User_Alias OPERATORES = jerry, tom
Host_Alias MAILSVRS = smtp, pop
Cmnd_Alias PKGTOOLS = /bin/rpm, /usr/bin/yum
OPERATORES MAILSVRS = PKGT001S
```

Sudo 配置记录的命令部分允许使用通配符 “*”，取反符号 “!”，当需要授权某个目录下的所命令，取消其中个别命令时特别有用。例如，若要授权用户 lisi 用户可以执行/sbin/ 目录下除 ifconfig 、 route 以外的其他所有命令程序，可以执行以下操作。

```
[root@zx~]# visudo
.....
lisi    localhost = /sbin/*,!/sbin/ifconfig,!/sbin/route
```

默认情况下，通过 sudo 方式执行的操作并不记录。若要启用 sudo 日志记录以备管理员查看，应在/etc/sudoers 文件中增加“Defaults logfile”设置。

```
[root@zx~]# visudo
.....
Defaults logfile = "/var/log/sudo"
```

2) 通过 sudo 执行特权命令

对于已获得授权的用户，通过 sudo 方式执行特权命令时，只需要将正常的命令行作为 Sudo 命令的参数即可，由于特权命令程序通常位于/sbin/ 、 /usr/sbin/ 等目录下，普通用户执行时应使用绝对路径，以下操作验证了使用 sudo 方式执行命令的过程。

```
[jerryt@zx~]$ /sbin/ifconfig eth0:0 192.168.1.11/24          //未使用 sudo
SICOCASFADDR :权限不够
.....
[jerryt@zx~]$ sudo /sbin/ifconfig eth0:0 192.168.1.11/24      //使用 sudo
[sudo] password for jerry:                                //验证 jerry 口令
```

在当前会话过程中，第一次通过 sudo 执行命令时，必须以用户自己的密码（不是 root 或其他用户的密码）进行验证，此后再通过 sudo 执行命令时，只要与前一次 sudo 操作的间隔时间不超过 5 分钟，则不再重复验证。

若要查看用户自己获得哪些 sudo 授权，可以执行“sudo-l”命令，未授权的用户将会得到“may not run sudo”的提示，已授权的用户则可以看到自己的 sudo 配置。

如果已经启用了 sudo 日志，则可以从/var/log/sudo 文件中看到用户的 sudo 操作记录。

第六章 Linux 文件系统的权限

6.1 Linux 文件概述

Linux 一切皆文件，这里的“一切”确确实实意味着一切。硬盘，硬盘分区，并行口，到网站的连接，以太网卡：所有这些都是文件。甚至目录也是文件。

1、Linux 文件属性

使用 ls -al 显示当前目录中所有文件。

```
[root@zx ~]# ll
文件所  文件属组  文件大小  更改时间  文件名
类型 权限 链接数 有者
-rw-----. 1 root root 2036 1月 4 06:55 anaconda-ks.cfg
-rw-r--r--. 1 root root 0 1月 4 01:46 a.txt
drwx-----. 2 root root 4096 1月 17 23:21 Desktop
-rw-r--r--. 1 root root 0 1月 4 01:43 file2
-rw-r--r--. 1 root root 59017 1月 4 06:55 install.log
-rw-r--r--. 1 root root 12853 1月 4 06:51 install.log.syslog
-rw-r--r--. 1 root root 137015 1月 17 23:10 zebra-0.95a.tar.gz
-rw-r--r--. 1 root root 48 1月 17 23:16 zebra-0.95a.tar.gz.st
drwxr-xr-x. 2 root root 4096 1月 4 23:54 公共的
drwxr-xr-x. 2 root root 4096 1月 4 23:54 模板
drwxr-xr-x. 2 root root 4096 1月 4 23:54 视频
drwxr-xr-x. 2 root root 4096 1月 4 23:54 图片
drwxr-xr-x. 2 root root 4096 1月 4 23:54 文档
drwxr-xr-x. 2 root root 4096 1月 4 23:54 下载
drwxr-xr-x. 2 root root 4096 1月 4 23:54 音乐
drwxr-xr-x. 3 root root 4096 1月 17 23:22 桌面
[root@zx ~]#
```

2、Linux 文件类型

- - : 这是一类常见的文件，也是常使用的一类文件，其特点是不包含有文件系统的结构信息。通常所接触到的文件，包括图形文件、数据文件、文档文件、声音文件等都属于这种文件。这种类型的文件按其内部结构又可细分为文本文件和二进制文件。
- l : 链接文件，接文件是一种特殊的文件，实际上是指向一个真实存在的文件的链接。这有点类似于 Windows 下的快捷方式。根据链接文件的不同，它又可以细分为硬链接文件和符号链接文件。
- d : 目录文件，目录文件是用于存放文件名及其相关信息的文件，是内核组织文件系统的基本节点。目录文件可以包含下一级目录文件或普通。对于习惯于使用 Windows 的用户来说，这可能有些难于理解，目录怎么会是文件呢？的确，在 Linux 中，目录文件是一种文件。但 Linux 的目录文件和其它操作系统中的“目录”的概念不同，它是 Linux 文件中的一种。当然，在实际使用中可以不仔细区分这

两种说法。实际上，在很多 Linux 的书籍和资料中就是将目录文件简称为目录的。不过，我们必需清楚此“目录”非彼“目录”。

- b : 块设备文件，设备文件是 UNIX 系统中的一种文件，用处是将复杂的多元的硬件抽象成文件，这样我们通过简单的打开方法就可以打开硬件设备了。
- c : 字符型设备文件，与块设备文件的区别在于，一般块设备的数据是以一定的大小一块来操作，而字符型设备文件是可以一个字节一个字节操作，不需要分块。通常 Linux 系统将设备文件放在 /dev 目录下，设备文件使用设备的主设备号和次设备号来指定某外部设备。根据访问数据方式的不同，设备文件又可以细分为块设备和字符设备文件。
- s : 网络类型文件，一般可以在 /tmp 目录下发现他的身影，是系统进行网络通信时使用的 socket 文件。
- p : 管道文件，管道文件是一种很特殊的文件，主要用于不同进程间的信息传递。当两个进程间需要进行数据或信息传递时，可以通过管道文件。一个进程将需传递的数据或信息写入管道的一端，另一进程则从管道的另一端取得所需的数据或信息。通常管道是建立在调整缓存中。

Linux 系统根据文件的访问权限、归属来对用户访问资源（如数据）的过程进行控制。下面重点介绍文件的权限和归属。

6.2 管理文件的权限和归属

6.2.1 查看文件的权限和归属

Linux 系统中，对文件来说，可以为三类用户设定权限：

文件拥有者 (user)

文件拥有组的成员 (group)

其他用户 (other)

它的权限可以分为三种：读的权限 (r)、写的权限 (w) 和执行的权限 (x)。不同的用户具有不同的读、写和执行的权限。

```
[root@zx ~]# ls -l /root/install.log
-rw-r--r--. 1 root root 59017 1月 4 06:55 /root/install.log
[root@zx ~]#
```

在看到的信息中，第一列的第一个字符表示文件类型，第 2~10 这 9 个字符表示权限，具体含义如下：

权限项	读	写	执行	读	写	执行	读	写	执行
字符表示	(r)	(w)	(x)	(r)	(w)	(x)	(r)	(w)	(x)
权限分配	文件所有者		文件所属组用户			其他用户			

所以 install.log 文件的权限是：用户 root 拥有 rw（读写）权限，root 组的成员拥有 r（读）的权限，其他人（others）拥有 r（读）的权限

6.2.2 设置文件的权限

建立新文件时，文件的缺省权限，跟用户有关，如果是 root 用户，他创建的文件权限为：r w - r - - r - -，而普通用户创建文件的权限为：r w - r w - r - -当它不符合用户需求或者当文件的运行环境将要发生变化时，需要修改文件的权限信息，Linux系统提供了 chmod, chown, chgrp 命令来重新设置文件的权限信息。

使用 chmod 命令设置文件或者目录的操作权限，可以采用两种形式的权限表示方法：字符形式和数字形式。

1、字符方式设置权限

字符方式的基本语法是：chmod [ugoa] +或者-或者= [rwx] [文件名称]

Chmod	User	运算符	Mode	文件名
	u	+	rwx	/etc/a.conf
	g		rwx	
	o		rwx	
	a		rwx	

三部分的含义及用法如下所述。

- ugoa 表示该权限设置所针对的用户类型，u 代表文件属主，g 代表文件属组内的用户，o 代表其他任何用户， a 代表所有用户。
- +- 表示设置权限的操作动作，+ 代表增加相应权限，- 代表减少相应权限，= 代表仅置对应的权限。
- rwx 是权限的符组合。

例：在当前目录中新建一个文件file1，查看权限。

```
[root@zx ~]# touch file1
[root@zx ~]# ll file1
-rw-r--r--. 1 root root 0 1月 18 22:54 file1
[root@zx ~]#
```

例：对上例中文件 file1，为文件属主增加执行权限，为宿组增加写的权限，其他用户增加执行权限。

```
[root@zx ~]# chmod u+x,g+w,o+x file1
[root@zx ~]# ll file1
-rwxrw-r-x. 1 root root 0 1月 18 22:54 file1
[root@zx ~]#
```

2、数字方式设置权限

赋予权限的另一种方法是 chmod XYZ [文件...]

XYZ 分别是 0—7 数字，其中 X 表示属主用户权限、B 表示属组权、C 表示所有其它用户权限。各个位置上的值要么是 0，或者是一个由赋予权限的相关值相加得到的单个阿拉伯数字之和。这些数字的意义如下所示：

权限项	读	写	执行	读	写	执行	读	写	执行
字符表示	(r)	(w)	(x)	(r)	(w)	(x)	(r)	(w)	(x)
数字表示	4	2	1	4	2	1	4	2	1
权限分配	文件所有者			文件所属组用户			其他用户		

一般而言，作为系统管理员，更喜欢使用数字方式，因为这种方式的速度明显快得多。

例：

```
[root@zx ~]# chmod 755 file1
[root@zx ~]# ll file1
-rwxr-xr-x. 1 root root 0 1月 18 22:54 file1
[root@zx ~]#
```

6.2.3 设置文件的归属

使用 chown 命令更改文件或者目录的用户所有权。

基本格式

chown [-R] user filename

[-R] : 递归修改子目录中的文件

user : 用户名（属主或属组）

filename : 操作的文件名称或目录名称

举例：将 file1 的属主修改为用户 lipi 所有。

```
[root@zx ~]# ll file1
-rwxr-xr-x. 1 root root 0 1月 18 22:54 file1
[root@zx ~]# chown lipi file1
[root@zx ~]# ll file1
-rwxr-xr-x. 1 lipi root 0 1月 18 22:54 file1
[root@zx ~]#
```

例：将文件 file1 的属主改为用户 zxstu01 和属组 zxstu01 所有

```
[root@zx ~]# ll file1
-rwxr-xr-x. 1 lipi root 0 1月 18 22:54 file1
[root@zx ~]# chown zxstu01:zxstu01 file1
[root@zx ~]#          属主      属组
[root@zx ~]# ll file1
-rwxr-xr-x. 1 zxstu01 zxstu01 0 1月 18 22:54 file1
[root@zx ~]# █
```

6.2.4 使用附加的权限

对文件或目录进行访问控制时，“读”、“写”、“执行”是最基本的三种权限类型。除此之外 Linux 还存在几种特殊的附加权限，用于为文件或目录提供额外的控制方式。可用的附加权限包括：**set 位权限 (SUID、SGID)**、**粘滞位 (Sticky Bit)**，下面分加进行讲解。

1、SET 位权限

SET 位权限多用于给可执行的程序或脚本文件进行设置，其中 SUID 表法属主用户增加 SET 位权限，DGID 表示属组内的用户增加 SET 位权限。**执行文件被设置了 SUID、SGID 后，任何用户在执行该文件时，将获得该文件属主、属组账号对应的身份。**

(1)、为什么要使用 SET 位权限？

举例说明：

ping 命令应用广泛，可以测试网络是否连接正常。ping 在运行中是采用了 ICMP 协议，需要发送 ICMP 报文。但是只有 root 用户才能建立 ICMP 报文，如何解决这个问题呢？就是 通过 SUID 位来解决。

再有 Linux 系统中 password 命令文件被设置了 SUID 权限，正因为如此，尽管普通用户无法直接修改/etc/shadow 文件，但仍然可以通过 passwd 命令修改自己的登录的密码，从而以 root 用户和身份间接更新 shadow 文件。

(2)、设置 SUID/SGID

为执行文件添加 SET 位权限同样可以通过 chmod 命令实现，**使用 u+s 、 g+s 分别设置 SUID、GUID 权限**。若使用数字形式，SUID 对应的八进制数字为 4，SGID 对应的八进制数字为 2，在权限模式可以采用 nnn 的形式，如 4755 表示设置了 SUID 权限、6755 表示同时设置 SUID、SGID 权限。

设置 SUID、SGID 权限后，使用 ls 命令查看文件的属性时，对应的位置的 x 将变成 s。

例：查看 ping 命令文件是否设置了 SUID 权限

```
[root@zx ~]# ls -l /bin/ping
-rwsr-xr-x. 1 root root 42136 7月 27 2010 /bin/ping
[root@zx ~]# █
```

例：为 file1 文件设置 SUID 权限位。

```
[root@zx ~]# ll file1
-rwxr-xr-x. 1 zxstu01 zxstu01 0 1月 18 22:54 file1
[root@zx ~]# chmod 4751 file1
[root@zx ~]# ll file1
-rwsr--x--x. 1 zxstu01 zxstu01 0 1月 18 22:54 file1
[root@zx ~]#
```

2、粘滞位权限

粘滞位权限主要用于为目录设置的附加权限，当目录被设置了粘滞位权限以后，即使用户对该目录有写入的权限，也不能删除该目录中其他用户的文件。

在 Linux 系统中比较典型的例子就是 /tmp、/var/tmp 目录，这两个目录做为 Linux 系统的临时文件夹，权限为 rwxrwxrwx，即允许任意用户、任一程序在该目录中进行创建、删除、修改等操作。然而试想一下，若任一普通用户都参删除系统服务运行中使用的临时文件，将造成什么后？这个问题就是通过设粘滞位权限来解决的，设置了粘滞位权限后，允许用户可在任意写入、删除自己的文件，但禁止删除其他用户的文件。

设置了粘滞位权限的目录，查看其属性时，其他用户的权限由 x 变成 t。

例：查看 /tmp 目录的权限。

```
[root@zx ~]# ls -ld /tmp
drwxrwxrwt. 13 root root 4096 1月 18 21:29 /tmp
[root@zx ~]#
```

使用 chmod 命令设置目录权限时，+t、-t 权限模式可以分别添加、移除粘滞位权限。将数字模式的 nnn 中第一位改为 1、0，也可以实现添加、移除粘滞位的权限。

例：在当前目录中创建一个名为 mydir 的目录，允许所有用户读、写、执行操用，并为该目录设置粘滞位权限，禁止删除其他用户的文件。

```
[root@zx ~]# mkdir mydir
[root@zx ~]# chmod 1777 mydir
[root@zx ~]# ls -ld mydir
drwxrwxrwt. 2 root root 4096 1月 19 01:16 mydir
[root@zx ~]#
```

6.3 umask

umask 命令确定了你创建文件的缺省权限。你的系统管理员必须要为你设置一个合理的 umask 值，以确保你创建的文件具有所希望的缺省权限，防止其他非同组用户对你的文件具有写权限。

在已经登录之后，可以按照个人的偏好使用 umask 命令来改变文件创建的缺省权限。相应的改变直到退出该 shell 或使用另外的 umask 命令之前一直有效。

一般来说，umask 命令是在/etc/profil 文件中设置的，每个用户在登录

时都会引用这个文件，所以如果希望改变所有用户的 umask，可以在该文件中加入相应的条目。如果希望永久性地设置自己的 umask 值，那么就把它放在自己\$HOME 目录下的. profile 或. bash_profile 文件中。

1、如何计算 umask 值

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

该命令的一般形式为：

```
umask nnn
```

umask 是从权限中“拿走”相应的位即可。文件的全部权限是 666，目录的全部权限是 777，然后减去响应的 umask 值就默认的文件和目录权限。

例如，对于 umask 值 002，相应的文件和目录缺省创建权限是什么呢？

- 1)、文件的最大权限 rwx rwx rwx (777)
- 2)、umask 值为 002 ----- -w-
- 3)、目录权限 rwx rwx r-x (775) 这就是目录创建缺省权限
- 4)、文件权限 rw- rw- r-- (664) 这就是文件创建缺省权限

2、如果想知道当前的 umask 值，可以使用 umask 命令：

```
#su sam /*切换到 sam 用户环境下
#umask /*查看 sam 的 umask
0022
```

前面多了个 0，是 uid/gid 的，但在 umask 中此位只能为 0 或是省略。

3、当新增文件或目录时，预设的使用权限，由 umask 这个内设值所规定的。

一般情况下，umask 会被设定在 shell 的启始档案中。

对 bash 的使用者来说，个人的启始档案是 \$HOME/.bashrc，使用者可以将 umask 设定在其中。

第七章 监控和管理 Linux 进程

7.1 进程管理

程序是为了完成某种任务而设计的软件，比如 OpenOffice 是程序。什么是进程呢？进程就是运行中的程序。

一个运行着的程序，可能有多个进程。比如我们公司所用的 WWW 服务器是 apache 服务器，当管理员启动服务后，可能会有好多人来访问，也就是说许多用户来同时请求 httpd 服务，apache 服务器将会创建有多个 httpd 进程来对其进行服务。

7.1.1 查看进程

了解系统中进程状态是对进程管理的前提，使用不同的命令工具可以从不同的角度查看进程状态。下面学习几个常用的进程查看命令。

1、ps——查看进程统计信息

ps 命令就是最基本同时也是非常强大的进程查看命令。使用该命令可以确定有哪些进程正在运行和运行的状态、进程是否结束、进程有没有僵死、哪些进程占用了过多的资源等等。总之大部分信息都是可以通过执行该命令得到的命令是 Linux 系统中最常用的进程查看工具，ps 为我们提供了进程的一次性的查看，它所提供的查看结果并不动态连续的；如果想对进程时间监控，应该用 top 工具；

ps 的常用参数：

- a 显示所有进程，与 x 结合将显示系统中所有的进程信息。
- x 显示当前用户在所有终端的进程信息。
- l 以长格式显示进程的信息。
- u 显示进程的拥有者信息。

例：查看当前用户会话中打开的进程。

```
[root@zx ~]# ps
  PID TTY          TIME CMD
14459 pts/0    00:00:00 bash
14474 pts/0    00:00:00 ps
[root@zx ~]#
```

例：显示系统中所有的进程信息。

```
[root@zx ~]# ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root        1  0.0  0.1  2824  1340 ?      Ss 20:37  0:01 /sbin/init
root        2  0.0  0.0      0     0 ?      S 20:37  0:00 [kthreadd]
root        3  0.0  0.0      0     0 ?      S 20:37  0:00 [migration/0]
root        4  0.0  0.0      0     0 ?      S 20:37  0:00 [ksoftirqd/0]
root        5  0.0  0.0      0     0 ?      S 20:37  0:00 [watchdog/0]
root        6  0.0  0.0      0     0 ?      S 20:37  0:00 [events/0]
root        7  0.0  0.0      0     0 ?      S 20:37  0:00 [cpuset]
root        8  0.0  0.0      0     0 ?      S 20:37  0:00 [khelper]
root        9  0.0  0.0      0     0 ?      S 20:37  0:00 [netns]
root       10  0.0  0.0      0     0 ?      S 20:37  0:00 [async/mgr]
root       11  0.0  0.0      0     0 ?      S 20:37  0:00 [pm]
```

其中各字段含义如下。

- USER：启动该进程的账户的名称。
- PID：进程 ID。

- %CPU: CPU 占用百分比。
 - %MEM: 内存占用百分比。
 - VSZ: 占用虚拟内存（swap 空间）的大小。
 - RSS: 占用常驻内存（物理内存）的大小。
 - TTY: 表明该进程在哪个终端上运行，“?”表示未知或不需终端。
 - STAT: 显示了进程的当前的状态，如 S (休眠)、R (运行)、Z (僵死)、< (高优先级)、N (低优先级)、s (父进程)、+ (前台进程)。
 - START: 启动该进程的时间。
 - TIME: 该进程占用 CPU 时间
 - COMMAND: 启用该进程的命令的名称。

例：查看所有进程信息，过滤出包含“bash”的进程信息。

```
[root@zx ~]# ps aux | grep bash
root      14459  0.0  0.2   6700  1612 pts/0    Ss   22:11   0:00 /bin/bash -l
root      15170  0.0  0.1   5936   740 pts/0    S+   22:30   0:00 grep bash
[root@zx ~]#
```

2、top——查看进程动态信息

`ps` 为我们提供了进程的一次性的查看，它所提供的查看结果并不动态连续的；如果想对进程时间监控，应该用 `top` 工具；`top` 默认情况每 3 秒刷新一次，类似于 Windows 系统中的“任务管理器”。

例：使用 top 命令查看进程的情况。

```
[root@zx ~]# top
```

top - 22:41:33 up 2:03, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 145 total, 1 running, 144 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.7%us, 0.3%sy, 0.0%ni, 99.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%
Mem: 675020k total, 616768k used, 58252k free, 62904k buffers
Swap: 511992k total, 0k used, 511992k free, 346544k cached

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1988	root	20	0	45268	14m	6224	S	1.0	2.2	0:10.69	Xorg
1	root	20	0	2824	1340	1124	S	0.0	0.2	0:01.88	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
4	root	20	0	0	0	0	S	0.0	0.0	0:00.01	ksoftirqd/0
5	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
6	root	20	0	0	0	0	S	0.0	0.0	0:00.06	events/0
7	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuset
8	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khelper
9	root	20	0	0	0	0	S	0.0	0.0	0:00.00	netns

在 top 命令显示的界面中，可以按 P 键按 CPU 占用率排序，按 M 键根据内存占用情况排序，按 N 键根据启动时间进行排序，按 q 键可以正常退出 top 程序。

3、 pgrep——查看进程 ID

`pgrep` 查看进程的 ID。

```
[root@zx ~]# pgrep init
1
[root@zx ~]#
```

4、pstree——查看进程树

pstree 命令可以显示系统中各进程的树型结构，能够更直观地判断各进程间的相互关系（父子进程）。

7.1.2 控制进程

上一小节学习如何查看系统中的进程信息，下面将继续学习进程的启动、调度和终止操作。

1、启动进程

键入需要运行的程序的程序名，执行一个程序，其实也就是启动了一个进程。在 Linux 系统中每个进程都具有一个进程号，用于系统识别和调度进程。启动一个进程有两个主要途径：**手工启动和调度启动**，后者是事先进行设置，根据用户要求自行启动。

1) 手工启动进程

由用户输入命令，直接启动一个进程便是手工启动进程。手工启动进程可分为前台启动和后台启动。

➤ 前台启动

这或许是手工启动一个进程的最常用的方式。一般地，用户键入一个命令“ls -l”，这就已经启动了一个进程，而且是一个前台的进程。用户在键入“ls -l”命令以后赶紧使用“ps -x”查看，却没有看到 ls 进程，也觉得很奇怪。其实这是因为 ls 这个进程结束太快，使用 ps 查看时该进程已经执行结束了。如果启动一个比较耗时的进程：find / -name fox.jpg 然后再把该进程挂起，使用 ps 查看，就会看到一个 find 进程在里面。

➤ 后台启动

直接从后台手工启动一个进程用得比较少一些，除非是该进程甚为耗时，且用户也不急着需要结果的时候。假设用户要启动一个需要长时间运行的进程，从后台启动这个进程是明智的选择。**从后台启动进程其实就是在命令结尾加上一个&号**。进程启动放到后台运行，而不占用前台的操作界面，方便用户进行其他操作。

2) 调度启动

有时候需要对系统进行一些比较费时而且占用资源的维护工作，这些工作适合在深夜进行，这时候用户就可以事先进行调度安排，指定任务运行的时间或者场合，到时候系统会自动完成这一切工作。

2、改变进程的运行方式

1) 把当前终端中运行的进程调入后台

当 Linux 系统中的命令正在前台执行时（运行尚未结束），按 **Ctrl+Z** 组合键可以将当前进程挂起（调入后台并停止运行），这种操作在需要暂停当前进程并进行其他操作时特别有用。

2) 查看后台的进程

需要查看当前终端中在后台运行的进程任务时，可用使用 `jobs` 命令，结合 “-l” 选项可以同进显示出该进程对应的 PID 号，在 `jobs` 命令是输出结果中，每一行记录对应一个后台进程的状态信息，行首的数字表示该进程在后台的任务编号。若当前终端没有后台进程，将不会显示任何信息。

3) 将后台的进程恢复到前台执行

使用 `fg` 命令可以将后台的进程任务重新调入终端的前台执行，只需指定后台任务对应的顺序编号（通过 `jobs` 命令查询获取）作为参数即可。若当前终端的后台中只有一个进程，可以不指定任务编号参数，`fg` 命令会自动将后台中唯一的进程调入前台执行。

3、终止进程

当用户在前台执行某个进程进，**可以按 `Ctrl+C` 组合键强行中断**（例如，命令长时间没有响应的情况下）。中断前台进程的运行后，系统将返回到命令提示符状态等待用户输入新的命令。当按下 `Ctrl+C` 组合键无法终止程序或者需要结束在其他终端的，后台运行的进程时，可以使用专用的进程终止工具 `kill`、`killall`、和 `pkill`。

1) 使用 `kill` 命令终止进程

使用 `kill` 命令终止进程，需要使用进程的 PID 号作为参数。无特定选项时，`kill` 命令将给该进程发送终止信号并正常退出运行，若该进程已经无法响应终止信号，则可以结合 “-9” 选项强行杀死进程。强制终止进程时可能会导致程序运行的部分的数据的丢失，因此不到不得以时不要轻易使用 “-9” 选项。

2) 使用 `killall` 命令终止进程

使用 `killall` 命令可以通过进程名来杀死进程，当需要结束系统中多个相同名称的进程时，使用 `killall` 命令更加方便，效率更高。`Killall` 命令也可以结合 “-9” 选项以强制结束进程。

3) 使用 `pkill` 命令终止进程

使用 `pkill` 命令可以根据进程的名称、用户、终端等多种属性来终止特定的进程，选项“-U”指用户、“-t” 指定终端。

第八章 控制服务和守护进程

8.1 systemd

`systemd` 对 Linux 来说，是一个 `init` 程序，可以作为 `SysVinit` 和 `Upstart` 的替代。作为一个系统和服务管理器，`systemd` 试图提供一种更好的初始化结构框架，来解决各种服务之间的依赖关系，使得在系统启动的时候更多的工作能够并行化。`systemd` 的特性有：

- ❖ 支持并行化任务
- ❖ 同时采用 socket 式与 D-Bus 总线式激活服务；
- ❖ 按需启动守护进程（daemon）；
- ❖ 利用 Linux 的 cgroups 监视进程；
- ❖ 支持快照和系统恢复；
- ❖ 维护挂载点和自动挂载点；

- ❖ 各服务间基于依赖关系进行精密控制

监视和控制 systemd 的主要命令是 `systemctl`。该命令可用于查看系统状态和管理系统及服务。

使用单元：一个单元配置文件可以描述如下内容之一：系统服务（`.service`）、挂载点（`.mount`）、sockets（`.sockets`、系统设备、交换分区/文件、启动目标（target）、文件系统路径、由 `systemd` 管理的计时器。使用 `systemctl` 控制单元时，通常需要使用单元文件的全名，包括扩展名（例如 `sshd.service`）。但是有些单元可以在 `systemctl` 中使用简写方式。

8.2 systemctl 命令

1、列出单元

`systemctl` 命令可以带上 `list-units`，也可以什么选项都不带来列出所有正在运行的单元。

```
[root@serverX ~]# systemctl
[root@serverX ~]# systemctl list-units
```

列出失败的单元

运行失败的单元可以用带`--failed` 选项的命令显示出来。

```
[root@serverX ~]# systemctl --failed
```

2、管理服务

1) 激活的服务

所有被激活的服务可以同下面这条命令来查看。

```
[root@serverX ~]# systemctl list-units -t service
```

2) 服务状态

```
[root@serverX ~]# systemctl status sshd.service
```

3) 启动一个服务

```
[root@serverX ~]# systemctl start sshd.service
```

4) 停止一个服务

```
[root@serverX ~]# systemctl stop sshd.service
```

5) 重启一个服务

```
[root@serverX ~]# systemctl restart sshd.service
```

6) 重新加载一个服务

在我们需要重新加载服务的配置文件又不想重启这个服务（例如 ssh）时，我们可以用这个命令。

```
[root@serverX ~]# systemctl reload sshd.service
```

3、. 管理引导时的服务

chkconfig 命令被用来管理系统引导时的服务。同样用 systemd 也可以管理引导时的系统服务。

1) 检查服务引导时是否运行

```
[root@serverX ~]# systemctl is-enabled sshd.service
```

2) 让服务在引导时运行

systemctl 命令是这样来 enable (使之在引导时启动) 一个服务的。(这相当于 sysvinit 中的 ‘chkconfig on’)

```
[root@serverX ~]# systemctl enable sshd.service
```

3) 取消服务在引导时运行

```
[root@serverX ~]# systemctl disable sshd.service
```

第九章 配置安全 OpenSSH 服务

9.1 什么是 OpenSSH 安全 Shell (SSH)?

OpenSSH 指的是安全 shell 软件系统的软件实现。SSH 用于安全地运行于一个远程系统 shell。如果你在一个遥远的 Linux 系统提供的 SSH 服务的用户帐户，SSH 通常用于远程登录，运行系统命令。SSH 命令还可以用来运行远程系统上的一个单独的命令。

9.2 ssh 登录简单实例

```
[student@host ~]$ ssh remotehost
student@remotehost's password:
[student@remotehost ~]$ exit
Connection to remotehost closed.
[student@host ~]$
```

```
[student@host ~]$ ssh remoteuser@remotehost
remoteuser@remotehost's password:
[remoteuser@remotehost ~]$
```

```
[student@host ~]$ ssh remoteuser@remotehost hostname
remoteuser@remotehost's password:
remotehost.example.com
[student@host ~]$
```

W 命令显示用户当前登录到计算机。这是特别有用的显示，用户登录使用 SSH 从远程位置，以及他们正在做什么。

```
[student@host ~]$ w -f
USER      TTY      FROM          LOGIN@    IDLE   JCPU   PCPU WHAT
student   tty1     :0           Wed08    2days  1:52m  0.07s pam: gdm-passwo
root      tty6     -           12:33    4:14m  16.27s 15.74s -bash
student   pts/0    :0.0        Wed08    5:11   1.63s  1.60s /usr/bin/gnome-
student   pts/1    :0.0        Wed08    43:44  14.48s 13.81s vim hello.c
student   pts/3    :0.0        Wed14    0.00s  0.06s  0.06s w
visitor   pts/6    server2.example. 09:22    3:14   0.02s  0.02s -bash
```

9.3 SSH 主机密钥

SSH 的安全通信通过公共密钥加密。当一个 SSH 客户端连接到 SSH 服务器，在客户端登录时，服务器发送一份公开密钥。这是用于通信信道建立安全的加密和认证服务器到客户端。

- » Host IDs are stored in `~/.ssh/known_hosts` on your local client system:

创建 SSH 密钥

```
$ cat ~/.ssh/known_hosts
remotehost,192.168.0.101 ssh-rsa AAAAB3NzaC1E...
```

- » Host keys are stored in `/etc/ssh/ssh_host_key*` on the SSH server.

```
$ ls /etc/ssh/*key*
ssh_host_dsa_key      ssh_host_key      ssh_host_rsa_key
ssh_host_dsa_key.pub  ssh_host_key.pub  ssh_host_rsa_key.pub
```

9.4 配置 SSH 公钥认证

用户可以验证 SSH 登录不使用公共密钥认证密码。SSH 允许用户进行身份验证使用私有的公钥方案。这意味着两个生成密钥，私有密钥和公共密钥。私钥文件作为身份验证凭据，像一个密码，必须保密和安全。公共密钥复制到用户要登录，和用于验证的密钥系统中。因此，你可以用你的密钥进行身份验证，不需要每次键入一个密码登录系统，但仍然安全。

密钥的生成是通过使用 `ssh-keygen` 命令。这会产生私钥`~/.ssh/id_rsa` 和 SSH 公钥 `id_rsa.pub`。

```
[student@desktopX ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/student/.ssh/id_rsa): Enter
Created directory '/home/student/.ssh'.
Enter passphrase (empty for no passphrase): redhat
Enter same passphrase again: redhat
Your identification has been saved in /home/student/.ssh/id_rsa.
Your public key has been saved in /home/student/.ssh/id_rsa.pub.
The key fingerprint is:
a4:49:cf:fb:ac:ab:c8:ce:45:33:f2:ad:69:7b:d2:5a student@desktopX.example.com
The key's randomart image is:
+--[ RSA 2048]----+
| |
| |
| . .
| . * S
| + + .
| o.E
| o oo+oo
| .=.*+ooo
+-----+
```

```
[student@desktopX ~]$ ssh-copy-id root@desktopY
```

When the key is copied to another system using **ssh-copy-id**, it copies the `~/.ssh/id_rsa.pub` file by default.

» Use **ssh-copy-id** to copy the public key to the correct location on a remote system. For example:

```
[student@desktopX ~]$ ssh-copy-id -i ~/.ssh/id_rsa.pub root@serverX.example.com
```

9.4 自定义 SSH 服务配置

OpenSSH 服务器配置文件 `/etc/ssh/sshd_config`

1、禁止 root 用户使用 SSH 登录

The OpenSSH server has an internal configuration file setting to prohibit a system login as user root, which is commented out by default in the `/etc/ssh/sshd_config` file:

```
#PermitRootLogin yes
```

By enabling the previous option in the `/etc/ssh/sshd_config` configuration file as follows, the root user will be unable to log into the system using the `ssh` command after the `sshd` service has been restarted: 默认密钥复制

```
PermitRootLogin no
```

The `sshd` service has to be restarted to put the changes into effect:

指定密钥复制

```
[root@serverX ~]# systemctl restart sshd
```

Another option is to only allow key-based ssh login as root with:

```
PermitRootLogin without-password
```

2、禁止使用 SSH 密码认证

There is an option in the `/etc/ssh/sshd_config` configuration file which turns on password authentication by default:

```
PasswordAuthentication yes
```

To prevent password authentication, the `PasswordAuthentication` option has to be set to `no` and the sshd service needs to be restarted:

```
PasswordAuthentication no
```

Keep in mind that whenever you change the `/etc/ssh/sshd_config` file, the sshd service has to be restarted:

```
[root@serverX ~]# systemctl restart sshd
```

第十章 分析存储日志

10.1 系统日志架构

System logging 记录内核和进程发生事件，日志有利于我们对系统审核和排错。默认情况下，日志存储在 `/var/log` 目录中。

RHEL7 系统日志消息由 `systemd-journald` 和 `rsyslog` 两个服务负责。

`Systemd-journald` 守护进程提供从内核、启动阶段、标准输出、守护进程启动运行错误及系统日志等中收集信息。

`Rsyslog` 服务负责按类型和优先级对日志排序，并写入 `/var/log` 目录中。

Overview of system log files

Log file	Purpose
<code>/var/log/messages</code>	Most syslog messages are logged here. The exceptions are messages related to authentication and email processing, that periodically run jobs, and those which are purely debugging-related.
<code>/var/log/secure</code>	The log file for security and authentication-related messages and errors.
<code>/var/log/maillog</code>	The log file with mail server-related messages.
<code>/var/log/cron</code>	The log file related to periodically executed tasks.
<code>/var/log/boot.log</code>	Messages related to system startup are logged here.

10.2 审核日志文件

`/etc/rsyslog.conf`

Overview of syslog priorities

Code	Priority	Severity
0	emerg	System is unusable.
1	alert	Action must be taken immediately.
2	crit	Critical condition.
3	err	Non-critical error condition.
4	warning	Warning condition.
5	notice	Normal but significant event.
6	info	Informational event.
7	debug	Debugging-level message.

日志文件轮询

日志是“轮询”是通过 logrotate，将满足条件的旧的日志文件单独打包成为 messages-20141030 的文件，后面的数字代表打包日期。条件的配置文件脚本在/etc/logrotate.d/ 中。

一定数量的轮询后，通常为四周后，旧的日志文件被丢弃，以释放磁盘空间。一个 cron 作业运行程序每日 logrotate 看日志需要轮询。

分析一个日志条目

系统日志由 rsyslog 最新的消息写在文件末尾。在日志文件由 rsyslog 所有日志条目记录在一个标准的格式。下面的例子将说明解剖一个日志文件消息在/var/log/messages 安全日志文件：

```
①Feb 11 20:11:48 ②localhost ③sshd[1433]: ④Failed password for student from
172.25.0.10 port 59344 ssh2
```

1 日志记录时间戳

2 主机名

3 程序或进程发送日志

4 实际发送的消息

使用 tail -f 命令监控日志

```
[root@foundation49 logrotate.d]# tail -f /var/log/messages -n 5
Aug 19 14:00:01 foundation49 systemd: Started Session 6 of user root.
Aug 19 14:01:01 foundation49 systemd: Created slice user-0.slice.
Aug 19 14:01:01 foundation49 systemd: Starting Session 7 of user root.
```

10.3 审查系统日志条目

在红帽企业 Linux 7，该 **systemd journal** 默认情况下存储在 **/run/log**，系统重新启动后内容

会被清除。要求转换角色为 root 进行查看。

```
[root@serverX ~]# journalctl
Feb 13 10:01:01 server1 run-parts(/etc/cron.hourly)[8678]: starting @yum-hourly.cron
Feb 13 10:01:01 server1 run-parts(/etc/cron.hourly)[8682]: finished @yum-hourly.cron
Feb 13 10:10:01 server1 systemd[1]: Starting Session 725 of user root.
Feb 13 10:10:01 server1 systemd[1]: Started Session 725 of user root.
Feb 13 10:10:01 server1 CROND[8687]: (root) CMD (/usr/lib64/sa/sa1 1 1)
```

```
[root@foundation49 ~]# journalctl --priority err
-- Logs begin at Tue 2014-08-19 13:17:43 CST, end at Tue 2014-08-19 14:52:26 CST. --
Aug 19 13:17:46 foundation49.ilt.example.com kernel: sd 6:0:0:0: [sda] Assuming drive cache: writ
Aug 19 13:17:46 foundation49.ilt.example.com kernel: sd 6:0:0:0: [sda] Assuming drive cache: writ
Aug 19 13:17:46 foundation49.ilt.example.com kernel: sd 6:0:0:0: [sda] Assuming drive cache: writ
Aug 19 13:17:50 foundation49.ilt.example.com kernel: end_request: I/O error, dev fd0, sector 0
Aug 19 13:17:50 foundation49.ilt.example.com kernel: end_request: I/O error, dev fd0, sector 0

[root@foundation49 ~]# journalctl _UID=1000 -n 5
-- Logs begin at Tue 2014-08-19 13:17:43 CST, end at Tue 2014-08-19 14:52:26 CST. --
Aug 19 14:49:43 foundation49.ilt.example.com su[34964]: pam_unix(su-1:auth): authentication failu
Aug 19 14:49:43 foundation49.ilt.example.com su[34964]: pam_succeed_if(su-1:auth): requirement "u
Aug 19 14:49:45 foundation49.ilt.example.com su[34964]: FAILED SU (to root) kiosk on pts/0
Aug 19 14:52:26 foundation49.ilt.example.com su[35051]: (to root) kiosk on pts/0

[root@foundation49 ~]# journalctl --since 13:00:00 --until 14:00:00 _SYSTEMD_UNIT="sshd.service"
-- Logs begin at Tue 2014-08-19 13:17:43 CST, end at Tue 2014-08-19 14:52:26 CST. --
Aug 19 13:17:58 foundation49.ilt.example.com sshd-keygen[840]: Generating SSH2 RSA host key: [ 0
Aug 19 13:17:58 foundation49.ilt.example.com sshd-keygen[840]: Generating SSH2 ECDSA host key: [ 0
Aug 19 13:17:58 foundation49.ilt.example.com sshd[895]: Server listening on 0.0.0.0 port 22.
Aug 19 13:17:58 foundation49.ilt.example.com sshd[895]: Server listening on :: port 22.
```

10.4 保护系统的日志

配置文件 ^a	/etc/systemd/journal.conf	所有
存储路径 ^b	/var/log/journal	最后5行
配置流程 ^c	优先级 (err 错误 info 信息)	
-b	启动消息	
-f	监控	
--since “2014-02-10 20:30:00” --until “2014-02-10 21:30:00” 指定范围		
--since today 显示今天		
-o	更改日志的输出模式	

```
[root@serverX ~]# journalctl | head -2
-- Logs begin at Wed 2014-03-05 15:13:37 CST, end at Thu 2014-03-06 21:57:54 CST. --
Mar 05 15:13:37 serverX.example.com systemd-journal[94]: Runtime journal is using 8.0M
(max 277.8M, leaving 416.7M of free 2.7G, current limit 277.8M).
```

The systemd journal can be made persistent by creating the directory `/var/log/journal` as user root:

```
[root@serverX ~]# mkdir /var/log/journal
```

Ensure that the `/var/log/journal` directory is owned by the root user and group `systemd-journal`, and has the permissions 2755.

```
[root@serverX ~]# chown root:systemd-journal /var/log/journal
[root@serverX ~]# chmod 2755 /var/log/journal
```

Either a reboot of the system or sending the special signal `USR1` as user root to the `systemd-journald` process is required.

```
[root@serverX ~]# killall -USR1 systemd-journald
```

Since the systemd journal is now persistent across reboots, `journalctl -b` can reduce the output by only showing the log messages since the last boot of the system.

```
[root@serverX ~]# journalctl -b
```

10.4 保持准确的时间

正确的时间同步，在多个系统环境中是非常重要的。网络时间协议（NTP）是机器提供从因特网上获得正确的时间信息的标准方法。一台机器可以从互联网上如 NTP 池项目公共 NTP 服务获得准确的时间信息。一个高质量的硬件时钟服务是本地客户效准时间另一种选择。

```
[student@serverX ~]$ timedatectl
    Local time: Thu 2014-02-13 02:16:15 EST
    Universal time: Thu 2014-02-13 07:16:15 UTC
        RTC time: Thu 2014-02-13 07:16:15
      Timezone: America/New_York (EST, -0500)
    NTP enabled: yes
  NTP synchronized: no
    RTC in local TZ: no
      DST active: no
Last DST change: DST ended at
                  Sun 2013-11-03 01:59:59 EDT
                  Sun 2013-11-03 01:00:00 EST
Next DST change: DST begins (the clock jumps one hour forward) at
                  Sun 2014-03-09 01:59:59 EST
                  Sun 2014-03-09 03:00:00 EDT
```

```
[root@serverX ~]# timedatectl set-timezone America/Phoenix
[root@serverX ~]# timedatectl
    Local time: Thu 2014-02-13 00:23:54 MST
    Universal time: Thu 2014-02-13 07:23:54 UTC
        RTC time: Thu 2014-02-13 07:23:53
       Timezone: America/Phoenix (MST, -0700)
      NTP enabled: yes
     NTP synchronized: no
      RTC in local TZ: no
        DST active: n/a
```

To change the current time and date settings with the **timedatectl** command, the **set-time** option is available. The time is specified in the "YYYY-MM-DD hh:mm:ss" format, where either date or time can be omitted. To change the time to 09:00:00, run:

```
[root@serverX ~]$ timedatectl set-time 9:00:00
[root@serverX ~]$ timedatectl
    Local time: Thu 2014-02-13 09:00:27 MST
    Universal time: Thu 2014-02-13 16:00:27 UTC
        RTC time: Thu 2014-02-13 16:00:28
       Timezone: America/Phoenix (MST, -0700)
      NTP enabled: yes
     NTP synchronized: no
      RTC in local TZ: no
        DST active: n/a
```

The **set-ntp** option enables or disables NTP synchronization for automatic time adjustment. The option requires either a **true** or **false** argument to turn it on or off. To turn on NTP synchronization, run:

```
[student@desktopX ~]$ timedatectl set-ntp true
```

Chrony

chrony 包含两个用来维持计算机系统时钟准确性的程序，这两个程序命名为 chronyd 和 chronyc。

chronyd 是一个在系统后台运行的守护进程。他根据网络上其他时间服务器时间来测量本机时间的偏移量从而调整系统时钟。对于孤立系统，用户可以手动周期性的输入正确时间（通过 chronyc）。在这两种情况下，chronyd 决定计算机快慢的比例，并加以纠正。chronyd 实现了 NTP 协议并且可以作为服务器或客户端。

chronyc 是用来监控 chronyd 性能和配置其参数的用户界面。他可以控制本机及其他计算机上运行的 chronyd 进程。

配置和监控 chronyd

守护进程: hronyd

配置文件路径: /etc/chrony.conf

To reconfigure the **chrony**d server to synchronize with classroom.example.com instead of the default servers configured in the **/etc/chrony.conf**, remove the other server entries and replace them with the following configuration file entry:

```
# Use public servers from the pool.ntp.org project.
server classroom.example.com iburst
```

After pointing **chrony**d to the local time source, classroom.example.com, the service needs to be restarted:

```
[root@serverX ~]# systemctl restart chronyd
```

The **chronyc** command acts as a client to the **chrony**d service. After setting up NTP synchronization, it is useful to verify the NTP server was used to synchronize the system clock. This can be achieved with the **chronyc sources** command or, for more verbose output with additional explanations about the output, **chronyc sources -v**:

```
[root@serverX ~]$ chronyc sources -v
210 Number of sources = 1

--- Source mode  '^' = server, '=' = peer, '#' = local clock.
/ .- Source state '*' = current synced, '+' = combined , '-' = not combined,
| /   '?' = unreachable, 'x' = time may be in error, '~' = time too variable.
||          .- xxxx [ yyyy ] +/- zzzz
||          |   xxxx = adjusted offset,
||          |   yyyy = measured offset,
||          |   zzzz = estimated error.
||          Log2(Polling interval) ..
||          \
||          MS Name/IP address      Stratum Poll  Reach LastRx Last sample
=====
^* classroom.example.com        8       6    17    23   -497ns[-7000ns] +/-  956us
```

The * character in the S (Source state) field indicates that the classroom.example.com server has been used as a time source and is the NTP server the machine is currently synchronized to.

第十一章 红帽企业 Linux 网络管理

11.1 计算机网络的基本概念

理解网卡名称传统上, Linux 的网络接口列举为 eth0, eth1, eth2, 等等。在红帽企业 Linux7 的默认命名行为是基于固件基础上分配的, 设备拓扑和设备类型的固定名称。

接口名称有以下特点:

以太网有线接口为 en, 无线局域网接口为 wl, 无线广域网接口为 ww。

如果固定名称不能确定的, 传统的名称, 如 ethN 的将被使用。

例如, 第一个嵌入式网络接口可能被命名为 ENO1 和 PCI 卡网络接口可能被命名为 enp2s0。新名称更容易地分辨出端口, 如果用户知道这两个名字之间的关系, 但权衡的是, 用户不能假设一个系统有一个接口调用接口 eth0。

10.2 验证网络配置

1、查看 IP 地址

```
[student@desktopX ~]$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,1UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:00:00:0a brd ff:ff:ff:ff:ff:ff
    3inet 172.25.0.10/24 brd 4172.25.0.255 scope global eth0
        valid_lft forever preferred_lft forever
    5inet6 fe80::5054:ff:fe00:b/64 scope link
        valid_lft forever preferred_lft forever
```

- 1.接口状态 2.硬件（MAC）地址 3.I Pv4 地址与子网
4.广播地址，范围，和设备的名称 5.I Pv6 信息

ip 命令还可以用于显示关于网络性能的统计信息。所接收（RX）和发送（TX）数据包，错误和下降计数器可以被用来识别所造成的堵塞，低存储器，和超支网络问题。

```
[student@desktopX ~]$ ip -s link show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:00:00:0a brd ff:ff:ff:ff:ff:ff
        RX: bytes packets errors dropped overrun mcast
        269850 2931 0 0 0
        TX: bytes packets errors dropped carrier collsns
        300556 3250 0 0 0
```

2、排除路由问题

```
[root@foundation50 ~]# ping -c 2 172.25.254.250
PING 172.25.254.250 (172.25.254.250) 56(84) bytes of data.
64 bytes from 172.25.254.250: icmp_seq=1 ttl=64 time=0.558 ms
64 bytes from 172.25.254.250: icmp_seq=2 ttl=64 time=0.840 ms

--- 172.25.254.250 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.558/0.699/0.840/0.141 ms

[root@foundation50 ~]# tracepath 172.25.254.250
1: foundation50.ilt.example.com 0.167ms pmtu 1500
1: foundation0.ilt.example.com 11.855ms reached
1: foundation0.ilt.example.com 11.842ms reached
Resume: pmtu 1500 hops 1 back 64
```

```
[root@foundation50 ~]# ip route
default via 172.25.254.254 dev br0 proto static metric 1024
172.25.50.0/24 dev br0 proto kernel scope link src 172.25.50.250
172.25.254.0/24 dev br0 proto kernel scope link src 172.25.254.50
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1
```

3、排除端口与服务问题

[student@desktopX ~]\$ ss -ta					
State	Recv-Q	Send-Q	Local Address:Port	Peer Address:Port	
LISTEN	0	128	*:sunrpc	*:*	
LISTEN	0	128	❶ *:ssh	*:*	
LISTEN	0	100	❷ 127.0.0.1:smtp	*:*	
LISTEN	0	128	*:36889	*:*	
ESTAB	0	0	❸ 172.25.X.10:ssh	172.25.254.254:59392	
LISTEN	0	128	:::sunrpc	:::*	
LISTEN	0	128	❹ ::::ssh	:::*	
LISTEN	0	100	❺ ::1:smtp	:::*	
LISTEN	0	128	:::34946	:::*	

- 1.用于 SSH 端口监听所有的 IPv4 地址。“*”来表示“所有的”当引用的 IPv4 地址或端口。
- 2.用于 SMTP 端口正在监听 127.0.0.1 IPv4 环回接口。
- 3.建立的 SSH 连接在 172.25.x.10 界面和源于一个与 172.25.254.254 地址系统。
- 4.用于 SSH 端口监听所有的 IPv6 地址。“::”的语法被用来代表所有 IPv6 接口。

```
[root@foundation50 ~]# ss -ta |grep ssh
LISTEN      0      128                      *:ssh
LISTEN      0      128                      ::::ssh

[root@foundation50 ~]# ss -tunlp|grep ssh
tcp   LISTEN      0      128          *:22      *:*      users:(("sshd",878,3))
tcp   LISTEN      0      128          :::22      *:*      users:(("sshd",878,4))
```

服务端口配置文件 /etc/service

```

# /etc/services:
# $Id: services,v 1.55 2013/04/14 ovasik Exp $
#
# Network services, Internet style
# IANA services version: last updated 2013-04-10
#
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, most entries here have two entries
# even if the protocol doesn't support UDP operations.
# Updated from RFC 1700, ``Assigned Numbers'' (October 1994). Not all ports
# are included, only the more common ones.
#
# The latest IANA port assignments can be gotten from
#     http://www.iana.org/assignments/port-numbers
# The Well Known Ports are those from 0 through 1023.
# The Registered Ports are those from 1024 through 49151
# The Dynamic and/or Private Ports are those from 49152 through 65535
#
# Each line describes one service, and is of the form:
#
# service-name    port/protocol [aliases ...]      [# comment]

tcpmux          1/tcp                           # TCP port service multiplexer
tcpmux          1/udp                           # TCP port service multiplexer
rje             5/tcp                           # Remote Job Entry
rje             5/udp                           # Remote Job Entry
echo            7/tcp                           sink null
echo            7/udp                           sink null
discard         9/tcp                           users
discard         9/udp                           users
sysstat         11/tcp                          users
sysstat         11/udp                          users
daytime          13/tcp                          users
daytime          13/udp                          users

```

10.3 使用 nmcli 配置网络

1、NetworkManager

2、使用 nmcli 查看网络信息

```

[root@desktopX ~]# nmcli con show
NAME           UUID                                  TYPE      DEVICE
static-eth0    f3e8dd32-3c9d-48f6-9066-551e5b6e612d 802-3-ethernet  eth0
System eth0    5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03 802-3-ethernet  --
guest          f601ca8a-6647-4188-a431-dab48cc63bf4  802-11-wireless wlp3s0
[root@desktopX ~]# nmcli con show --active
NAME           UUID                                  TYPE      DEVICE
static-eth0    f3e8dd32-3c9d-48f6-9066-551e5b6e612d 802-3-ethernet  eth0
guest          f601ca8a-6647-4188-a431-dab48cc63bf4  802-11-wireless wlp3s0

```

--active 查看活动的设备

```
[root@desktopX ~]# nmcli con show "static-eth0"
...
ipv4.method:           manual
ipv4.dns:              172.25.254.254, 8.8.8.8
ipv4.dns-search:
ipv4.addresses:        { ip = 172.25.X.10/24, gw = 172.25.X.254 }
ipv4.routes:
ipv4.ignore-auto-routes: no
ipv4.ignore-auto-dns:   no
ipv4.dhcp-client-id:   --
ipv4.dhcp-send-hostname: yes
ipv4.dhcp-hostname:    --
ipv4.never-default:   no
ipv4.may-fail:         yes
ipv6.method:           auto
...
```

3、使用 nmcli 创建网络连接

Examples of creating new connections

Follow along with the next steps while your instructor discusses **nmcli** syntax.

1. Define a new connection named "default" which will autoconnect as an Ethernet connection on the eth0 device using DHCP.

```
[root@desktopX ~]# nmcli con add con-name "default" type ethernet ifname eth0
```

2. Create a new connection named "static" and specify the IP address and gateway. Do not autoconnect.

```
[root@desktopX ~]# nmcli con add con-name "static" ifname eth0 autoconnect no type
ethernet ip4 172.25.X.10/24 gw4 172.25.X.254
```

3. The system will autoconnect with the DHCP connection at boot. Change to the static connection.

```
[root@desktopX ~]# nmcli con up "static"
```

4. Change back to the DHCP connection.

```
[root@desktopX ~]# nmcli con up "default"
```

```
[root@desktopX ~]# nmcli con show "static"
connection.id:                         static
connection.uuid:                        f3e8dd32-3c9d-48f6-9066-551e5b6e612d
connection.interface-name:               eth0
connection.type:                        802-3-ethernet
connection.autoconnect:                 yes
connection.timestamp:                  1394905322
connection.read-only:                  no
...
```

Examples of connection modifications

Follow along with the next steps while your instructor discusses **nmcli** syntax.

1. Turn off autoconnect.

```
[root@desktopX ~]# nmcli con mod "static" connection.autoconnect no
```

2. Specify a DNS server.

```
[root@desktopX ~]# nmcli con mod "static" ipv4.dns 172.25.X.254
```

3. Some configuration arguments may have values added or removed. Add a +/- symbol in front of the argument. Add a secondary DNS server.

```
[root@desktopX ~]# nmcli con mod "static" +ipv4.dns 8.8.8.8
```

4. Replace the static IP address and gateway.

```
[root@desktopX ~]# nmcli con mod "static" ipv4.addresses "172.25.X.10/24
172.25.X.254"
```

5. Add a secondary IP address without a gateway.

```
[root@desktopX ~]# nmcli con mod "static" +ipv4.addresses 10.10.10.10/16
```

nmcli commands

Command	Use
nmcli dev status	List all devices.
nmcli con show	List all connections.
nmcli con up "<ID>"	Activate a connection.
nmcli con down "<ID>"	Deactivate a connection. The connection will restart if autoconnect is yes.
nmcli dev dis <DEV>	Bring down an interface and temporarily disable autoconnect.
nmcli net off	Disable all managed interfaces.
nmcli con add ...	Add a new connection.
nmcli con mod "<ID>" ...	Modify a connection.
nmcli con del "<ID>"	Delete a connection.

11.4 图形化管理网卡配置

通过 nm-connection-editor 命令打开图形化进行管理

11.5 编辑网络配置文件

配置文件路径: /etc/sysconfig/network-scripts/ifcfg-*

配置文件结构

Configuration Options for ifcfg File

Static	Dynamic	Either
BOOTPROTO=none	BOOTPROTO=dhcp	DEVICE=eth0
IPADDR0=172.25.X.10		NAME="System eth0"
PREFIX0=24		ONBOOT=yes
GATEWAY0=172.25.X.254		UUID=f3e8dd32-3...
DEFROUTE=yes		USERCTL=yes
DNS1=172.25.254.254		

11.5 配置主机名和域名解析

通过 hostname 查看主机名

```
[root@desktopX ~]# hostname
desktopX.example.com
```

主机名配置文件路径: /etc/hostname

配置主机名命令: hostnamectl

```
[root@desktopX ~]# hostnamectl set-hostname desktopX.example.com
[root@desktopX ~]# hostnamectl status
    Static hostname: desktopX.example.com
              Icon name: computer
                Chassis: n/a
      Machine ID: 9f6fb63045a845d79e5e870b914c61c9
        Boot ID: aa6c3259825e4b8c92bd0f601089ddf7
  Virtualization: kvm
Operating System: Red Hat Enterprise Linux Server 7.0 (Maipo)
      CPE OS Name: cpe:/o:redhat:enterprise_linux:7.0:beta:server
          Kernel: Linux 3.10.0-97.el7.x86_64
        Architecture: x86_64
[root@desktopX ~]# cat /etc/hostname
desktopX.example.com
```

本地域名解析配置文件路径: /etc/hosts

DNS 服务器配置文件路径: /etc/resolv.conf

```
[root@desktopX ~]# cat /etc/resolv.conf
# Generated by NetworkManager
domain example.com
search example.com
nameserver 172.25.254.254
```

NetworkManager will update the `/etc/resolv.conf` file using DNS settings in the connection configuration files. Use the `nmcli` to modify the connections.

```
[root@desktopX ~]# nmcli con mod ID ipv4.dns IP
[root@desktopX ~]# nmcli con down ID
[root@desktopX ~]# nmcli con up ID
[root@desktopX ~]# cat /etc/sysconfig/network-scripts/ifcfg-ID
...
DNS1=8.8.8.8
...
```

The default behavior of `nmcli con mod ID ipv4.dns IP` is to replace any previous DNS settings with the new IP list provided. A `+/-` symbol in front of the `ipv4.dns` argument will add or remove an individual entry.

```
[root@desktopX ~]# nmcli con mod ID +ipv4.dns IP
```

第十二章 文件归档和系统复制

12.1 使用 tar 管理压缩档案

1、什么是 tar ?

归档和压缩的文件是有用的创建备份和在网络上传送的数据方法。一个用于创建和备份档案工作的最古老和最常见的命令是 tar 命令。Tar 命令能把多个文件打包成一个文件，在打包同时档案可以被压缩使用 gzip, bzip2, 或 xz 压缩。

2、tar 命令选项 ?

c (创建一个归档)

t (列表存档的内容)

x (提取档案)

f (指定文件名)

v (显示过程)

p (保存文件权限)

3、管理 tar 包

1) 创建一个 tar 包

```
[user@host ~]# tar cf archive.tar file1 file2 file3
[user@host ~]# ls archive.tar
archive.tar
```

2) 查看 tar 包中的文件

```
[root@host ~]# tar tf /root/etc.tar
etc/
etc/fstab
etc/crypttab
etc/mtab
...
```

3) 提取 tar 包中的文件

Extract the archive `/root/etc.tar` to the `/root/etcbackup` directory:

```
[root@host ~]# mkdir /root/etcbackup
[root@host ~]# cd /root/etcbackup
[root@host etcbackup]# tar xf /root/etc.tar
```

4、创建一个压缩的 tar 归档文件

创建一个压缩的 tar 归档文件，下面的 tar 选项可以指定：

- z gzip 压缩 (`filename.tar.gz` 或 `filename.tgz`)
- j bzip2 压缩 (`filename.tar.bz2`)
- J xz 压缩 (`filename.tar.xz`)

Create (c option) a gzip-compressed (z option) tar archive `/root/etcbackup.tar.gz` of the `/etc` directory on serverX:

```
[root@serverX ~]$ tar czf /root/etcbackup.tar.gz /etc
```

Create (c option) a bzip2-compressed (j option) tar archive `/root/logbackup.tar.bz2` of the `/var/log` directory on serverX:

```
[root@serverX ~]$ tar cjf /root/logbackup.tar.bz2 /var/log
```

Create (c option) a xz-compressed (J option) tar archive `/root/sshconfig.tar.bz2` of the `/etc/ssh` directory on serverX:

```
[root@serverX ~]$ tar cJf /root/sshconfig.tar.xz /etc/ssh
```

5、解压一个压缩的 tar 归档文件

Extract (x option) the contents of a gzip-compressed (z option) tar archive named **/root/etcbackup.tar.gz** to the directory **/tmp/etcbackup**:

```
[root@serverX ~]$ mkdir /tmp/etcbackup
[root@serverX ~]$ cd /tmp/etcbackup
[root@serverX etcbackup]$ tar xzf /root/etcbackup.tar.gz
```

Extract (x option) the contents of a bzip2-compressed (j option) tar archive named **/root/logbackup.tar.bz2** to the directory **/tmp/logbackup**:

```
[root@serverX ~]$ mkdir /tmp/logbackup
[root@serverX ~]$ cd /tmp/logbackup
[root@serverX logbackup]# tar xjf /root/logbackup.tar.bz2
```

Extract (x option) the contents of a xz-compressed (J option) tar archive named **/root/sshbackup.tar.xz** to the directory **/tmp/sshbackup**:

```
[root@serverX ~]$ mkdir /tmp/sshbackup
[root@serverX ~]$ cd /tmp/sshbackup
[root@serverX sshbackup]# tar xJf /root/sshbackup.tar.xz
```

tar 参数总结

Overview of tar options

Option	Meaning
c	Create a new archive.
x	Extract from an existing archive.
t	List the contents of an archive.
v	Verbose; shows which files get archived or extracted.
f	File name; this option needs to be followed by the file name of the archive to use/create.
p	Preserve the permissions of files and directories when extracting an archive, without subtracting the umask.
z	Use gzip compression (.tar.gz).
j	Use bzip2 compression (.tar.bz2). bzip2 typically achieves a better compression ratio than gzip .
J	Use xz compression (.tar.xz). xz typically achieves a better compression ratio than bzip2 .

12.2 系统之间复制文件安全

1、scp 命令

ssh 命令是有用在远程系统安全运行 shell。它也可以被用于安全地复制文件从一台机到另一台机器。SCP 命令将文件从一台远程主机到本地系统或从本地到远程主机。它利用 SSH

服务器认证和加密数据传输。

This example shows how to copy the local files on desktopX, `/etc/yum.conf` and `/etc/hosts`, securely to the account student on the remote system serverX into the directory `/home/student/`:

```
[student@desktopX ~]$ scp /etc/yum.conf /etc/hosts serverX:/home/student
student@serverX's password: student
yum.conf                                              100%   813      0.8KB/s  00:00
hosts                                                 100%   227      0.2KB/s  00:00
```

A user can copy a file from a remote account on a remote machine to the local file system with `scp`. In this example, copy the file `/etc/hostname` from the account student on the serverX machine to the local directory `/home/student/`.

```
[student@desktopX ~]$ scp serverX:/etc/hostname /home/student/
student@serverX's password: student
hostname                                              100%    22      0.0KB/s  00:00
```

To copy a whole directory tree recursively, the `-r` option is available. In the following example, the remote directory `/var/log` on serverX is copied recursively to the local directory `/tmp/` on desktopX. To be able to read all the contents of the `/etc` directory, the root user must connect to the remote location.

```
[student@desktopX ~]$ scp -r root@serverX:/var/log /tmp
root@serverX's password: redhat
...
```

2、SFTP

一种交互式工具,可以使用的命令上传或下载文件到 SSH 服务器,。会话与 SFTP 是类似于经典的 FTP 会话使,但数据传输安全和加密。sftp 访问的目标是远程用户的家目录

```
[student@desktopX ~]$ sftp serverX
student@serverX's password: student
Connected to serverX.
sftp>
```

```
sftp> mkdir hostbackup
sftp> cd hostbackup
sftp> put /etc/hosts
Uploading /etc/hosts to /home/student/hostbackup/hosts
/etc/hosts                                              100%   227      0.2KB/s  00:00
sftp>
```

```
sftp> get /etc/yum.conf
Fetching /etc/yum.conf to yum.conf
/etc/yum.conf                                              100%   813      0.8KB/s  00:00
sftp> exit
[student@desktopX ~]$
```

10.3 使用 rsync 安全实现系统间文件同步

正确、有效的备份方案是保障系统及数据安全的重要手段,在服务器中通常会结合计划任务、Shell 脚本来执行本地备份。为了进一步提高备份的可靠性,使用异地备份也是非常有必要的。本章将要 rsync (Remote Sync, 远程同步) 工具的使用,以实现快速、安全、高

效的异地备份，比如构建 Web 镜像站点。

Rsync 是一个开源的快速备份工具，可以在不同主机之间镜像同步整个目录树，支持增量备份、保持链接和权限，且采用优化的同步算法、传输前执行压缩，因此非常适用于异地备份、镜像服务器等应用。

Rsync 的官方站点位于 <http://rsync.samba.org/>，目前最新版本是 3.0.9，由 Wayne Davison 进行维护。作为一种最常用的文件备份工具，rsync 往往是 Linux/UNIX 系统默认的安装的基本组件之一。

Rsync 命令选项

- r, 递归整个目录树的同步
- l, 同步符号链接
- p, 保留权限
- t, 保留保存时间
- g, 保留所属组所有权
- o, 保存文件的所有者
- H, 硬链接处理
- a, 全部
- v, 显示过程

1、在同服务器两个目录间同步

The basic way of using **rsync** is to synchronize two local folders. In the following example, the **/var/log** directory gets a synchronized copy in the **/tmp** folder. The **log** directory with its content is created in the **/tmp** directory.

```
[student@desktopX ~]$ su -
Password: redhat
[root@desktopX ~]# rsync -av /var/log /tmp
...
```

To only synchronize the content of a folder without newly creating the folder in the target directory, a trailing slash needs to be added at the end of the source directory. In this example, the **log** directory is not created in the **/tmp** folder. Only the content of the **/var/log/** directory is synchronized into the **/tmp** folder.

```
[root@desktopX ~]# rsync -av /var/log/ /tmp
...
```

rsync -av /var/log /tmp //同步时会在 tmp 目录中创建 log 目录。
 rsync -av /var/log/ /tmp //同步 log 目录下的内容到 /tmp 目录下。

2、在两个不同服务器间实现文件同步

```
[root@desktopX ~]$ rsync -av /var/log serverX:/tmp
root@serverX's password: redhat
...
```

In the same way, the remote folder **/var/log** on serverX can be synchronized to the local directory **/tmp** on desktopX:

```
[root@desktopX ~]$ rsync -av serverX:/var/log /tmp
root@serverX's password: redhat
...
```

第十三章 安装和升级软件包

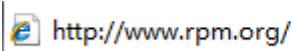
13.1 安装系统软件更新订阅

13.2 RPM 软件包和 YUM

1、RPM 包管理的基本概念

RPM 最早是由 Red Hat 公司提出的软件包管理标准，后来随着版本的升级又融入了许多其他优秀特性，成为了 Linux 中公认的软件包管理标准。目前使用 RPM 作为软件包管理格式的主要 Linux 发行版有 Red Hat Linux、Suse Linux 和 Mandriva Linux。

RPM 现在是 RPM Package Manager 的缩写（原来是 Redhat Package Manager 的缩写），由 RPM 社区负责维护，可以按照以下地址登录到 RPM 的官方站点查询 RPM 软件包格式的详细资料。



RPM 安装包文件的名称拥有固定的格式，安装光盘中的 RPM 软件包被集中保存在光盘文件系统的“Server/Packages”目录中。

例：挂载 RHEL 系统安装光盘，“bash-4.2.45.5.el7.x86_64.rpm”包文件为例来说明 RPM 软件包的文件名格式。

```
[root@localhost ~]# mount /dev/cdrom /mnt
mount: /dev/sr0 写保护，将以只读方式挂载
[root@localhost ~]#
[root@localhost ~]# cd /mnt/Packages/
[root@localhost Packages]#
[root@localhost Packages]# ll bash*
-r--r--r--. 79 liweijie liweijie 1031480 4月 1 21:24 bash-4.2.45.5.el7.x86_64.rpm
-r--r--r--. 123 liweijie liweijie 87360 4月 1 21:24 bash-completion-2.1-6.el7.noarch.rpm
[root@localhost Packages]#
```

在文件“bash-4.2.45.5.el7.x86_64.rpm”的名称中，“bash”是软件的名称；“4.2.45.5.”是软件版本号，“x86_64”是软件所运行的硬件平台，“x86_64”代表软件需要运行在 Intel 公司的“x86”以上的 64 位 CPU 处理器上；“rpm”是文件的扩展名，用以标识当前文件是 RPM 格式的软件包。

2、YUM 管理软件包

安装额外的软件包和更新系统，一般都是从网络包仓库安装，最经常通过红帽子订阅管理服务。

rpm 命令可以用来安装，更新，删除，和查询的 RPM 包。然而，它没有自动解决软件包依赖关系能力。PackageKit 和 YUM 工具是 RPM 前端应用，可用于安装单独的包或包的集合（有时被称为软件包组）。yum 命令搜索大量的库包和它们的依赖，所以可以解决依赖问题，依赖包自动一并安装。

YUM 主配置文件/etc/yum.conf 里，额外的存储仓库配置文件位于/etc/yum.repos.d 目录中。存储仓库配置文件包括 Repo ID（在方括号中），一个包的名称和版本库的 URL 位置，URL 可以指向一个本地目录（文件）或远程网络共享（HTTP，FTP，等）。

挂载系统光盘到/mnt 目录，创建一个本地 YUM 仓库。

```
[ root@localhost Packages] # cd /etc/yum.repos.d/
[ root@localhost yum.repos.d] # ls
[ root@localhost yum.repos.d] # vim rhel7.repo
[ root@localhost yum.repos.d] #
```

```
[ root@localhost yum.repos.d] # cat rhel7.repo
[base]
name=rhel7
baseurl=file:///mnt
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
[ root@localhost yum.repos.d] #
```

用 yum repolist 等命令查看一个仓库、包和包组。

```
[root@serverX ~]# yum repolist
Loaded plugins: langpacks
repo id          repo name                                status
!rhel_dvd        Remote classroom copy of dvd           4,529
repolist: 4,529
[root@serverX ~]# yum list yum*
Loaded plugins: langpacks
Installed Packages
yum.noarch        3.4.3-118.2.el7      @anaconda/7.0
yum-langpacks.noarch 0.4.2-3.el7      @rhel_dvd
yum-metadata-parser.x86_64 1.1.4-10.el7    @anaconda/7.0
yum-rhn-plugin.noarch 2.0.1-4.el7      @rhel_dvd
yum-utils.noarch  1.1.31-24.el7     @rhel_dvd
Available Packages
yum-plugin-aliases.noarch 1.1.31-24.el7    rhel_dvd
yum-plugin-changelog.noarch 1.1.31-24.el7    rhel_dvd
yum-plugin-tmprepo.noarch  1.1.31-24.el7    rhel_dvd
yum-plugin-verify.noarch  1.1.31-24.el7    rhel_dvd
yum-plugin-versionlock.noarch 1.1.31-24.el7    rhel_dvd
[root@serverX ~]# yum list installed
Loaded plugins: langpacks
Installed Packages
GConf2.x86_64      3.2.6-8.el7      @rhel_dvd
ModemManager.x86_64 1.1.0-6.git20130913.el7  @rhel_dvd
ModemManager-glib.x86_64 1.1.0-6.git20130913.el7  @rhel_dvd
...
[root@serverX ~]# yum grouplist
...
Installed groups:
  Base
  Desktop Debugging and Performance Tools
  Dial-up Networking Support
  Fonts
  Input Methods
...
```

13.3 YUM 管理软件包安装和更新

1、YUM 查找软件包

- » **yum help** will display usage information.
- » **yum list** displays installed and available packages.

```
[root@serverX ~]# yum list 'http*'
Loaded plugins: langpacks
Available Packages
httpcomponents-client.noarch           4.2.5-4.el7          rhel_dvd
httpcomponents-core.noarch             4.2.4-6.el7          rhel_dvd
httpd.x86_64                           2.4.6-17.el7         rhel_dvd
httpd-devel.x86_64                     2.4.6-17.el7         rhel_dvd
httpd-manual.noarch                   2.4.6-17.el7         rhel_dvd
httpd-tools.x86_64                     2.4.6-17.el7         rhel_dvd
```

- » **yum search KEYWORD** lists packages by keywords found in the name and summary fields only.
- To search for packages that have "web server" in their name, summary, and description fields, use **search all**:

```
[root@serverX ~]# yum search all 'web server'
Loaded plugins: langpacks
=====
Matched: web server =====
freeradius.x86_64 : High-performance and highly configurable free RADIUS server
hsqldb.noarch : HypersQL Database Engine
httpd.x86_64 : Apache HTTP Server
libcurl.i686 : A library for getting files from web servers
libcurl.x86_64 : A library for getting files from web servers
mod_revocator.x86_64 : CRL retrieval module for the Apache HTTP server
mod_security.x86_64 : Security module for the Apache HTTP Server
python-paste.noarch : Tools for using a Web Server Gateway Interface stack
```

2、YUM 安装、删除和更新软件包

1) yum 安装软件包

Installing and removing software with yum

- » **yum install PACKAGE_NAME** obtains and installs a software package, including any dependencies.

```
[root@serverX ~]# yum install httpd
Loaded plugins: langpacks
Resolving Dependencies
--> Running transaction check
--> Package httpd.x86_64 0:2.4.6-17.el7 will be installed
--> Processing Dependency: httpd-tools = 2.4.6-17.el7 for package: httpd-2.4.6-17.el7.x86_64
--> Processing Dependency: /etc/mime.types for package: httpd-2.4.6-17.el7.x86_64
--> Processing Dependency: libapr-1.so.0()(64bit) for package: httpd-2.4.6-17.el7.x86_64
--> Processing Dependency: libaprutil-1.so.0()(64bit) for package: httpd-2.4.6-17.el7.x86_64
--> Running transaction check
--> Package apr.x86_64 0:1.4.8-3.el7 will be installed
--> Package apr-util.x86_64 0:1.5.2-6.el7 will be installed
--> Package httpd-tools.x86_64 0:2.4.6-17.el7 will be installed
--> Package mailcap.noarch 0:2.1.41-2.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
 Package           Arch      Version       Repository    size
 =====
 Installing:
 httpd            x86_64   2.4.6-17.el7  rhel_dvd     1.1 M
 Installing for dependencies:
 apr              x86_64   1.4.8-3.el7  rhel_dvd    100 k
 apr-util         x86_64   1.5.2-6.el7  rhel_dvd     90 k
 httpd-tools      x86_64   2.4.6-17.el7  rhel_dvd     76 k
 mailcap          noarch   2.1.41-2.el7  rhel_dvd     31 k

 Transaction Summary
 =====
 Install 1 Package (+4 Dependent packages)

 Total download size: 1.4 M
 Installed size: 4.3 M
 Is this ok [y/d/N]:
```

2) yum 删除软件包

- » **yum remove PACKAGE NAME** removes an installed software package, including any supported packages.

```
[root@serverX ~]# yum remove httpd
```

3) yum 升级软件包

- » **yum update PACKAGE NAME** obtains and installs a newer version of the software package, including any dependencies. Generally the process tries to preserve configuration files in place, but in some cases, they may be renamed if the packager thinks the old one will not work after the update. With no PACKAGE NAME specified, it will install all relevant updates.

```
[root@serverX ~]# yum update
```

Since a new kernel can only be tested by booting to that kernel, the package is specifically designed so that multiple versions may be installed at once. If the new kernel fails to boot, the old kernel is still available. Using **yum update kernel** will actually *install* the new kernel. The configuration files hold a list of packages to "always install" even if the administrator requests an update.

3、YUM 安装、删除软件组

```
[root@localhost 桌面]# yum group list
已加载插件：langpacks, product-id, subscription-manager
This system is not registered to Red Hat Subscription Management
subscription-manager to register.
没有安装组信息文件
Maybe run: yum groups mark convert (see man yum)
Available environment groups:
    最小安装
    基础设施服务器
    文件及打印服务器
    基本网页服务器
    虚拟化主机
    带 GUI 的服务器
可用组：
    传统 UNIX 兼容性
    兼容性程序库
    图形管理工具
    安全性工具
    开发工具
    控制台互联网工具
```

查看组中软件包

```
[root@localhost 桌面]# yum group info 开发工具
已加载插件：langpacks, product-id, subscription-manager
This system is not registered to Red Hat Subscription Mana
bscription-manager to register.
没有安装组信息文件
Maybe run: yum groups mark convert (see man yum)

组：开发工具
组编号：development
描述：基本开发环境。
必要的软件包：
+autoconf
+automake
binutils
+bison
```

例如安装开发工具

```
[root@localhost 桌面]# yum group install 开发工具
已加载插件：langpacks, product-id, subscription-manager
This system is not registered to Red Hat Subscription Mana
bscription-manager to register.
没有安装组信息文件
Maybe run: yum groups mark convert (see man yum)
正在解决依赖关系
--> 正在检查事务
---> 软件包 autoconf.noarch.0.2.69-11.el7 将被 安装
---> 正在处理依赖关系 m4 >=1.4.14，它被软件包 autoconf-
---> 正在处理依赖关系 perl(Data::Dumper)，它被软件包 aut
需要
---> 软件包 automake.noarch.0.1.13.4-3.el7 将被 安装
---> 正在处理依赖关系 perl(TAP::Parser)，它被软件包 autc
需要
--> 正在处理依赖关系 perl(Thread::Queue)，它被软件包 al
```

4、查看 YUM 事务历史

```
[root@localhost 桌面]# tail /var/log/yum.log
Sep 17 23:11:24 Installed: apr-1.4.8-3.el7.x86_64
Sep 17 23:11:24 Installed: apr-util-1.5.2-6.el7.x86_64
Sep 17 23:11:24 Installed: httpd-tools-2.4.6-17.el7.x86_64
Sep 17 23:11:24 Installed: mailcap-2.1.41-2.el7.noarch
Sep 17 23:11:29 Installed: httpd-2.4.6-17.el7.x86_64
[root@localhost 桌面]#
```

查看 YUM 安装与删除的历史事务摘要

```
[root@localhost 桌面]# yum history
已加载插件：langpacks, product-id, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.
ID | 登录用户 | 日期和时间 | 操作 | 变更数
---|---|---|---|---
3 | root <root> | 2014-09-17 23:16 | Install | 1
2 | root <root> | 2014-09-17 23:11 | Install | 5 <
1 | 系统 <空> | 2014-09-12 02:54 | Install | 1191 >
history list
[root@localhost 桌面]#
```

Undo 历史事务

```
[root@localhost 桌面]# yum history undo 3
已加载插件：langpacks, product-id, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.
Undoing transaction 3, from Wed Sep 17 23:16:34 2014
    安装 lftp-4.4.8-3.el7.x86_64 @base
正在解决依赖关系
--> 正在检查事务
```

YUM 命令总结

Task:	Command:
List installed and available packages by name	<code>yum list [NAME-PATTERN]</code>
List installed and available groups	<code>yum grouplist</code>
Search for a package by keyword	<code>yum search KEYWORD</code>
Show details of a package	<code>yum info PACKAGE_NAME</code>
Install a package	<code>yum install PACKAGE_NAME</code>
Install a package group	<code>yum groupinstall "GROUPNAME"</code>
Update all packages	<code>yum update</code>
Remove a package	<code>yum remove PACKAGE_NAME</code>
Display transaction history	<code>yum history</code>

13.4 使用 YUM 仓库

1、查看所有可用 YUM 库, 打开一个已关闭的 YUM 仓库 (关闭用 disable)

```
[root@serverX ~]# yum repolist all
Loaded plugins: langpacks
repo id          repo name           status
rhel-7-public-beta-debug-rpms Red Hat Enterprise Linux 7 Public disabled
rhel-7-public-beta-rpms     Red Hat Enterprise Linux 7 Public enabled: 8,520
rhel-7-public-beta-source-rpms Red Hat Enterprise Linux 7 Public disabled
repolist: 8,520
```

```
[root@serverX ~]# yum-config-manager --enable rhel-7-public-beta-debug-rpms
Loaded plugins: langpacks
=====
repo: rhel-7-public-beta-debug-rpms =====
[rhel-7-public-beta-debug-rpms]
async = True
bandwidth = 0
base_persistdir = /var/lib/yum/repos/x86_64/7Server
baseurl = https://cdn.redhat.com/content/beta/rhel/everything/7/x86_64/debug
cache = 0
cachedir = /var/cache/yum/x86_64/7Server/rhel-7-public-beta-debug-rpms
cost = 1000
deltarpm_percentage =
enabled = 1
...
```

2、使用第三方软件仓库

什么是 EPEL?

EPEL 的全称叫 Extra Packages for Enterprise Linux 。EPEL 是由 Fedora 社区打造，为 RHEL 及衍生发行版如 CentOS、Scientific Linux 等提供高质量软件包的项目。装上了 EPEL 之后，就相当于添加了一个第三方源。

If the URL for a yum repository is known, a configuration file can be created with **yum-config-manager**.

```
[root@serverX ~]# yum-config-manager --add-
repo="http://dl.fedoraproject.org/pub/epel/beta/7/x86_64/"
Loaded plugins: langpacks
adding repo from: http://dl.fedoraproject.org/pub/epel/beta/7/x86_64/

[dl.fedoraproject.org_pub_epel_beta_7_x86_64_]
name=added from: http://dl.fedoraproject.org/pub/epel/beta/7/x86_64/
baseurl=http://dl.fedoraproject.org/pub/epel/beta/7/x86_64/
enabled=1
```

A file was created in the **/etc/yum.repos.d** directory with the output shown. This file can now be modified to provide a customized name and the location of the GPG key. Administrators should download the key to a local file rather than allowing **yum** to retrieve the key from an external source.

```
[EPEL]
name=EPEL 7 Beta
baseurl=http://dl.fedoraproject.org/pub/epel/beta/7/x86_64/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7
```

Installing the Red Hat Enterprise Linux 7 EPEL repo package:

```
[root@serverX ~]# rpm --import http://dl.fedoraproject.org/pub/epel/RPM-GPG-KEY-EPEL-7
[root@serverX ~]# yum install http://dl.fedoraproject.org/pub/epel/beta/7/x86_64/epel-
release-7-0.1.noarch.rpm
```

```
rpm --import http://dl.fedoraproject.org/pub/epel/RPM-GPG-KEY-EPEL-7
http://dl.fedoraproject.org/pub/epel/7/x86_64/epel-release-7-1.noarch.rpm
```

13.5 检查 RPM 包的文件

rpm 是一底层工具，能获取文件包和已安装包的信息。他从本地数据库或包文件本身获取信息。

1、rpm 命令的格式

rpm 命令能够实现几乎所有对 RPM 软件包的管理功能，执行 man rpm 命令可以显示 rpm 命令的命令的帮助信息。

例：执行“man rpm”获取 rpm 命令的手册页信息。

```
RPM(8)
)

NAME
    rpm - RPM Package Manager

SYNOPSIS
    QUERYING AND VERIFYING PACKAGES:
        rpm {-q|--query} [select-options] [query-options]
        rpm {-V|--verify} [select-options] [verify-options]
        rpm --import PUBKEY ...
        rpm {-K|--checksig} [--nosignature] [--nodigest]
                         PACKAGE_FILE ...

    INSTALLING, UPGRADING, AND REMOVING PACKAGES:
        rpm {-i|--install} [install-options] PACKAGE_FILE ...
        rpm {-U|--upgrade} [install-options] PACKAGE_FILE ...
        rpm {-F|--freshen} [install-options] PACKAGE_FILE ...
:
```

以上只是 rpm 命令用法的部分内容，从执行的结果看出，rpm 命令具有相当复杂的命令格式，rpm 命令使用不同的命令选项可以实现以下的 RPM 包管理功能：

- 查询已安装在 Linux 系统中的 RPM 软件包信息
- 查询 RPM 软件包安装文件信息
- 安装 RPM 软件包到当前 Linux 系统

- 从当前 Linux 系统中卸载已安装的 RPM 软件包
- 升级当前 Linux 系统中已安装的 RPM 软件包

下面就来学习使用 rpm 命令实现各种不同的 RPM 软件包管理功能。

2、使用 rpm 命令查询软件包

Linux 系统管理员可以通过 rpm 命令的查询功能收集到当前系统中 RPM 软件包的各种信息，从而作为 RPM 软件包管理的依据，因此查询 RPM 软件包的信息是 Linux 系统管理员很重要的工作之一。

软件包查询时 rpm 命令的基本功能，通过使用不同的查询选项， rpm 命令可以实现对 RPM 包的各种查询功能。 rpm 命令查询功能的基本格式如下：

格式： rpm -q

rpm 命令使用 “-q” 选项实现查询功能，不同的查询需要在 “-q” 选项后附加其他查询选项。

rpm -q: 查询某一个 RPM 包是否已安装
 rpm -qi: 查询某一个 RPM 包的详细信息
 rpm -ql: 列出某 RPM 包中所包含的文件
 rpm -qf: 查询某文件属于哪一个 RPM 包
 rpm -qa: 列出当前系统所有已安装的包
 rpm -qc 列出软件包在系统中安装的配置文件

(1) 查询系统中安装的所有 RPM 包

rpm 命令配合 “-qa” 选项用于查询 Linux 系统中已经安装的所有软件包，命令格式如下：

格式： rpm -qa

例：显示 Linux 系统中所有已经安装的 RPM 软件包的名称。（下例中只显示前 5 个）

```
[root@zx Packages]# rpm -qa | head -5
libXi-devel-1.3-3.el6.i686
libv4l-0.6.3-2.el6.i686
subversion-javahl-1.6.11-2.el6.i686
mtools-4.0.12-1.el6.i686
gnome-desktop-2.28.2-8.el6.i686
[root@zx Packages]#
```

“rpm - qa” 命令查询并显示系统中已安装的所有软件包的列表，因此显示的查询结果较长，命令的执行时间也较长，可以配合 “more” 命令一起使用，实现查询结果的分屏显示。

例：分屏显示 Linux 系统中所有已经安装的 RPM 软件包的名称。

```
[root@zx ~]# rpm -qa | more
```

“rpm - qa” 命令可以配合 “grep” 命令一起使用，用于查询 RPM 包名称中包含指定关键字符串的软件包。

例：查询系统中所有名称中包含“httpd”字符串的软件包，可以使用如下命令：

```
[root@zx ~]# rpm -qa | grep httpd
httpd-2.2.15-5.el6.i686
httpd-tools-2.2.15-5.el6.i686
[root@zx ~]#
```

和“wc”命令搭配使用，可查询系统中一共安装了多少个软件包：

例：统计当前系统中安装的 rpm 包的个数。

```
[root@zx ~]# rpm -qa | wc -l
1513
[root@zx ~]#
```

(2) 查询软件包是否安装

rpm 命令搭配“-q”选项一起使用，用于查询 Linux 系统中指定名称的软件包是否安装，命令格式如下：

格式： rpm -q RPM 包名称

“rpm -q”命令需要指定待查询的软件包名称作为命令参数，如果系统中已经安装了该软件包，命令执行结果会显示完整的软件包名称（软件包名称+软件包的版本号），否则将提示软件包没有被安装的信息。

例：查看系统中是否已经安装 bash 和 bsh 软件包。

```
[root@zx ~]# rpm -q bash
bash-4.1.2-3.el6.i686
[root@zx ~]#
[root@zx ~]# rpm -q bsh
package bsh is not installed
[root@zx ~]#
```

“rpm -q”命令中指定的软件包名称需要准确的拼写，该命令不会在软件报名中进行局部匹配的查询。

例：查看系统中已经安装与 http 相关的程序包。

```
[root@zx ~]# rpm -qa | grep http
httpd-2.2.15-5.el6.i686
httpd-tools-2.2.15-5.el6.i686
jakarta-commons-httpclient-3.1-0.6.el6.i686
[root@zx ~]#
```

(3) 查询软件包详细信息

在使用“rpm -q”命令确定了某个软件包在当前系统中已经安装后，Linux 系统管理员需要进一步了解软件包的较详细的信息，“rpm -qi”命令可以实现该功能。

rpm 命令配合“-qi”选项用于查询 Linux 系统中指定名称软件包的详细信息，命令格式如下：

格式: rpm -qi RPM 包名称

例: 查看软件包 bash 的详细信息。

```
[root@zx ~]# rpm -qi bash
Name        : bash
Version     : 4.1.2
Release     : 3.el6
23时 52分 28秒
Install Date: 2012年 01月 04日 星期三 06时 33分 11秒      Build Host: ls20-bc2-14
.build.redhat.com
Group       : System Environment/Shells    Source RPM: bash-4.1.2-3.el6.src.rpm
Size        : 3107578                      License: GPLv3+
Signature   : RSA/8, 2010年 08月 16日 星期一 23时 30分 36秒, Key ID 199e2f91fd431d51
Packager    : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
URL         : http://www.gnu.org/software/bash
Summary     : The GNU Bourne Again shell
```

“rpm -qi”命令的执行结果中包括如下几项 RPM 包的详细信息:

- 软件包中软件的名称 (Name)
- 软件的版本信息 (version 和 Release)
- 软件包的建立时间 (Build Date)
- 软件包的安装时间 (Install Date)
- 软件包的大小 (Size)
- 软件遵从的许可协议 (License)
- 软件的打包者 (Packager)
- 软件包的概括描述 (Summary) 和详细描述 (Description) 信息

通过阅读 “rpm -qi” 命令的执行结果, 可以对指定软件包有一个比较详细的了解。

(4) 查询已安装软件包中的文件列表

在使用 “rpm -qi” 命令查看了某软件包的详细信息后, Linux 系统管理员需要了解软件包中包括了哪些文件, 即安装该软件包时在当前的 Linux 系统中安装了哪些文件, “rpm -ql” 命令可以实现此查询功能。

rpm 命令配合 “-ql” 选项用于查询 Linux 系统中指定名称的软件包中所包括的文件列表, 命令格式如下:

格式: rpm -ql RPM 包名称

例: 查看 bash 软件包安装的文件列表。

```
[root@zx ~]# rpm -ql bash
/bin/bash
/bin/sh
/etc/skel/.bash_logout
/etc/skel/.bash_profile
/etc/skel/.bashrc
/usr/bin/bashbug-32
/usr/share/doc/bash-4.1.2/COPYING
```

“rpm -ql” 命令中被查询的软件包可能包括相当多的文件, 因此可以使用 grep 命令

过滤查询结果，只显示文件或路径名中包括指定关键字的文件列表。

例：查看 httpd 软件包安装的文件列表，过滤出文件名中包 “conf” 的文件。

```
[root@zx ~]# rpm -ql httpd | grep conf
/etc/httpd/conf
/etc/httpd/conf.d
/etc/httpd/conf.d/README
/etc/httpd/conf.d/welcome.conf
/etc/httpd/conf/httpd.conf
/etc/httpd/conf/magic
/etc/sysconfig/httpd
/usr/lib/httpd/modules/mod_log_config.so
[root@zx ~]#
```

(5) 查询系统中文件所属的软件包

Linux 系统管理员在执行某些系统管理任务时可能需要查询系统中的某个文件属于哪个软件包，即 Linux 系统是通过哪个软件包安装了指定的文件，“rpm -qf” 可以实现此查询功能。

rpm 命令配合 “-qf” 选项用于查询 Linux 系统中指定文件所属的软件包。

格式： rpm -qf 文件路径和名称

“rpm -qf” 命令需要指定待查询的文件名称作为命令参数，文件名中需要包括文件的路径名（绝对路径或相对路径名），命令执行结果将显示包含该文件的软件包名称。

例：查看系统中 vim 程序由哪个软件包安装。

```
[root@zx ~]# rpm -qf /usr/bin/vim
vim-enhanced-7.2.411-1.4.el6.i686
[root@zx ~]#
```

在 Linux 系统中并不是所有文件都是通过 RPM 软件包的方式安装的，系统中得很多文件是通过 Linux 系统安装过程和系统运行时生成的，即这些文件不会属于任何 RPM 软件包。

```
[root@zx ~]# rpm -qf /var/log/dmesg
file /var/log/dmesg is not owned by any package
[root@zx ~]#
```

(6) 查询 RPM 安装包文件中的信息

Linux 系统管理员在获得了 RPM 安装包文件进行安装之前，需要了解 RPM 安装包文件的相信信息，以及 RPM 安装包会安装到系统中的文件，“rpm -qi” 命令可以实现此查询功能。

rpm 命令配合 “-qi” 和 “-qpl” 选项用于查询指定 RPM 安装包文件的信息，命令格式：

格式： rpm -qi RPM 包文件名

rpm -qpl RPM 包文件名

例：显示 vim-enhanced-7.2.411-1.4.el6.i686.rpm 软件包文件的详细信息。

```
[root@zx ~]# cd /media/cdrom/Server/Packages/
[root@zx Packages]# rpm -qpi vim-enhanced-7.2.411-1.4.el6.i686.rpm
```

“rpm - qpi”命令所显示的软件包详细信息的格式与“rpm - qi”命令的执行结果类似，不同点在于“rpm - qpi”命令显示的是 RPM 安装包文件的详细信息，而“rpm - qi”命令显示的是当前 Linux 系统中已安装的软件包的详细信息。

“rpm - qpl”命令使用 RPM 安装包的文件名作为命令参数，显示该 RPM 软件包中包含的文件列表，即如果安装该软件包将在当前 Linux 系统中安装的文件列表。

例：显示 vim-enhanced-7.2.411-1.4.el6.i686.rpm 软件包如查在系统中安装，将安装的文件列表。

```
[root@zx Packages]# rpm -qpl vim-enhanced-7.2.411-1.4.el6.i686.rpm
warning: vim-enhanced-7.2.411-1.4.el6.i686.rpm: Header V3 RSA/SHA256 Signature
e, key ID fd431d51: NOKEY
/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
```

“rpm - qpl”命令所显示的软件包文件列表信息的格式与“rpm - qi”命令的执行结果类似，不同点在于“rpm - qpi”命令显示的是 RPM 安装包文件的文件列表信息，而“rpm - qi”命令显示的是当前 Linux 系统中已安装的软件包的文件列表信息。

3、使用 rpm 命令安装软件包

安装软件包是 Linux 系统管理员需要完成的另一类重要任务，可以使用 rpm 命令完成软件包的安装工作。

(1) RPM 软件包的基本安装

rpm 命令配合“-i”选项用于安装指定的 RPM 软件包到当前 Linux 系统，命令格式如下：

格式： rpm - i RPM 安装包文件名

“rpm - i”命令使用待安装的 RPM 安装包文件名作为参数，安装该软件包的文件到当前系统。

“rpm - i”命令如果成功安装指定的软件包将不提示任何信息，软件包的安装过程需要一定的安装时间。在软件包安装成功后可以使用 rpm 的查询命令验证软件包在当前系统中的存在。

(2) 在安装软件包的同时显示详细信息

“rpm - i”命令虽然可以安装 RPM 软件包，但是在安装过程中没有任何屏幕显示信息，因此过程不是很直观，rpm 命令配合“-ivh”选项使用时，可以在安装过程中显示更多的信息。

格式： rpm -ivh RPM 安装包文件

“rpm - ivh”命令使用待安装的 RPM 软件包的文件名称作为命令参数，在安装软件包

的过程中会以百分比的形式显示安装的进度或一些其他信息。

“`rpm - ivh`”命令与“`rpm - i`”命令实现同样的软件包安装功能，但是提供了更加友好的安装过程界面，使用户可以了解到安装过程中更多的信息。

```
[root@localhost ~]# mount /dev/cdrom /mnt
mount: /dev/sr0 写保护，将以只读方式挂载
[root@localhost ~]#
[root@localhost ~]# cd /mnt/Packages/
[root@localhost Packages]# rpm -ivh lftp-4.4.8-3.el7.x86_64.rpm
准备中...
正在升级/安装...
1:lftp-4.4.8-3.el7                                ##### [100%]
[root@localhost Packages]# ^C
[root@localhost Packages]#
```

4、使用 `rpm` 命令卸载软件包

(1) RPM 软件包的卸载

`rpm` 命令与“`-e`”选项配合使用可以实现 RPM 软件包的卸载。

命令格式：`rpm - e 软件包名称`

```
[root@localhost Packages]#
[root@localhost Packages]# rpm -e lftp
[root@localhost Packages]#
```

13.6 使用 yum 安装本地包文件

```
[root@serverX ~]# yum localinstall wonderwidgets-1.0-4.x86_64.rpm
[root@serverX ~]# rpm -q wonderwidgets
wonderwidgets-1.0-4.x86_64
```



Note

`rpm -ivh PACKAGEFILE.rpm` can also be used to install package files. However, using `yum` helps maintain a transaction history kept by `yum` (see `yum history`).

13.7 从 RPM 包解压文件

Select files can also be extracted by specifying the path of the file:

```
[root@serverX ~]# rpm2cpio wonderwidgets-1.0-4.x86_64.rpm | cpio -id "*txt"
11 blocks
[root@serverX ~]# ls -l usr/share/doc/wonderwidgets-1.0/
total 4
-rw-r--r--. 1 root root 76 Feb 13 19:27 README.txt
```

13.8 源码包安装

Installation

Linux: [Most distributions](#) include and use NTFS-3G by default. Please use that one unless it's an [old version](#). If you wish to install NTFS-3G from the source code then make sure you have installed the basic development tools (gcc compiler, libc-dev libraries). Then type:

```
./configure
make
make install # or 'sudo make install' if you aren't root
```

Non-Linux: Please see the OS specific installation and source packages above.

Usage

If there was no error during installation then the NTFS volume can be mounted in read-write mode for everybody as follows. Unmount the volume if it had already been mounted, replace /dev/sda1 and /mnt/windows, if needed.

```
mount -t ntfs-3g /dev/sda1 /mnt/windows
```

Please see the [NTFS-3G Manual](#) for more options and examples.

You can also make NTFS to be mounted during boot by adding the following line to the **end of the /etc/fstab file**:

```
/dev/sda1 /mnt/windows ntfs-3g defaults 0 0
```

例：在进行 axel 应用程序的编译安装之前需要完成几项准备工作。

1、确认系统中已经安装了编译环境

在对任何应用程序的源代码进行编译安装之前都需要确认当前系统中已安装了 gcc 编译环境。

```
[root@zx Packages]# rpm -qa | grep gcc
gcc-gfortran-4.4.4-13.el6.i686
gcc-4.4.4-13.el6.i686
libgcc-4.4.4-13.el6.i686
gcc-c++-4.4.4-13.el6.i686
[root@zx Packages]# █
```

如果当前 Linux 系统中没有 gcc 编译器环境，那么应该使用 rpm 命令进行安装；由于安装时软件包之间有依赖关系，使用 rpm 命令进行安装非常繁琐，建议使用 yum 工具进行安装。

2、下载 Zebra 源代码包

<http://wilmer.gaast.net/downloads/>

3、解开源代码包

```
[root@zx sott]# tar zxvf axel-1.0a.tar.gz -C /usr/src

[root@zx src]# cd /usr/src
[root@zx src]# ll
总用量 16
drwxr-xr-x. 3 lipi users 4096 2月 20 2002 axel-1.0a
drwxr-xr-x. 2 root root 4096 12月 4 2009 debug
drwxr-xr-x. 3 root root 4096 1月 4 06:37 kernels
drwxrwxrwx. 13 lipi lipi 4096 1月 17 21:24 zebra-0.95a
[root@zx src]#
```

tar 命令将“axel-1.0a.tar.gz”源代码压缩包解压后，再将所有的 axel 的源代码文件释放到默认在当前目录下建立的目录“axel-1.0a”中。

```
[root@zx src]# ll axel-1.0a/
总用量 204
-rw-r--r--. 1 lipi users 9231 12月 7 2001 API
-rw-r--r--. 1 lipi users 4403 2月 16 2002 axel.1
-rw-r--r--. 1 lipi users 16357 2月 16 2002 axel.c
-rw-r--r--. 1 lipi users 2874 2月 17 2002 axel.h
-rw-r--r--. 1 lipi users 3773 2月 20 2002 axelrc.example
-rw-r--r--. 1 lipi users 9234 2月 20 2002 CHANGES
-rw-r--r--. 1 lipi users 5421 2月 20 2002 conf.c
-rw-r--r--. 1 lipi users 1661 2月 16 2002 conf.h
-rwxr-xr-x. 1 lipi users 5941 1月 23 2002 configure
-rw-r--r--. 1 lipi users 8405 1月 25 2002 conn.c
```

4、进入源代码目录

在对源代码进行配置和编译之前还需要完成最后一件准备工作，进入源代码文件所在的子目录“zebra-0.95a”。

```
[root@zx src]# cd axel-1.0a/
[root@zx axel-1.0a]# pwd
/usr/src/axel-1.0a
[root@zx axel-1.0a]#
```

5、编译前的配置

所有的源代码软件在进行编译前都需要执行 configure 命令完成程序编译前的配置工作。configure 命令需要进入源代码目录后执行，该命令可以使用“--help”选项获取帮助。

使用“--prefix”选项可以指定应用程序编译完成后的安装路径，命令格式如下：

命令格式：./configure --prefix=程序安装目录的绝对路径

```
[root@zx axel-1.0a]# ./configure --prefix=/usr/local/axel
The strip option is enabled. This should not be a problem usually, but on some
systems it breaks stuff.

Configuration done:
  Internationalization disabled.
  Debugging disabled.
  Binary stripping enabled.
[root@zx axel-1.0a]#
```

configure 命令的配置需要一定的时间，配置过程会在屏幕上显示大量的输出信息，这些信息有利于管理员了解程序配置的过程。

6、编译与安装

1) 程序编译过程

在使用 configure 命令对的源代码进行配置后，需要使用 make 命令进行程序的二进制编译。

命令格式: make

```
[root@zx axel-1.0a]# make
gcc -c axel.c -o axel.o -Wall -O3
gcc -c conf.c -o conf.o -Wall -O3
gcc -c conn.c -o conn.o -Wall -O3
gcc -c ftp.c -o ftp.o -Wall -O3
gcc -c http.c -o http.o -Wall -O3
gcc -c search.c -o search.o -Wall -O3
gcc -c tcp.c -o tcp.o -Wall -O3
tcp.c: 在函数 'get_if_ip' 中:
tcp.c:98: 警告 : dereferencing pointer 'x' does break strict-aliasing rules
tcp.c:97: 附注 : initialized from here
gcc -c text.c -o text.o -Wall -O3
gcc axel.o conf.o conn.o ftp.o http.o search.o tcp.o text.o -o axel -lpthread
strip axel
[root@zx axel-1.0a]#
```

执行 make 命令是进行源代码编译的过程，make 命令需要比 configure 命令更长的时间，并且同样会有大量的屏幕信息输出。

2) 程序安装过程

在使用 make 命令对程序源代码进行编译后就可以安装已编译完成的程序到预先配置的目录了。编译之后的程序安装命令如下：

命令格式: make install

```
[root@zx axel-1.0a]# make install
mkdir -p /usr/local/axel/bin/
cp axel /usr/local/axel/bin/axel
mkdir -p /usr/local/axel/etc/
cp axelrc.example /usr/local/axel/etc/axelrc
mkdir -p /usr/local/axel/share/man/man1/
cp axel.1 /usr/local/axel/share/man/man1/axel.1
[root@zx axel-1.0a]#
```

“make install” 命令将按照“configure”命令的“--prefix”选项中设定的安装路径将已编译完成的应用程序安装到目标目录；在进行程序安装之前应确保程序安装的目标目录存在。

“make install”命令执行的时间与配置和编译过程相比不是很长，屏幕会有输出信息显示。

3) 验证编译安装的程序

当使用“make install”命令完成应用程序的编译安装后需要对安装的应用程序进行验证。

```
[root@zx axel-1.0a]# ls /usr/local/axel
bin etc share
[root@zx axel-1.0a]# /usr/local/axel/bin/axel --help
Usage: axel [options] url1 [url2] [url...]

--max-speed=x           -s x      Specify maximum speed (bytes per second)
--num-connections=x    -n x      Specify maximum number of connections
--output=f              -o f      Specify local output file
--search[=x]            -S [x]   Search for mirrors and download from x servers
--no-proxy              -N       Just don't use any proxy server
--quiet                 -q       Leave stdout alone
--verbose               -v       More status information
--alternate             -a       Alternate progress indicator
--help                  -h       This information
--version               -V       Version information

Report bugs to lintux@lintux.cx
[root@zx axel-1.0a]# █
```

使用 Axel 工具从网上下载软件。

```
[root@zx axel-1.0a]# ln -s /usr/local/axel/bin/axel /usr/bin //建立符号链接
[root@zx axel-1.0a]# axel -n 10 -o /root ftp://ftp.zebra.org/pub/zebra/zebra-0.95a.tar.gz
连接数 下载目录
Initializing download: ftp://ftp.zebra.org/pub/zebra/zebra-0.95a.tar.gz
File size: 1370152 bytes
Opening output file /root/zebra-0.95a.tar.gz
Starting download

[  0%] ..... [  6.1KB/s]
[  3%] ..... [  9.6KB/s]
[  7%] ..... Connection 0 finished
..... █
```

第十四章 访问 Linux 文件系统

14.1 识别文件系统和设备

1、存储管理概念

A long listing of the `/dev/vda` device file on serverX reveals its special file type as **b**, which stands for block device:

```
[student@serverX ~]$ ls -l /dev/vda
brw-rw----. 1 root disk 253, 0 Mar 13 08:00 /dev/vda
```



Note

Exceptions are hard drives in virtual machines, which typically show up as `/dev/vd<letter>` or `/dev/xvd<letter>`.

1、检查文件系统

`df` 命令 查看已挂载的存储设备信息

```
[student@serverX ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda1       6.0G  3.9G  2.2G  65% /
devtmpfs        929M    0  929M   0% /dev
tmpfs          937M   80K  937M   1% /dev/shm
tmpfs          937M   2.2M  935M   1% /run
tmpfs          937M    0  937M   0% /sys/fs/cgroup
```

`du` 命令 查看目录与目录中文件的容量

```
[root@serverX ~]# du -h /var/log
...
4.9M  /var/log/sa
68K   /var/log/prelink
0     /var/log/qemu-ga
14M   /var/log
```

14.2 挂载和卸载文件系统

1、手动挂载文件系统

Mount by device file of the partition that holds the file system.

```
[root@serverX ~]# mount /dev/vdb1 /mnt/mydata
```

Mount the file system by universal unique id, or the UUID, of the file system.

```
[root@serverX ~]# mount UUID="46f543fd-78c9-4526-a857-244811be2d88" /mnt/mydata
```

The `blkid` command gives an overview of existing partitions with a file system on them and the UUID of the file system, as well as the file system used to format the partition.

```
[root@serverX ~]# blkid
/dev/vda1: UUID="46f543fd-78c9-4526-a857-244811be2d88" TYPE="xfs"
```

2、手动卸载文件系统

To unmount a file system, the **umount** command expects the mount point as an argument.

Change to the **/mnt/mydata** directory. Try to umount the device mounted on the **/mnt/mydata** mount point. It will fail.

```
[root@serverX ~]# cd /mnt/mydata
[root@serverX mydata]# umount /mnt/mydata
umount: /mnt/mydata: target is busy.
        (In some cases useful info about processes that use
         the device is found by lsof(8) or fuser(1))
```

Unmounting is not possible if the mount point is accessed by a process. For **umount** to be successful, the process needs to stop accessing the mount point.

The **lsof** command lists all open files and the process accessing them in the provided directory. It is useful to identify which processes currently prevent the file system from successful unmounting.

```
[root@serverX mydata]# lsof /mnt/mydata
COMMAND  PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
bash    1593  root  cwd    DIR  253,2          6  128 /mnt/mydata
lsof    2532  root  cwd    DIR  253,2         19  128 /mnt/mydata
lsof    2533  root  cwd    DIR  253,2         19  128 /mnt/mydata
```

Once the processes are identified, an action can be taken, such as waiting for the process to complete or sending a SIGTERM or SIGKILL signal to the process. In this case, it is sufficient to change the current working directory to a directory outside the mount point.

```
[root@serverX mydata]# cd
[root@serverX ~]# umount /mnt/mydata
```

注意：如果你当前工作目录在挂载点目录中，卸载会提示目标忙，不能卸载。

3、U 盘等移动存储设备

在图型界面自动挂载到/run/media/<user>/<lable> 下。

14.3 创建文件间链接

1、建立硬链接

一个硬链接是一个新的与一个参考的文件系统上的现有文件的目录条目。在文件系统中的每个文件都有一个硬链接的默认。为了节省空间，而不是复制，一种新的硬链接可以引用相同的文件。一个新的硬链接都需要有一个不同的文件名，如果是在同一个目录下创建现有的硬链接，或需要驻留在一个不同的目录。所有硬链接指向同一个文件具有相同的权限，连接数，用户/组的所有制，时间戳，文件内容。指向同一个文件的内容硬链接需要在相同的文件系统。

The **ls -l** shows the hard link count after the permissions and before the owner of a file.

```
[root@serverX ~]# echo "Hello World" > newfile.txt
[root@serverX ~]# ls -l newfile.txt
-rw-r--r--. 1 root root 0 Mar 11 19:19 newfile.txt
```

Create a hard link **newfile-link2.txt** for the existing file **newfile.txt** in the same directory.

```
[root@serverX ~]# ln newfile.txt /tmp/newfile-hlink2.txt
[root@serverX ~]# ls -l newfile.txt /tmp/newfile-hlink2.txt
-rw-rw-r--. 2 root root 12 Mar 11 19:19 newfile.txt
-rw-rw-r--. 2 root root 12 Mar 11 19:19 newfile-hlink2.txt
```

Even if the original file gets deleted, the content of the file is still available as long as at least one hard link exists.

```
[root@serverX ~]# rm -f newfile.txt
[root@serverX ~]# ls -l /tmp/newfile-link2.txt
-rw-rw-r--. 1 root root 12 Mar 11 19:19 /tmp/newfile-link2.txt
[root@serverX ~]# cat /tmp/newfile-link2.txt
Hello World
```

2、建立软链接

使用 **ln -s** 命令创建一个软链接，也称为一个符号链接。一个软链接是一种特殊类型的文件，点到一个现有的文件或目录。软链接可以指向另一个文件系统的文件或目录。不像硬链接，符号链接可以指向一个不同的文件系统的文件。

```
[root@serverX ~]# ln -s /root/newfile-link2.txt /tmp/newfile-symlink.txt
[root@serverX ~]# ls -l newfile-link2.txt /tmp/newfile-symlink.txt
lrwxrwxrwx. 1 root root 11 Mar 11 20:59 /tmp/newfile-symlink.txt -> /root/newfile-link2.txt
-rw-rw-r--. 1 root root 12 Mar 11 19:19 newfile-link2.txt
```

Create a soft link **/root/configfiles** pointing to the **/etc** directory.

```
[root@serverX ~]# ln -s /etc /root/configfiles
[root@serverX ~]# cd /root/configfiles
[root@serverX configfiles]# pwd
/root/configfiles
```

```
[root@serverX ~]# rm -f newfile-link2.txt
[root@serverX ~]# ls -l /tmp/newfile-symlink.txt
lrwxrwxrwx. 1 root root 11 Mar 11 20:59 /tmp/newfile-symlink.txt -> newfile-link2.txt
[root@serverX ~]# cat /tmp/newfile-symlink.txt
cat: /tmp/newfile-symlink.txt: No such file or directory
```

14.3 定位系统上的文件

系统管理员需要搜索的文件系统的某些标准匹配的文件的工具。本节讨论的两个命令，可以在文件系统中的文件搜索。**locate** 命令搜索预生成的数据库的文件名或文件路径，瞬间返回结果。**find** 命令搜索文件系统中的实时抓取通过文件系统。

1、locate

先用 **updatedb** 更新数据库

Search for files with "passwd" in the name or path in directory trees readable by user student on serverX.

```
[student@serverX ~]$ locate passwd
/etc/passwd
/etc/passwd-
/etc/pam.d/passwd
/etc/security/opasswd
/usr/bin/gpasswd
/usr/bin/grub2-mkpasswd-pbkdf2
/usr/bin/lppasswd
/usr/bin/passwd
/usr/bin/userpasswd
/usr/bin/vino-passwd
/usr/bin/vncpasswd
```

Results are returned even when the file name or path is only a partial match to the search query.

```
[root@serverX ~]# locate image
/home/myuser/boot.image
/home/someuser/my_family_image.png
/home/student/myimages-vacation/picture.png
```

Locate -i 文件名 // 选项可以查找时忽略大小写。

Locate -n 5 文件名 // 选项限制搜索结果数量不超过 5 个。

2、find

find 命令是 Linux 系统中功能非常强大的文件和目录查找命令，可以根据目标的名称、类型、大小等不同属性进行查找。

语法如下：

```
find [路径] [条件表达式]
```

路径指系统从这里开始沿着目录树向下查找文件。有多个路径相互用空格分离，如果不写路径，那么默认为当前目录。

条件表达式是查找匹配条件，它的参数非常多，这里只介绍一些常用的参数。

-name: 根据文件名查找，支持统配符*和?。

-iname 文件名不去分大小写

-uid, -gid 用户 id, 组 ID

-perm 权限（以数字表示，/排除，+大于，- 小于）

-atime n: 搜索在过去 n 天读取过的文件。

-ctime n: 搜索在过去 n 天修改过的文件。

-link 文件数量 (+1)

-group groupname: 搜索所有组为 groupname 的文件。

-user 用户名: 搜索所有文件属主为用户名 (ID 或名称) 的文件。

-size n: 根据文件大小查找，一般使用 “+”、“-” 设置大小或小于，单位为 k(小写)、M、G。

-print: 输出搜索结果，并且打印。

-type: 根据文件类型进行查找。

b 块设备文件、c 字符设备文件、d 目录文件、f 普通文件

应用举例：

(1) 根据文件名查找

例 1：查找一个文件名是 grub.conf 的文件，可以使用如下命令：

```
find / -name grub.conf
find 命令后的“/”表示搜索整个硬盘。
```

(2) 快速查找文件

例 2：查找在/etc 目录下一个文件名是 grub.conf 的文件，可以使用如下命令：

```
find /etc -name grub.conf
```

(3) 根据部分文件名查找方法

例 3：查找在/etc 目录文件名以“g”开头的、以“.conf”结尾的文件。

```
find /etc -name 'g*.conf'
```

其中*是通配符表示任意字符，? 表示一个字符。

(4) 根据文件类型查找

例 4：查找/boot 目录下所有目录。

```
find /boot -type d
```

(5) 根据文件所属的用户查找

例 5：查找/var/log 目录中属于用户 liweijie 的文件或目录。

```
Find /boot -user liweijie
```

(4) 使用混合查找方式查找文件

-a 表法 and

-o 表示 or

例 6：在/boot 目录中查找超过 1024KB 而且文件名以 vmlinuz 开头的文件。

```
Find /var/log -size +1024k -a -name "vmlinuz*"
```

例 6：在/boot 目录中查找超过 1024KB 或者文件名以 vmlinuz 开头的文件。

```
Find /boot -size +1024k -o -name "vmlinuz*"
```

(5) 对查找到的结果进行处理（加 exec 关键字）

```
-exec command {} \;
```

-- 将查到的文件执行 command 操作, {} 和 \; 之间有空格

例 7：从根目录开始查 tmpfile，一旦查到马上删除

```
find / -name "tmpfile" -exec rm {} \;
```

例 8：在/etc 目录中查找 grub.conf 文件，并把文件复制到/tmp 目录中

```
Find /etc -name grub.conf -exec cp '{}' '/temp' ';'
```

第十五章 使用虚拟化系统

15.1 管理本地虚拟主机

1、KVM

KVM (Kernel-based Virtualization Machine 基于内核的虚拟机) 是一个完整的虚拟化解决方案，建立在标准的红帽企业 Linux 内核之上。它可以运行多个 Windows 和 Linux 客户操作系统。在红帽企业 Linux 的 KVM 虚拟机管理程序与 libvirt API 和公用事务的管理程序，如 virt-manager 和 virsh。由于红帽企业 Linux 是红帽企业虚拟化的基础和 Red Hat OpenStack 平台，KVM 是红帽云基础设施产品一致的组件。

kvm 是一个开源的系统虚拟化模块，自 Linux 2.6.20 之后集成在 Linux 的各个主要发行版本中。它使用 Linux 自身的调度器进行管理，所以相对于 Xen，其核心源码很少。KVM 目前已成为学术界的主流 VMM 之一。KVM 的虚拟化需要硬件支持 (如 Intel VT 技术或者 AMD V 技术)。是基于硬件的完全虚拟化。而 Xen 早期则是基于软件模拟的 Para-Virtualization，新版本则是基于硬件支持的完全虚拟化。但 Xen 本身有自己到进程调度器，存储管理模块等，所以代码较为庞大。广为流传的商业系统虚拟化软件 VMware ESX 系列也是基于软件模拟的 Para-Virtualization。

2、验证 CPU 虚拟化支持

The KVM hypervisor requires either an Intel processor with the Intel VT-x and Intel 64 extensions for x86-based systems, or an AMD processor with the AMD-V and the AMD64 extensions. To verify that the host system hardware supports the correct extensions, view **/proc/cpuinfo**.

```
[root@serverX ~]# grep --color -E "vmx|svm" /proc/cpuinfo
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx lm constant_tsc
arch_perfmon pebs bts rep_good aperfmpf perf_pni dtes64 monitor ds_cpl vmx smx est
tm2 ssse3 cx16 xtpr pdcm sse4_1 xsave lahf_lm dts tpr_shadow vnmi flexpriority
```

The No eXecute (NX) feature, called eXecute Disable (XD) by Intel and Enhanced Virus Protection by AMD, is not necessary for building a host on Red Hat Enterprise Linux, but is required for a Red Hat Enterprise Virtualization hypervisor (RHEV-H).

```
[root@serverX ~]# grep --color -E "nx" /proc/cpuinfo
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx lm constant_tsc
arch_perfmon pebs bts rep_good aperfmpf perf_pni dtes64 monitor ds_cpl vmx smx est
tm2 ssse3 cx16 xtpr pdcm sse4_1 xsave lahf_lm dts tpr_shadow vnmi flexpriority
```

vmx: 如果输出的结果包含 vmx，它是 Intel 处理器虚拟机技术标志；

svm: 如果包含 svm，它是 AMD 处理器虚拟机技术标志。

如果你甚么都得不到，那应你的系统并没有支持虚拟化的处理，不能使用 kvm。

lm: 长模式。表示支持 64 位。Linux 发行版本必须在 64bit 环境中才能使用 KVM
确保 BIOS 里开启 VT

Intel(R) Virtualization Tech [Enabled]

如果有必要，还需要在 BIOS 中开启 VT-d，并关机冷启动计算机。

15.2 安装一个新的虚拟主机

4、安装虚拟化需要软件包

Building a RHEL virtualization host requires the **qemu-kvm** and **qemu-img** packages at minimum, to provide the user-level KVM emulator and disk image manager.

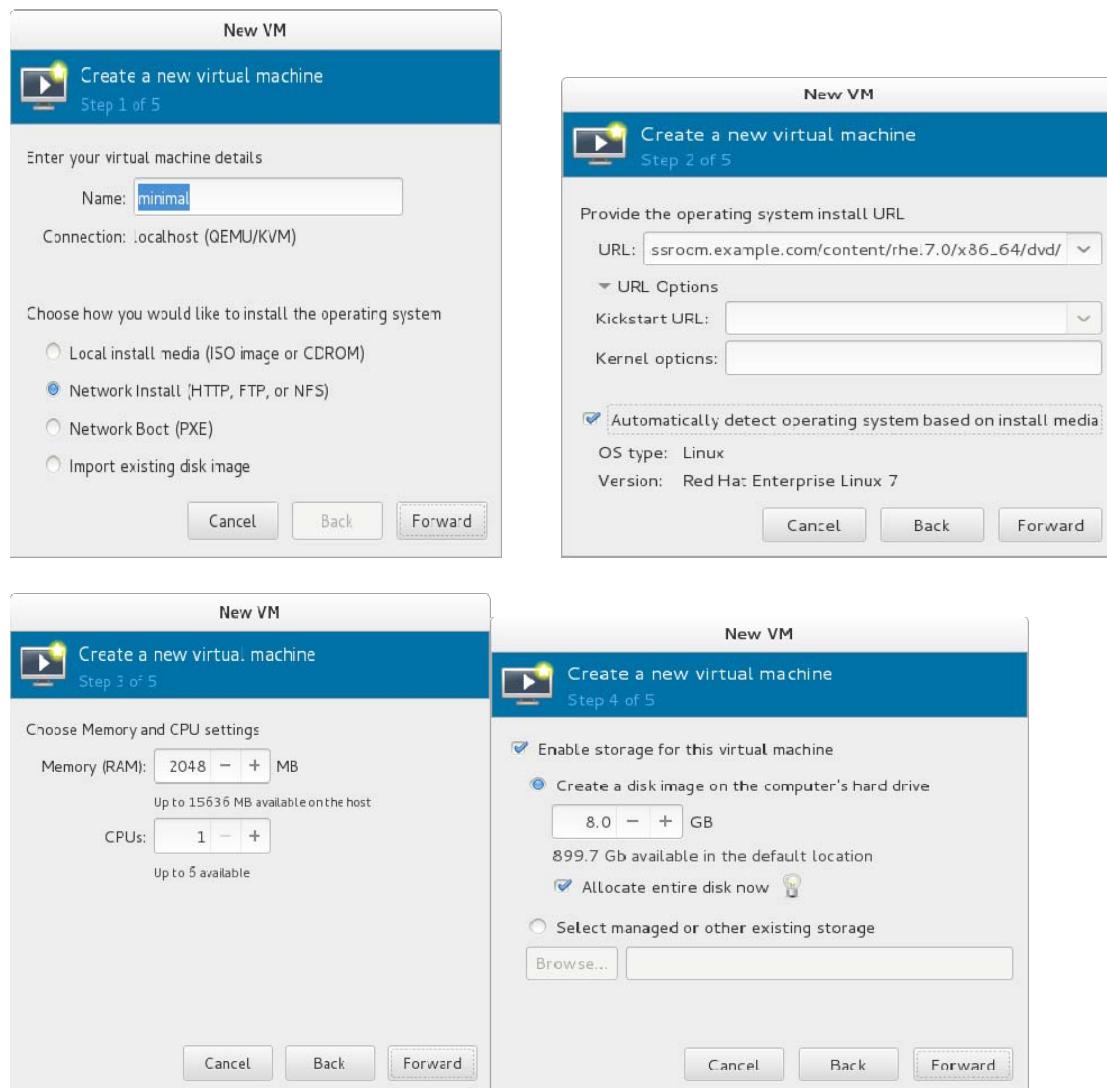
```
[root@serverX ~]# yum install qemu-kvm qemu-img
```

```
[root@serverX ~]# yum install virt-manager libvirt libvirt-python python-virtinst  
libvirt-client
```

5、安装一个新的虚拟化主机

从菜单的应用→系统工具→虚拟机管理器启动虚拟机管理器，或以 root 身份运行 **virt-manager** 命令。

单击创建一个新的虚拟机的按钮来打开新的虚拟机向导。安装前请确保安装介质或媒体可用



李 伟 阶
QQ: 17533203
整理于 2014 年 9 月