跟阿铭学Linux

第二版

欢迎加QQ群讨论

群1(163262181)

群2(148412746)

CHAPTER

ONE

第一章 前言

学习Linux请加QQ群： 群1(163262181) 群2(148412746)

时隔2年多，跟阿铭学Linux(第二版)已面世，相对第一版，这本书更全面，更系统。本教程由浅入深，内容精湛、通俗易懂！案例丰富、容易操作，内容涉足基础的系统操作也涉足工作中常用的各种服务软件的应用、部署，优化，实践性非常强，即使是0基础学员，只要您能够坚持把所有章节都学完，相信您一定会受益匪浅，对于有工作经验的，你也可以把它当做工具书，它能解决你工作当中的一些需求和难题，另外为了帮助更多的初学者，阿铭创建了一个论坛 (http://www.lishiming.net)，欢迎大家交流讨论，论坛也开通了VIP服务，那是阿铭手把手来带大家成为Linux方面的专业人才而设立的，旨在更高效，快速的学习和就业，以培养中高级的Linux管理人才为己任！目前，Linux系统管理员、Linux系统工程师的工作职位需求非常大，圈内经常需要这方面的优秀人才，工资待遇普遍高于8K,高级的Linux系统工程师年薪高达40W. 所以，赶紧系统全面的学习一下吧，阿铭带您进入Linux的世界！ 跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

阿铭出此书的目的就是为了帮助新人快速进入这个行业，市面上大多数的书都是重理论少实战，而且很复杂，因为这样才能体现写书人的本领，一个命令也能给你讲几页纸，而阿铭这个教程会让你学的很轻松，因为阿铭会帮你过滤掉很多用不上的知识，你能学到的都是你以后工作当中要用到的，第一版教材《跟阿铭学Linux》截止目前已下载10000多次，阅读量阿铭实在没法统计，阿铭真心感谢他们的支持，是他们让我再有动力去更新这本书。也许在不久的将来阿铭还会出视频教程，欢迎大家前来捧场！早在一年前就有很多朋友建议我出第二版，但由于工作太忙迟迟未能如愿以偿。两年后的今天，阿铭下定决心重新改进我的教程，让更多的Linux爱好者收益！阿铭只想帮助那些对Linux感兴趣，并且想快速入门的朋友。

阿铭在第一版中，曾提过写这本书的目的是为了写给我第一个学生，让其快速成长。两年过去了，我的这个学生早已经成为公司里不可或缺的核心技术人员，当然薪水也已经上万啦！不瞒大家，我的第一位学生是个女生，虽然毕业于计算机专业，但是她在看我的书之前，对于Linux一点都不懂！是这本书带她入门，改变了她的职业生涯。阿铭写这些，只是为了给大家打打气，Linux入门不难，阿铭有信心带您入门，甚至让您成为一名高级的Linux系统工程师！您是不是计算机专业不重要，您有没有高学历也不重要，重要的是您能够坚持下去，认真的跟阿铭做完书中提到的每一个小实验。

最后，衷心地祝愿所有看这本书的读者朋友，学有所成！

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

TWO

第二章 关于LINUX的历史

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

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

Linux的历史确实有必要让读者了解的，但是不了解也并不会影响您将来的Linux技术水平。哈哈，阿铭其实就不怎么了解Linux的历史，所以对于Linux的历史在本教程中不会涉及到。如果您感兴趣的话，那您去网上搜一下吧，一大堆呢足够让您看一天的。虽然我不太想啰嗦太多，但是关于Linux最基本的认识，我还是想简单介绍一下的。也算是我对Linux的创始人Linus Torvalds 先生的尊重。

在介绍Linux的历史前，我想先针对大家如何对Linux的发音说一下。我发现我身边的朋友对Linux的发音大致有这么几种： “里那克斯”与 “里泥克斯”“里扭克斯”等。其实官方的标准发音为 ['li:nəks]，因为这个发音是创始人Linus的发音。如果您不认识这个音标，那么就读成“里那克斯”。而阿铭习惯发音成“里泥克斯”，当然您发音成什么，并没有人会说您，完全是一个人的习惯而已。

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

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

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

阿铭只是简单的介绍了一下Linux，如果您想详细了解Linux的历史，那么请自己去查询一下相关的资料吧。

教程答疑: 请移步这里.

如果您已经成为阿铭论坛的vip会员了，请访问(http://www.lishiming.net/thread-5385-1-1.html)进一步学习吧。阿铭欢迎您成为成为阿铭学院的一员，和阿铭共同进步！详情请点击(http://www.aminglinux.com)

CHAPTER

THREE

第三章 对LINUX系统管理员的建议

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

Linux系统和windows系统有太多不一样的地方，我相信99%的读者朋友最早接触电脑肯定不是Linux系统，要么是windows要么是苹果操作系统。所以，当您刚刚使用Linux操作系统时，肯定有诸多不习惯的地方，但不要因为这些不习惯而放弃学习Linux. 阿铭最早接触Linux的时候也是感觉诸多不习惯，但是我对Linux的痴迷心使我不仅坚持一直使用Linux，还使我找到了一份和Linux息息相关的工作。可以说，现在我的世界已经离不开Linux.

4.1 要安装什么版本的Linux操作系统

上一章节中，阿铭已经提到过，目前比较流行的Linux有很多种，Redhat，CentOS，Ubuntu，Debain等等。不管您选择哪一种使用，其实都无所谓，阿铭介绍的知识点大多都是通用的。但是，阿铭建议您安装CentOS，因为这个版本的Linux在中国来说用在服务器上的最多，而且它和Redhat是一样的。另外，最主要的原因是因为阿铭在后续章节中所做的所有实验都是在CentOS上来实现的。

4.2 图形界面还是命令窗口

刚刚学习Linux的朋友，使用图形界面是在所难免的，也许是处于对它的好奇心也许是因为不习惯。阿铭早期学习Linux时，安装的Linux操作系统也是从使用图形界面开始的。但是后来意识到Linux的图形界面运行起来远远没有windows或者IOS(苹果操作系统)流畅，所以就不再使用图形界面了。随着阿铭参加工作，使用Linux越来越多，阿铭发现，公司的服务器上根本就没有安装图形界面支持，这是因为，图形界面在Linux操作系统中是作为一个软件来跑的，而且它比较耗费内存。更何况，如果远程连接图形的话还比较耗费带宽，既然命令行能完成的事情为什么还要搞个吃力不讨好的图形支持。最后，阿铭建议读者朋友，从一开始学习Linux起就应该使用命令窗口。

4.3 养成安全严谨的习惯

作为Linux系统管理员，您面对的是服务器而不是自己的计算机。我们在日常管理工作中，做任何一件事情都有可能引起重大事故，所以您一定要养成严谨的习惯。

- 养成备份的习惯

服务器上跑的数据的重要性是不言而喻的，所以，一定要注意数据的安全。我们在做任何操作之前，一定要想清楚，这样做是否是可逆的（操作之后，是否还可以恢复到操作之前的样子）如果不可以，一定要既得备份数据，否则，一旦出错您会后悔死，阿铭在日常工作中就遇到过这样的事故！

- 尽量少使用root

在后面章节中，阿铭还会介绍root这个超级账户，在这里先一笔带过。root相当于windows里面的adminstrator, 它任何权限都有，所以为了避免引起不必要的事故，阿铭劝您还是使用普通用户吧。能用普通用户完成的任务，尽量不要使用root。

- 敲命令不是越快越好

如果您使用了一段时间的Linux, 我相信您会越来越熟练各种命令，敲命令的速度肯定也会越来越快。但是，并不是越快越好，每个人都会有疏忽的时候，一旦敲错了命令那产生的后果是不可预知的。所以，阿铭劝您还是慢点敲键盘吧，如果快也没有关系，但是敲回车的时候一定要检查一下当前的命令是否是您想要的。

- 不要把服务器密码信息记录在文档里

有的朋友会把登陆服务器的密码记录在文档里，而且还存到U盘或者移动硬盘里，这是一件多么不安全的事情，如果U盘或者移动硬盘丢了，被别有用心的人捡到，那后果不堪设想。阿铭给您的建议是，设置一个只有您自己记得住的密码，密码要包含大小写字母和数字，长度要大于8位。不要把密码存到文档里，要记在脑袋里。当然服务器少还好，要是多怎么办？后续章节阿铭会推荐给您一个存密码的工具，而不是记录在文档里。

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

FOUR

第四章 安装LINUX操作系统

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

目前我们安装Linux操作系统是为了更好的了解和学习Linux操作系统。如果您有条件，最好是把Linux安装在一台真正的计算机上，如果您的电脑不允许您安装成Linux，那也没有关系。阿铭教您使用虚拟机来安装Linux操作系统。

5.1 安装虚拟机

当前流行的虚拟机有好多种，阿铭用的是vmware虚拟机。目前最新版本已经到版本v9.0了，但阿铭一直在使用v6.0.1, 如果您不想去下载最新版，那直接使用阿铭提供的链接下载v6.0.1吧，下载地址（含注册机）：点这里 下载后安装一下vmware6.0.1, 安装完后需要输入注册码才可以使用，阿铭提供的注册机很好用，直接生成一个即可。

如果您觉得vmware不好用，您也可以选择VirtualBox，它是完全免费的，官方下载地址 另外也阿铭提供一个下载地址: 这里

5.2 下载Linux操作系统镜像文件

安装虚拟机的好处就是，可以直接把镜像文件当成光盘使用。阿铭在前面提到，在后续章节中做的所有实验都是依据CentOS来做的，所以您也需要下载一个32位的 [1] CentOS的镜像。下载地址：点这里 只需要下载其中的DVD1.iso 即可。

5.3 安装CentOS (图形化安装)

前面的准备工作可能需要很久，因为CentOS的镜像文件太大了。下载好后，就可以跟着阿铭一起安装CentOS了。在这一版教程里，阿铭不再插入图片了，因为在后面阿铭会录制视频来帮助大家更好的学习，这也许会对您的学习造成一点点障碍，但阿铭尽可能的讲清楚给大家，如果您实在不明白就请加一下QQ群（148412746），到群里咨询阿铭吧。阿铭先介绍如何图形化安装CentOS，然后介绍以文本模式的方式安装CentOS.

1. 创建安装Linux的虚拟机

[1] 讲到这里，阿铭有一个小知识点需要向您分享一下，首先要声明这个知识点是需要您掌握的。Linux操作系统或者windows、apple等操作系统都有两种版本: i386 和 X86_64, 这两种版本的区别在于，前者是32位后者是64位。不知道您有没有遇到过这样的问题，32位windows操作系统内存如果安装4G或者更高它是不识别的，只认到3G多，而64位操作系统没有这个影响。同样的，Linux操作系统也是有这样的问题，阿铭提供的下载地址是32位的，因为我们是在做实验，分配1G内存足够了。如果您要是在大于4G的计算机上安装Linux操作系统，那么就需要安装64位的操作系统。

打开vmware workstation软件，在右侧的Home标签中点 ``New Virtual Machine'' --> 点 ``下一步'' --> 选择Custom 继续点 ``下一步'' --> 继续点 ``下一步'' --> 选择 ``Linux'' --> 点 ``下一步'' --> Location 这里选择一个空间大的盘符，比如阿铭选择 ``E:CentOS'' 点 ``下一步'' -->Processors 数量选择One, 继续点 ``下一步'' --> 内存建议修改为1024M或者更大，前提是您的真机内存足够大(不能太小，否则图形安装界面出不来）然后点 ``下一步'' --> Network Connection 保持默认，点 ``下一步'' -->继续点 ``下一步'' --> 继续 ``下一步'' --> Vitual disk type 选择 ``IDE'', 点 ``下一步'' --> Disk size 改为16.0 （如果您的计算机磁盘空间太小，那就改小一点，但不要低于5G）点 ``下一步''--> 最后点 ``完成''

2. 配置虚拟光驱

此时还不能安装CentOS，还需要配置一下刚刚建立的虚拟机。点 ``Edit virtual machine settings'' --> 选择Floppy那一项，点下面的 ``Remove'', 因为用不到软盘了，所以直接删除掉; CD-ROM 这一项是关键，需要在 ``Use ISO image:'' 前面点一下，然后点Browse, 选择您刚刚下载的CentOS镜像文件，最后点ok.

3. 开始安装CentOS

点 ``Start this vitrual machine'' --> 鼠标点一下虚拟机的界面,进入虚拟机，否则不能进行后续操作（要想退出，需要同时按Alt + Ctrl），然后选择 ``Install system with basic video driver'' 回车 --> Disk Found 这是问您是否需要检测镜像文件，因为是从官方站点下载的镜像，肯定没有问题，所以我们选择 ``skip'' --> 点 ``Next'' --> 这一步是让我们选择在接下来的安装过程中需要使用的语言，我们选择 ``chinese(simplified) (中文（简体）)''，点 ``Next'' (到这里如果您发现，和阿铭的不一样，那肯定是您的虚拟机不支持图形显示，那么请直接跳到下一小节参考如何文本模式安装吧) --> 键盘这一项我们保持默认，即美国英式键盘, 点 ``下一步'' --> 使用的设备保持默认，点 ``下一步'' --> 如果弹出一个窗口提示 ``处理驱动器时出错''，则需要我们选择 ``重新初始化(R)'', 然后点 ``下一步'' --> 看一下时区是不是 ``亚洲/上海''，如果是则点 ``下一步''，否则就选择 ``亚洲/上海''，点 ``下一步'' --> 输入根密码（root账户的密码），您一定要记住哦，否则您进不了系统，反正是做实验，阿铭建议您输入123456, 点 ``下一步'' --> 接下来会问我们 ``进行哪种类型的安装''，我们选择最后一项 ``创建自定义布局''，即我们自定义分区，点 ``下一步''

4. 分区

此时就该分区 [2] 了，阿铭的虚拟机总共有一块16G的磁盘，我会这样划分：(1) /boot分区分100M (2) swap分区分2G (3) 剩余划分给 /分区。下面看看阿铭如何操作吧，先点 ``创建'', 然后选择 ``标准分区'', 再点 ``生成'', 挂载点选/boot, 大小改为100, 最后点 ``确定''，这样就创建成功了一个/boot分区 --> 依此类推，创建swap分区，创建时先不用选择挂载点，直接选择文件系统类型为swap，然后大小改为2048, 最后点 ``确定'' --> 最后创建一个/ 分区，挂载点选择 /, 大小不管，其他大小选项选择 ``使用全部可用空间'', 最后点 ``确定''，这样就按照阿铭先前构想的方案创建好分区了。然后点 ``下一步''。 此时弹出警告，格式化会删除掉磁盘上的所有数据，我们选择 ``格式化'', 然后又一次弹出警告，点击 ``将修改写入磁盘(W)''

5. 选择安装包

格式化完磁盘后，出现引导装载程序的选项，保持默认点击 ``下一步'' --> 目前的安装方案会最小化安装CentOS, 为了让大家更能详细了解安装过程，所以阿铭带大家选择 ``现在自定义''，点 ``下一步'' --> ``基本系统''这一项，在右侧选择 ``基本''，其他不用勾选 --> web服务、开发、弹性存储、数据库、服务器、系统管理、虚拟化、负载平衡器、高可用这几项一个都不用选 --> 桌面这一项阿铭也不会选择任何一项，但如果您对图形界面好奇，或者想用一用Linux的图形界面，那您就把右侧的几个项全部勾选了吧 --> 应用程序这一项把 ``Tex支持''勾选上 --> 语言支持这一项默认就已经把 ``中文支持'' 勾选上了，所以我们也不用修改它，接着点 ``下一步''，接下来就是漫长的等待安装系统了。安装时间的长短取决您勾选的包的多少以及您的电脑的性能，阿铭的虚拟机安装了8分钟就搞定了。等安装完所有的包后会提示我们，需要 ``重新引导''，重启后登陆root账户，进入CentOS系统。

[2] 关于Linux的分区，往往Linux系统管理员都有些讲究，肯定不能像阿铭在虚拟机上这样简单分区。以后的您也许会做系统管理员，所以如何分区是需要您掌握的。阿铭就给大家一个万能的分区方案吧：(1) /boot分区给100M; (2) swap分区大小为内存的2倍，如果您的机器内存非常大（大于8G时）给16G就够了，给多了也是浪费磁盘资源，8G或者8G以下就给内存大小的2倍；(3)/usr/分区20G; (4) / 分区给20G; (5)剩下的给/data/分区或者随便您自定义一个挂载点。

5.4 安装CentOS (文本模式安装)

首先是安装虚拟机、配置虚拟机，步骤和上一小节的一样，不同的是光驱启动，进入安装界面后，我们不再选择 ``Install system with basic video driver'' 而是选择第一行 ``Install or upgrade an existing system''. 回车后出现 Disk Found提示，问我们是否要检测镜像文件，我们选择 ``skip''.

- 等会弹出欢迎页： Welcome to CentOS! 直接回车。
- 选择语言，因为使用的是文本模式，选择中文也没有用，所以直接回车。
- 键盘也保持默认，直接回车。
- 此时出现警告，提示说磁盘设备可能需要重新初始化，选择 ``Re-initialize''.
- Time Zone Selection 这一项是时区设置选项，System clock uses UTC 前面的 * 要去掉（按空格即可以去掉），因为中国的时区是CST，然后按 ``Tab'' 键光标移动到下方，按向下箭头找到 ``Asia/Shanghai'', 然后按 ``Tab'' 选择OK 回车。
- Root Password 这一栏是给root用户设置密码。阿铭建议您输入一个简单的容易记住的，不然等会忘记是什么就麻烦了。比如阿铭设置的密码是 `123456', 输入完后按 ``Tab'' 再次输入一遍，然后按 ``Tab'' 选中OK回车。
- 此时会提示，说密码太弱了，让我们设置一个复杂的，这也是出于安全考虑友善提示用户的，不过咱们是做实验，无所谓了。所以选择 ``Use Anyway'' 回车。
- 弹出 ``Partition Type'' (分区类型) 窗口， 分为三种模式：Use entire drive 意思是说使用全部的磁盘；Replace existing Linux system 意思是替换现有的Linux系统； Use free space 意思是使用剩余空间。由于我们是全新安装，所以无所谓选择哪一项。 阿铭选择的是第一项 ``Use entire drive''. 下面的一行是让我们选择哪个磁盘，如果您的机器上有多块磁盘，这里会显示多行，阿铭的虚拟机只有一个磁盘，所以使用 ``Tab'' 选择Ok回车。
- 弹出提示说，将要把存储配置写入磁盘里。我们选择 ``Write changes to disk'' 回车。
- 剩下的就是自动安装了，大概3分钟左右就完成了安装。

也许您会问，那分区、自定义软件包的步骤跑哪里去了？是呀，阿铭一开始也有疑问，当我查了好久资料后，才知道，原来是官方为了某个bug而屏蔽了自定义分区和自定义安装包的功能。所以，想自定义分区的朋友，只能通过图形化安装喽！

第四章扩展学习: http://www.lishiming.net/thread-5386-1-1.html

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

FIVE

第五章 初步认识LINUX

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

安装完CentOS后，相信您会迫不及待的要登陆进系统里看看这神秘的Linux到底是个什么东西？如果您安装了图形界面，相信您进入系统后会不知所措，总是想点击鼠标的右键去刷新桌面，阿铭最早也是这样的，毕竟用了很久的windows，突然换个操作系统不习惯了。有的朋友也许没有安装图形界面，输入用户名(root)以及密码(123456)后进去，发现更不知所错了。没有关系，随着阿铭逐渐深入的讲解，您会学到越来越多的知识。

6.1 CentOS6是如何启动的

Linux的启动其实和windows的启动过程很类似，不过windows我们是无法看到启动信息的，而Linux启动时我们会看到许多启动信息，例如某个服务是否启动。Linux系统的启动过程大体上可分为五部分：内核的引导、运行init、系统初始化、建立终端、用户登录系统。

1. 内核引导

当计算机打开电源后，首先是BIOS开机自检，按照BIOS中设置的启动设备（通常是硬盘）来启动。紧接着由启动设备上的grub程序开始引导Linux，当引导程序成功完成引导任务后，Linux从它们手中接管了CPU的控制权，然后CPU就开始执行Linux的核心映象代码，开始了Linux启动过程。也就是所谓的内核引导开始了，在内核引导过程中其实是很复杂的，我们就当它是一个黑匣子，反正是Linux内核做了一些列工作，最后内核调用加载了init程序，至此内核引导的工作就完成了。交给了下一个主角init.

2. 运行init

init 进程是系统所有进程的起点，您可以把它比拟成系统所有进程的老祖宗，没有这个进程，系统中任何进程都不会启动。init 最主要的功能就是准备软件执行的环境，包括系统的主机名、网络设定、语言、文件系统格式及其他服务的启动等。 而所有的动作都会通过 init的配置文件/etc/inittab来规划，而inittab 内还有一个很重要的设定内容，那就是默认的 runlevel (开机运行级别)。先来看看运行级别Run level,Linux就是通过设定run level来规定系统使用不同的服务来启动，让Linux的使用环境不同。我们来看看这个inittab文件里面的支持级别。

```
# inittab is only used by upstart for the default runlevel.
#
# ADDING OTHER CONFIGURATION HERE WILL HAVE NO EFFECT ON YOUR SYSTEM.
#
# System initialization is started by /etc/init/rcS.conf
#
# Individual runlevels are started by /etc/init/rc.conf
#
# Ctrl-Alt-Delete is handled by /etc/init/control-alt-delete.conf
#
# Terminal gettys are handled by /etc/init/tty.conf and /etc/init/serial.conf,
```

```
# with configuration in /etc/sysconfig/init.
#
# For information on how to write upstart event handlers, or how
# upstart works, see init(5), init(8), and initctl(8).
#
# Default runlevel. The runlevels used are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
id:3:initdefault:
```

inittab配置文件格式和之前老版本CentOS5或者更老版本比有很大改动。Runlevels共七个级别，0表示关机，1表示单用户，2表示没有网络的命令行级别，3命令行级别（大多服务器都用这个级别），4为保留级别，5为图形化级别，6为重启。这个文件中除了最后一行外，其他都为注释行，也就是说最后一行才是关键，它用来指定服务器跑哪个级别，这里除了可以设置2,3,5外其他级别都不能设置。在该文件的前面部分，可以看到很多行都提及到某个配置文件，而所有配置文件都是在/etc/init/目录下。

3. 系统初始化

系统初始化，就是去执行/etc/init/下的各个配置文件。inittab配置文件中有这么一行 ``System initialization is started by /etc/init/rcS.conf'' 也就是说系统初始化会先执行/etc/init/rcS.conf 而该配置文件中又有一行 ``exec /etc/rc.d/rc.sysinit'' 所以，重心又转移到了这个rc.sysinit文件上，它会做如下工作：激活交换分区，检查磁盘，加载硬件模块以及其它一些需要优先执行任务。当rc.sysinit程序执行完毕后，将返回init继续下一步，又到了/etc/init/rc.conf, 在这个配置文件里，最关键的一行为 ``exec /etc/rc.d/rc $RUNLEVEL'' 而$RUNLEVEL是在/etc/inittab中定义的(最下面的那一行)，以阿铭的/etc/inittab为例，表示$RUNLEVE=3, 所以此时会执行 ``/etc/rc.d/rc 3'' 此时实际上是把/etc/rc.d/rc3.d/ 下的脚本都给执行了，随后/etc/rc.d/rc.local也会被执行，通常我们会把开机启动执行的命令放到这个脚本下。服务执行完，系统初始化也就完成了。接下来该建立终端了。

4. 建立终端

建立终端是由配置文件/etc/init/tty.conf, /etc/init/serial.conf和/etc/sysconfig/init等配置文件来完成的。在2、3、4、5的运行级别中都将以respawn方式运行mingetty程序，mingetty程序能打开终端、设置模式。同时它会显示一个文本登录界面，这个界面就是我们经常看到的登录界面，在这个登录界面中会提示用户输入用户名，而用户输入的用户将作为参数传给login程序来验证用户身份。

5. 用户登陆系统

对于运行级别为5的图形方式用户来说，他们的登录是通过一个图形化的登录界面。登录成功后可以直接进入KDE、Gnome等窗口管理器。而本文主要讲的还是文本方式登录的情况：当我们看到mingetty的登录界面时，我们就可以输入用户名和密码来登录系统了。

Linux的账号验证程序是login，login会接收mingetty传来的用户名作为用户名参数。然后login会对用户名进行分析：如果用户名不是root，且存在 ``/etc/nologin'' 文件，login将输出nologin文件的内容，然后退出。这通常用来系统维护时防止非root用户登录。只有 ``/etc/securetty'' 中登记了的终端才允许root用户登录，如果不存在这个文件，则root可以在任何终端上登录。''/etc/usertty'' 文件用于对用户作出附加访问限制，如果不存在这个文件，则没有其他限制。

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

login程序成功后，会向对应的终端在输出最近一次登录的信息(在 ``/var/log/lastlog'' 中有记录)，并检查用户是否有新邮件(在 ``/usr/spool/mail/'' 的对应用户名目录下)。然后开始设置各种环境变

量：对于bash来说，系统首先寻找 ``/etc/profile'' 脚本文件，并执行它；然后如果用户的主目录中存在 .bash_profile 文件，就执行它，在这些文件中又可能调用了其它配置文件，所有的配置文件执行后后，各种环境变量也设好了，这时会出现大家熟悉的命令行提示符，到此整个启动过程就结束了。

6.2 图形界面与命令行界面切换

Linux预设提供了六个命令窗口终端机让我们来登录。默认我们登录的就是第一个窗口，也就是tty1，这个六个窗口分别为tty1,tty2 ⋯ tty6，您可以按下Ctrl + Alt + F1 ~ F6 来切换它们。如果您安装了图形界面，默认情况下是进入图形界面的，此时您就可以按Ctrl + Alt + F1 ~ F6来进入其中一个命令窗口界面。当您进入命令窗口界面后再返回图形界面只要按下Ctrl + Alt + F7 就回来了。如果您用的vmware 虚拟机，命令窗口切换的快捷键为 Alt + Space + F1~F6. 如果您在图形界面下请按Alt + Shift + Ctrl + F1~F6 切换至命令窗口。

6.3 学会使用快捷键

1. Ctrl + C：这个是用来终止当前命令的快捷键，当然您也可以输入一大串字符，不想让它运行直接Ctrl + C，光标就会跳入下一行。
2. Tab： 这个键是最有用的键了，也是阿铭敲击概率最高的一个键。因为当您打一个命令打一半时，它会帮您补全的。不光是命令，当您打一个目录时，同样可以补全，不信您试试。
3. Ctrl + D： 退出当前终端，同样您也可以输入exit。
4. Ctrl + Z： 暂停当前进程，比如您正运行一个命令，突然觉得有点问题想暂停一下，就可以使用这个快捷键。暂停后，可以使用fg 恢复它。
5. Ctrl + L： 清屏，使光标移动到第一行。

6.4 学会查询帮助文档 — man

这个man 通常是用来看一个命令的帮助文档的。格式为 '' man 命令 '' 例如输入命令：

```
man ls
```

则会显示如下结果：

```
LS(1)                            User Commands                           LS(1)

NAME
    ls - list directory contents

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

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

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

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

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

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

这样可以查看 ``ls'' 这个命令的帮助文档，进入后按 `q' 键退出。

6.5 Linux 系统目录结构

登录系统后，在当前命令窗口下输入:

```
ls /
```

您会看到

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

在讲目录结构之前，阿铭先介绍一下这个 ``ls'' 命令是干什么的， ``ls'' 其实就是英文单词 `list' 的缩写，它的作用是列出指定目录或者文件，刚刚在上一节中提到的 ``man ls'' 可以查看其具体的使用方法。对于 ``ls'' 这个最常用的命令，阿铭举几个简单例子让您快速掌握。

Example:

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

[root@localhost ~]# ls -a
.   anaconda-ks.cfg  .bash_profile  .cshrc       install.log.syslog
..  .bash_logout     .bashrc        install.log  .tcshrc

[root@localhost ~]# ls -l
总用量 36
-rw-------. 1 root root   980 5月   7 18:00 anaconda-ks.cfg
-rw-r--r--. 1 root root 19305 5月   7 18:00 install.log
-rw-r--r--. 1 root root  5890 5月   7 17:58 install.log.syslog

[root@localhost ~]# ls install.log
install.log

[root@localhost ~]# ls /var/
account  crash  empty  lib    lock  mail  opt       run    tmp
cache    db     games  local  log   nis   preserve  spool  yp
```

讲解

1. 不加任何选项也不跟目录[1]名或者文件名

   会列出当前目录下的文件和目录，不包含隐藏文件。[2]

2. 加 ``-a'' 选项不加目录名或者文件名

   会列出当前目录下所有文件和目录，含有隐藏文件。

[1] Linux下的目录其实就是windows里的文件夹，但通常我们都叫做目录，而不叫文件夹，希望您也要改一下这个习惯。
[2] Linux下的隐藏文件是通过文件名来控制的，所有的隐藏文件的文件名都是以 . 开头的，比如 .1.txt 这样1.txt就是隐藏文件了。

3. 加 ``-l'' 选项不加目录名或者文件名

   会列出当前目录下除隐藏文件外的所有文件和目录的详细信息，包含其权限、所属主、所属组、以及文件创建日期和时间。

4. 后面不加选项只跟文件名

   会列出该文件，其实这样没有什么意思，通常都是加上一个 ``-l'' 选项，用来查看该文件的详细信息。

5. 后面不加选项只跟目录名

   会列出指定目录下的文件和目录

好了，关于''ls'' 阿铭就讲这几个例子，当然它的可用选项还有很多哦，不过阿铭只给介绍最常用的。因为阿铭不想一股脑灌输给您太多知识点，那样没有什么用，但您也不用担心学不全，阿铭讲的知识点足够您日常工作和学习中用的了。如果实在是遇到不懂的选项，直接用 ``man'' 来查帮助文档吧。下面咱们回到先前的话题，接着讨论Linux的目录结构。

- /bin bin是Binary的缩写。这个目录存放着最经常使用的命令。
- /boot这里存放的是启动Linux时使用的一些核心文件，包括一些连接文件以及镜像文件。
- /dev dev是Device(设备)的缩写。该目录下存放的是Linux的外部设备，在Linux中访问设备的方式和访问文件的方式是相同的。
- /etc这个目录用来存放所有的系统管理所需要的配置文件和子目录。
- /home用户的主目录，在Linux中，每个用户都有一个自己的目录，一般该目录名是以用户的账号命名的。
- /lib这个目录里存放着系统最基本的动态连接共享库，其作用类似于Windows里的DLL文件。几乎所有的应用程序都需要用到这些共享库。
- /lost+found这个目录一般情况下是空的，当系统非法关机后，这里就存放了一些文件。
- /media Linux系统会自动识别一些设备，例如U盘、光驱等等，当识别后，Linux会把识别的设备挂载到这个目录下。
- /mnt系统提供该目录是为了让用户临时挂载别的文件系统的，我们可以将光驱挂载在/mnt/上，然后进入该目录就可以查看光驱里的内容了。
- /opt 这是给主机额外安装软件所摆放的目录。比如您安装一个ORACLE数据库则就可以放到这个目录下。默认是空的。
- /proc这个目录是一个虚拟的目录，它是系统内存的映射，我们可以通过直接访问这个目录来获取系统信息。这个目录的内容不在硬盘上而是在内存里，我们也可以直接修改里面的某些文件，比如可以通过下面的命令来屏蔽主机的ping命令，使别人无法ping您的机器：

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

- /root该目录为系统管理员，也称作超级权限者的用户主目录。
- /sbin s就是Super User的意思，这里存放的是系统管理员使用的系统管理程序。
- /selinux 这个目录是Redhat/CentOS所特有的目录，Selinux是一个安全机制，类似于windows的防火墙，但是这套机制比较复杂，这个目录就是存放selinux相关的文件的。
- /srv 该目录存放一些服务启动之后需要提取的数据。
- /sys 这是linux2.6内核的一个很大的变化。该目录下安装了2.6内核中新出现的一个文件系统 sysfs ，sysfs文件系统集成了下面3种文件系统的信息：针对进程信息的proc文件系统、针对设备的devfs文件系统以及针对伪终端的devpts文件系统。该文件系统是内核设备树的一个直观反映。当一个内核对象被创建的时候，对应的文件和目录也在内核对象子系统种被创建。

- /tmp这个目录是用来存放一些临时文件的。
- /usr 这是一个非常重要的目录，用户的很多应用程序和文件都放在这个目录下，类似与windows下的program files目录。
- /usr/bin 系统用户使用的应用程序。
- /usr/sbin 超级用户使用的比较高级的管理程序和系统守护程序。
- /usr/src 内核源代码默认的放置目录。
- /var这个目录中存放着在不断扩充着的东西，我们习惯将那些经常被修改的目录放在这个目录下。包括各种日志文件。

在Linux系统中，有几个目录是比较重要的，平时需要注意不要误删除或者随意更改内部文件。/etc：上边也提到了，这个是系统中的配置文件，如果您更改了该目录下的某个文件可能会导致系统不能启动。/bin, /sbin, /usr/bin, /usr/sbin: 这是系统预设的执行文件的放置目录，比如 ls 就是在/bin/ 目录下的。值得提出的是，/bin, /usr/bin 是给系统用户使用的指令（除root外的通用账户），而/sbin, /usr/sbin 则是给root使用的指令。 /var： 这是一个非常重要的目录，系统上跑了很多程序，那么每个程序都会有相应的日志产生，而这些日志就被记录到这个目录下，具体在/var/log 目录下，另外mail的预设放置也是在这里。

6.6 如何正确关机、重启

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

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

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

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

- sync 将数据由内存同步到硬盘中。
- shutdown 关机指令，您可以man shutdown 来看一下帮助文档。例如您可以运行如下命令关机：
- shutdown -h 10 ‘ 这个命令告诉大家，计算机将在10分钟后关机，并且会显示在登陆用户的当前屏幕中。
- shutdown -h now 立马关机
- shutdown -h 20:25 系统会在今天20:25关机
- shutdown -h +10 十分钟后关机
- shutdown -r now 系统立马重启
- shutdown -r +10 系统十分钟后重启
- reboot 就是重启，等同于 shutdown -r now
- halt 关闭系统，等同于shutdown -h now 和 poweroff

最后总结一下，不管是重启系统还是关闭系统，首先要运行sync命令，把内存中的数据写到磁盘中。关机的命令有 shutdown -h now, halt, poweroff 和 init 0，重启系统的命令有 shutdown -r now, reboot, init 6.

6.7 忘记root密码怎么办

以前阿铭忘记windows的管理员密码，由于不会用光盘清除密码最后只能重新安装系统。现在想想那是多么愚笨的一件事情。同样Linux系统您也会遇到忘记root密码的情况，如果遇到这样的情况怎么办呢？重新安装系统吗？当然不用！进入单用户模式更改一下root密码即可。如何进入呢。

1. 重启系统

3秒钟内，按一下回车键。此时您会看到如下提示信息：

```
GNU GRUB version 0.97 .......
CentOS (2.6.32-358.el6.i686)
```

阿铭没有写完全提示信息，相信您肯定能进入到这一界面。此时CentOS (2.6.32-358.el6.i686) 这一行是高亮的，即我们选中的就是这一行，这行的意思是Linux版本为CentOS，后面小括号内是内核版本信息。另外在这个界面里，我们还可以获取一些信息，输入 `e' 会在启动前编辑命令行； 输入 `a' 会在启动前更改内核的一些参数； 输入 `c' 则会进入命令行。而我们要做的是输入 `e'.

2. 进入单用户模式

输入 `e' 后，界面变了，显示如下信息：

```
root (hd0,0)
kernel /vmlinuxz-2.6.32-358.el6.i686 ro root=UUID=......（此处省略）
initrd /initramfs-2.6.32-358.el6.i686.img
```

暂时您不用管这些都代表什么意思，您只要跟着阿铭做即可。按一下向下的箭头键，选中第二行，输入 `e'，出现如下提示：

```
<_NO_DM rhgh quiet
```

您只需要在后面加一个 ``single'' 或者 ``1'' 或者 ``s''

```
<_NO_DM rhgh quiet single
```

然后先按回车然后按 `b'，启动后就进入单用户模式。

3. 修改root密码

输入修改root密码的命令 `passwd'：

```
[root@localhost /]# passwd
Changeing password for user root.
New password:
Retry new password:
passwd: all authentication tokens updated successfully.
```

修改后，重启系统

```
[root@localhost /]# reboot
```

6.8 使用系统安装盘的救援模式

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

1. 光盘启动

阿铭做实验用的是vmware虚拟机，设置光盘启动的步骤也许和您的不一样，但道理是一样的，相信聪明的您一定不会在这里出问题。开机启动，按F2进入bios设置，按方向键选择 ``Boot'' 那一项，然后使用上下方向键和+/-号来移动，最终让CD-ROM Drive 挪动到最上面一行。最后按F10，再按回车进入光盘启动界面。

2. 进入rescue模式

   光盘启动后，使用上下方向键选择 `Rescue installed system' 回车

   - 语言我们默认，直接回车
   - 键盘类型，也默认，直接回车
   - Rescue Method 也保持默认，因为我们使用的就是光驱里的光盘，回车
   - 这一步问我吗是否在使用rescue模式的时候启用网络，这个根据实际情况，在这里阿铭选择NO(使用tab键) 回车
   - 接下来这一步，提示我们Rescue 环境将会找到我们已经安装的Linux系统，并将其挂载到/mnt/sysimage 下，这一步阿铭将会选择 `Continue' 然后回车
   - 回车后，将会看到一个小提示框，它告诉我们Linux系统挂载到了 ``/mnt/sysimage''．如果想获得root环境，需要执行命令 ``chroot /mnt/sysimage'' 继续回车
   - 继续回车
   - 此时又出现一个框，有三种模式可以选择：shell 模式会直接进入命令行，可以进行的操作有编辑文件、修改用户密码等； fakd 是诊断模式； reboot 会直接重启； 这一步阿铭选择第一个shell模式，然后回车

3. 进入root环境

此时还不能操作Linux系统上的文件，因为目前还在光盘上的系统上，您有没有使用过windows PE系统？其实目前我们所在的环境就类似于windows上的PE系统。要想修改原来Linux系统上的文件还需要执行一个命令：

```
bash-4.1# chroot /mnt/sysimage
sh-4.1#
```

您会发现命令行前后有一处变化：原来的 ``bash-4.1'' 变成了 ``sh-4.1''，此时我们才可以像在原来的Linux系统上做一些操作，比如更改root密码或者修改某个文件等等。

本章内容阿铭讲的有点罗嗦，另外建议您再扩展学习一下吧(http://www.lishiming.net/thread-5396-1-1.html).

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

SIX

第六章 LINUX系统的远程登陆

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

Linux大多应用于服务器，而服务器不可能像PC一样放在办公室，它们是放在IDC机房的，所以阿铭平时登录Linux系统都是通过远程登录的。Linux系统中是通过ssh服务实现的远程登录功能。默认sshd服务开启了22端口，而且当我们安装完系统时，这个服务已经安装，并且是开机启动的。所以不需要我们额外配置什么就能直接远程登录Linux系统。sshd服务的配置文件为 /etc/ssh/sshd_config，您可以修改这个配置文件来实现您想要的sshd服务。比如您可以更改启动端口为11587.

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

您不妨跟着阿铭一起来用一用Putty这个小巧的工具。

7.1 下载Putty

阿铭建议您到Putty的官方站点去下载英文版原版的putt. 网上曾经报过，某个中文版的Putt被别有用心的黑客给动了手脚，给植了后门。所以，阿铭提醒各位，以后不管下载什么软件尽量去官方站点下载。阿铭给出下载地址： http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html 连接远程Linux服务器的工具只需要下载putty.exe即可 http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe 下载后直接双击运行就可以了不需要安装。

7.2 给您的Linux配置IP

要想远程连接Linux服务器，首先需要知道服务器的IP。因为阿铭用的虚拟机，而且虚拟机所跑的真机是自动获得的ip，所以虚拟机也可以自动获得ip。如果您是一步一步跟阿铭装的Linux那么您的Linux目前肯定是没有IP的，下面阿铭教您几种配置IP的方法：

1. 自动获取IP

只有一种情况可以自动获取IP地址，那就是您的Linux所在的网络环境中有DHCP服务。[1] 总之，只要您的真机可以自动获取IP，那么安装在虚拟机的Linux同样也可以自动获取IP. 方法很简单，只需要运行一个命令。

[1] DHCP服务，是自动分配IP的服务，我们平时所在的办公室网络环境里，都有DHCP服务。另外家用的路由器像Tplink 或者 dlink 都有DHCP服务的功能。

```
[root@localhost ~]# dhclient
```

运行这条命令后，会出现一大堆信息，您不用关心是什么。然后运行 `ifconfig' 命令查看IP是什么：

```
[root@localhost ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:D9:F0:52
          inet addr:10.72.137.85  Bcast:10.72.137.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fed9:f052/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:27135 errors:0 dropped:0 overruns:0 frame:0
          TX packets:53 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3488498 (3.3 MiB)  TX bytes:7550 (7.3 KiB)
          Interrupt:18 Base address:0x1080

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

通过这个命令可以查看系统有几块网卡和网卡的IP，阿铭的系统eth0的IP是 10.72.137.85. 如果您的Linux有多块网卡，那么在Linux中它会显示成eth1, eth2 依此类推。

2. 手动配置IP

如果您的虚拟机不能自动获取IP，那么只能手动配置，配置方法为：

```
[root@localhost ~]# vi /etc/sysconfig/network-scripts/ifcfg-eth0
```

使用vi 命令打开 ``/etc/sysconfig/network-scripts/ifcfg-eth0'' 这个配置文件。关于命令 vi 阿铭会在后续章节详细介绍，暂时您只要了解这个命令是用来编辑文件的即可。输入上述命令后回车，打开了该配置文件。使用方向键的向下箭头让光标移动到最后面一行，然后按字母键 `o'，进入编辑模式，增加如下内容：

```
IPADDR=10.72.137.85
NETMASK=255.255.255.0
GATEWAY=10.72.137.1
```

请注意，由于阿铭不知道您的网络具体环境，所以也不晓得您应该配置什么样的IP，请不要直接照搬阿铭给出的例子，这样配置肯定是不行的，请配置成和您的真机（windows）在同一个网段的IP。关于netmask以及gateway的概念请自行在网上查询，这是关于网络技术的基础知识。另外还需要把光标移动到 ``ONBOOT=no'' 这一行，改为:

```
ONBOOT=yes
```

``BOOTPROTO=dhcp'' 改为:

```
BOOTPROTO=none
```

之后按一下键盘左上角的 ``ESC''键，然后输入 :wq，它会显示在屏幕的左下方，然后按回车，这样就保存该配置文件了。之后，需要重启一下网络服务：

```
[root@localhost ~]# service network restart
正在关闭接口 eth0：                                        [确定]
关闭环回接口：                                             [确定]
```

```
弹出环回接口：                                             [确定]
弹出界面 eth0：                                            [确定]
```

这样网络重启后，eth0 的IP就生效了。使用 ``ifconfig eth0'' 命令查看一下：

```
[root@localhost ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0C:29:D9:F0:52
          inet addr:10.72.137.85  Bcast:10.72.137.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fed9:f052/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:27135 errors:0 dropped:0 overruns:0 frame:0
          TX packets:53 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3488498 (3.3 MiB)  TX bytes:7550 (7.3 KiB)
          Interrupt:18 Base address:0x1080
```

接下来请检测一下您配置的IP是否可以ping通。阿铭使用的windows7系统，所以使用cmd打开命令窗口，进行检测。打开cmd的快捷键是 windows + r.

```
C:\Users\Administrator>ping 10.72.137.85

正在 Ping 10.72.137.85 具有 32 字节的数据:
来自 10.72.137.85 的回复: 字节=32 时间=1ms TTL=64
来自 10.72.137.85 的回复: 字节=32 时间<1ms TTL=64
来自 10.72.137.85 的回复: 字节=32 时间<1ms TTL=64
来自 10.72.137.85 的回复: 字节=32 时间<1ms TTL=64

10.72.137.85 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms, 最长 = 1ms, 平均 = 0ms
```

3. 利用vmware的NAT给Linux配置IP

这一部分内容，阿铭曾经在论坛里写过一个帖子 http://www.lishiming.net/thread-626-1-1.html 如果您已经配置好IP且可以ping通，这一部分设置则不需要再做了，但有必要了解一下，也许有一天您会用到。这一部分配置适合这样的场景：您的办公网不能通过dhcp获得IP，或者您不想让您的Linux处在和办公网一个网段，而且也想让Linux上网。

- 设置虚拟机上的nat

  Edit --> Virtual Network setting --> NAT --> Vmnet 8 Gateway IP address：192.168.205.2 Netmask：255.255.255.0 NAT service: Started --> 确定

- 修改虚拟机的网卡设置

  双击虚拟机右下角的网卡小图标，鼠标移动过去后会显示 ``Ethernet: ...'' Device status 那两项都需要打对钩； Network connection 需要选择最后一项(Custom:Specific virtual network) 选择 Vmnet8(NAT) 最后点ok

- 到您的电脑上 [2]

  右击''网上邻居'' --> 属性 --> 右击 ``VMware Network Adapter VMnet8'' --> 属性 --> 双击 ``Internet 协议（TCP/IP)'' --> 手动设置IP为 192.168.205.1 子网掩码为 255.255.255.0 网关 和 dns 都设置为 192.168.205.2 --> 确定 --> 确定

- 设置您虚拟机IP

[2] 这一部分阿铭是在windows XP上配置的，windows7 的配置道理也是一样的。

在您的Linux上编辑eth0的配置文件 `vi /etc/sysconfig/network-scripts/ifcfg-eth0` 内容如下：

```
DEVICE=eth0
BOOTPROTO=none
HWADDR=00:0C:29:33:F7:3A
ONBOOT=yes
IPADDR=192.168.205.3
NETMASK=255.255.255.0
GATEWAY=192.168.205.2
```

- 设置DNS地址

  运行命令 `vi /etc/resolv.conf` 内容如下：

```
nameserver 192.168.205.2
```

- 重启网络服务

  运行命令 `service network restart`

如果您遇到类似这样的问题：重启网络服务后，发现/etc/resolv.conf中设置的DNS地址消失了，您可以参考这个帖子解决您的问题：http://www.lishiming.net/thread-5548-1-1.html

7.3 用putty登陆您的Linux

上一小节阿铭带您设置IP，就是给这一部分做铺垫，没有IP是没有办法远程连接Linux的。双击先前下载的putty.exe文件，这个小工具特别小巧仅仅有几百K，但是您可不要小看它，功能可是不少呢，而且这个工具的帮助文档够您看好几天的了，关键是全都是英文。如果您的英文能力差一些也没有关系，相信随着您用Linux越来越多，您的英文能力也会越来越强。

- 填写远程Linux基本信息

  Host Name (or IP address) 这一栏填写您在上一小节刚刚配置的IP，阿铭的Linux IP为``10.72.137.85''.

  Port 这一栏保持默认不变。

  Connection type 也保持默认。

  Saved Sessions 这里自定义一个名字，主要用来区分主机，因为将来您的主机会很多，写个简单的名字即方便记忆又能快速查找。

- 定义字符集

  计算机里最烦人的就是字符集了，尤其是Linux，搞不好就会乱码。阿铭在第三章教您安装CentOS时已经安装了中文语言支持，所以安装好的系统是支持中文的，在putty这里设置也要支持中文。点一下左侧的 ``Window'' --> ``Translation'', 看右侧的 ``Character set translation on received data'', 选择UTF-8. 之后再点一下左侧的 ``Session'', 然后点右侧的 ``save''.

- 远程连接您的Linux

  保存session后，点最下方的 ``Open''. 初次登陆时，都会弹出一个友情提示，它的意思是要打开的Linux还未在本机登记，问我们是否要信任它。如果是可信任的，则点 `是' 登记该主机，否则点 `否' 或者 `取消'，我们当然要点 `是'. 之后弹出登陆提示：

```
login as: root
root@10.72.137.85's password:
Last login: Wed May  8 08:02:17 2013 from 10.72.137.89
```

输入用户名以及密码后，就登陆Linux系统了。登陆后会提示最后一次登陆系统的时间以及从哪里登陆。

7.4 使用密钥认证机制远程登录Linux

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

1. 下载生成密钥工具

在本章前面阿铭提供的putty下载地址里，您一定看到了很多可以下载的东西，不过阿铭只让您下载了一个putty.exe. 因为当时只用到这一个工具，其实完整的putty程序包含很多个小工具的，所以这次阿铭建议您直接下载个完整包 http://the.earth.li/~sgtatham/putty/latest/x86/putty.zip 下载后解压，其中puyttygen.exe就是咱们这一小节中所要用到的密钥生成工具。

2. 生成密钥对

关于密钥的工作原理，如果您感兴趣可以到网上查一查，阿铭不想介绍太多无关知识点，不过，了解一下也没有什么不好。双击puttygen.exe, 右下角 ``Number of bits in a generated key'' 把 ``1024'' 改成 ``2048'', 然后点 ``Generate'', 这样就开始生成密钥了，请来回动一下鼠标，这样才可以快速生成密钥对，大约十几秒后就完成了。 ``Key comment:'' 这里可以保持不变也可以自定义，其实就是对该密钥的简单介绍； ``Kye passphrase:'' 这里用来给您的密钥设置密码，这样安全一些，当然也可以留空，阿铭建议您设置一个密码；''Confirm passphrase:'' 这里再输入一遍刚刚您设置的密码。

3. 保存私钥

点 ``Save private key'', 选择一个存放路径，定义一个名字，点 ``保存''。请保存到一个比较安全的地方，谨防丢掉或被别人看到。

4. 复制公钥到Linux

回到刚才生成密钥的窗口，在 ``Key'' 的下方有一段长长的字符串，这一串就是公钥的内容了，把整个公钥字符串复制下来。然后粘贴到您的Linux的 `/root/.ssh/authorized_keys` 文件里。下面请跟着阿铭一起来做操作：

```
[root@localhost ~]# mkdir /root/.ssh
[root@localhost ~]# chmod 700 /root/.ssh
```

首先创建/root/.ssh 目录，因为这个目录默认是不存在的，然后是更改权限。 关于 `mkdir` 和 `chmod` 两个命令，阿铭会在后续章节详细介绍，暂时您只要知道是用来创建目录和更改权限的就行了。然后是把公钥内容粘贴进 /root/.ssh/authorized_keys 文件。

```
[root@localhost ~]# vi /root/.ssh/authorized_keys
```

回车后，按一下 `i' 进入编辑模式，然后直接点击鼠标右键就粘贴了，这是putty工具非常方便的一个功能。粘贴后，按一下 `Esc' 键，然后输入 `:wq` 回车保存退出该文件。

5. 关闭Selinux

如果不关闭selinux, [3] 使用密钥登陆会提示 ``Server refused our key'', 关闭方法：

```
[root@localhost ~]# setenforce 0
```

[3] selinux是Redhat、CentOS特有的安全机制，这个东西很复杂，我们从来都不要开启它，因为selinux开启后，会产生诸多莫名其妙的bug. 在后续章节中，阿铭会详细介绍它的。

这个只是暂时命令行关闭selinux，下次重启Linux后selinux还会开启。永久关闭selinux的方法是：

```
[root@localhost ~]# vi /etc/selinux/config
```

回车后，把光标移动到 ``SELINUX=enforcing'' 按一下 i 键，进入编辑模式，修改为

```
SELINUX=disabled
```

按 ``Esc''，输入 :wq 回车，然后重启系统

6. 设置putty通过密钥登陆

打开putty.exe点一下您保存好的session，然后点右侧的 ``Load''，在左侧靠下面点一下 ``SSH'' 前面的 + 然后选择 ``Auth''，看右侧 ``Private key file for authentication:'' 下面的长条框里目前为空，点一下 ``Browse''，找到我们刚刚保存好的私钥，点''打开''。此时这个长条框里就有了私钥的地址，当然您也可以自行编辑这个路径。然后再回到左侧，点一下最上面的 ``Session''，在右侧再点一下 ``Save''.

7. 使用密钥验证登陆Linux

保存好后session， 点一下右下方的 ``Open''. 出现登陆界面，您会发现和原来的登陆提示内容有所不同了。

```
login as: root
Authenticating with public key "rsa-key-20130509"
Passphrase for key "rsa-key-20130509":
Last login: Thu May  9 16:17:13 2013 from 10.72.137.43
[root@localhost ~]#
```

现在不再输入root密码，而是需要输入密钥的密码，如果您先前在生产密钥的时候没有设置密码，您输入root后会直接登陆系统。

第六章扩展学习: http://www.lishiming.net/thread-5401-1-1.html

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

SEVEN

第七章 LINUX文件与目录管理

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

本章主要介绍Linux系统里文本和目录的相关操作。有一句话: `在Linux里一切皆文件', 是呀文件是Linux的基石，小到一个配置文件，大到一块磁盘都是用文件来控制的。这一部分内容比较基础，有较多基础命令出现，希望您多练习，多使用。用的多了自然而然就熟练了。

8.1 绝对路径和相对路径

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

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

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

命令：cd

这个命令是用来变更用户所在目录的，后面如果什么都不跟，就会直接到当前用户的根目录下，我们做实验用的是 `root' 账户，所以运行 cd 后，会进入root账户的根目录 `/root'. 后面跟目录名，则会直接切换到指定目录下：

```
[root@localhost ~]# cd /tmp/
[root@localhost tmp]# pwd
/tmp
[root@localhost tmp]# cd
[root@localhost ~]# pwd
/root
```

pwd 这个命令打印出当前所在目录，cd 后面只能是目录名，而不能是文件名，如果跟了文件名会报错：

```
[root@localhost ~]# cd /etc/passwd
-bash: cd: /etc/passwd: 不是目录
```

./ 表示当前目录，../ 表示当前目录的上一级目录：

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

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

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

命令：mkdir

用来创建目录的，这个命令在上一章节中用到过。 `mkdir' 其实就是make directory的缩写。其语法为 mkdir [-mp] [目录名称] ，其中-m, -p为其选项， `-m' 这个选项用来指定要创建目录的权限，不常用，阿铭不做重点解释。 `-p' 这个选项很管用，先来做个试验，您会一目了然的：

```
[root@localhost ~]# mkdir /tmp/test/123
mkdir: 无法创建目录 '/tmp/test/123': 没有那个文件或目录
[root@localhost ~]# mkdir -p /tmp/test/123
[root@localhost ~]# ls /tmp/test
123
```

当我们想创建 /tmp/test/123 目录，可是提示不能创建，原因是 /tmp/test 目录不存在，您会说，这个Linux怎么这样傻，/tmp/test 目录不存在就自动创建不就OK了嘛，的确Linux确实很傻，如果它发现要创建的目录的上一级目录不存在就会报错。然而Linux并不是那么傻，因为它也为我们想好了解决办法，即 `-p' 选项，这个选项可以帮我们创建一大串级联目录，这个选项还有一个好处，那就是当您创建一个已经存在的目录时，不会报错：

```
[root@localhost ~]# ls -ld /tmp/test/123
drwxr-xr-x. 2 root root 4096 5月   9 19:10 /tmp/test/123
[root@localhost ~]# mkdir /tmp/test/123
mkdir: 无法创建目录 '/tmp/test/123': 文件已存在
[root@localhost ~]# mkdir -p /tmp/test/123
[root@localhost ~]# ls -ld /tmp/test/123
drwxr-xr-x. 2 root root 4096 5月   9 19:10 /tmp/test/123
```

在上一章节里，阿铭已经介绍过 ls 命令，但是并没有向您介绍它的 `-d' 选项，这个选项是针对目录的，通常都是和 `-l' 同时使用写成 `-ld'. 它可以查看指定目录的属性，比如在本例中，它可以查看 `/tmp/test/123' 目录的创建时间。 mkdir -p 后面跟一个已经存在的目录名时，它不会做任何事情，只是不报错而已。

命令：rmdir

用来删除空目录，后面可以是一个也可以是多少，多个的话用空格分隔。该命令阿铭很少使用，因为它只能删除目录，不能删除文件，还有一个命令 rm 既可以删除目录又可以删除文件，阿铭用的比较多。rmdir 有一个和mkdir一样的选项 `-p', 同样可以级联删除一大串目录，但是级联的目录中其中一个目录里还有目录或者文件时就不好用了。

```
[root@localhost ~]# ls /tmp/test
123
[root@localhost ~]# rmdir /tmp/test/
rmdir: 删除 '/tmp/test/' 失败: 目录非空
[root@localhost ~]# rmdir /tmp/test/123
[root@localhost ~]# ls /tmp/test
[root@localhost ~]#
```

所以，得出的结论是， `rmdir' 只能删除空目录，即使加上 `-p' 选项也只能删除一串的空目录，可见这个命令有很大的局限性，偶尔用下还可以。

命令：rm

这个命令是最常用的， `rm' 同样也有很多选项。您可以通过 man rm 来获得详细帮助信息。在这里阿铭只介绍最常用的两个选项。

`-r'：删除目录用的选项，等同于rmdir.

```
[root@localhost ~]# mkdir -p /tmp/test/123
[root@localhost ~]# rm -r /tmp/test/123
rm：是否删除目录 '/tmp/test/123'? y
```

但是和rmdir不同的是，使用 rm -r 删除目录时，会问一下是否删除，如果输入 `y' 则会删除，输入 `n' 则不删除。当然 rm -r 也不会向rmdir不能删除非空目录，它是可以删除非空目录的。

`-f'：表示强制删除，不再提示是否要删除，而是直接就删除了，而后面跟一个不存在的文件或者目录时，也不会报错，如果不加 `-f' 选项会报错。

```
[root@localhost ~]# rm /tmp/test/123/123
rm: 无法删除 '/tmp/test/123/123': 没有那个文件或目录
[root@localhost ~]# rm -f /tmp/test/123/123
```

要删除一个目录时，即使加上 `-f' 选项也会报错，所以删除目录一定要加 `-r' 选项。

```
[root@localhost ~]# rm -f /tmp/test/123
rm: 无法删除 '/tmp/test/123': 是一个目录
[root@localhost ~]# rm -rf /tmp/test/123
```

关于rm，阿铭使用最多便是 `-rf' 两个选项合用了。不管删除文件还是目录都可以。但是方便的同时也要多注意，万一您的手太快后边跟了/那样就会把您的系统文件全部删除的，切记切记。

8.2 环境变量PATH

在讲环境变量之前阿铭先介绍一个命令 which, 它用来查找某个命令的绝对路径。

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

`rm' 和 `ls' 是两个特殊的命令，使用 alias 命令做了别名。我们用的 `rm' 实际上是 `rm -i' 加上 `-i' 选项后，删除文件或者命令时都会问一下是否确定要删除，这样做比较安全。 `alias' 可以设置命令的别名也可以设置文件的别名，阿铭会在以后章节中详细介绍。 `which' 这个命令阿铭平时只用来查询某个命令的绝对路径，不常使用。

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

```
[root@localhost ~]# echo $PATH
/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin
```

因为/bin 在PATH的设定中，所以自然就可以找到ls了。如果您将 ls 移动到 /root 底下的话，然后您自己本身也在 /root 底下，但是当您执行 ls 的时候，他就是不理您？怎么办？这是因为 PATH 没有 /root 这个目录，而您又将 ls 移动到 /root 底下了，自然系统就找不到可执行文件了，因此就会告诉您 `command not found!'

```
[root@localhost ~]# mv /bin/ls /root/
[root@localhost ~]# ls
-bash: /bin/ls: 没有那个文件或目录
```

`mv' 用来移动目录或者文件，还可以重命名，稍后讲解。

那么该怎么克服这种问题呢？有两个方法，一种方法是直接将 /root 的路径加入 $PATH 当中！如何增加？可以使用命令 PATH=$PATH:/root:

```
[root@localhost ~]# PATH=$PATH:/root
[root@localhost ~]# echo $PATH
/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin:/root
[root@localhost ~]# ls
anaconda-ks.cfg  install.log  install.log.syslog  ls
```

另一种方法就是使用绝对路径：

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

命令: cp

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

-r：如果您要拷贝一个目录，必须要加-r选项，否则您是拷贝不了目录的, 和 `rm' 类似。

```
[root@localhost ~]# mkdir 123
[root@localhost ~]# cp 123 456
cp: 略过目录'123'
[root@localhost ~]# cp -r 123 456
[root@localhost ~]# ls -l
总用量 44
drwxr-xr-x. 2 root root  4096 5月  10 04:05 123
drwxr-xr-x. 2 root root  4096 5月  10 04:05 456
```

-i：安全选项，和 `rm' 类似，如果遇到一个存在的文件，会问是否覆盖。在Redhat/CentOS系统中，我们使用的cp其实是cp -i

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

下面简单做一个小试验，很快您就会明白-i 选项的作用了。

```
[root@localhost ~]# cd 123
[root@localhost 123]# ls
[root@localhost 123]# touch 111
[root@localhost 123]# touch 222
[root@localhost 123]# cp -i 111 222
cp：是否覆盖 '222'？ n
[root@localhost 123]# echo 'abc' > 111
[root@localhost 123]# echo 'def' > 222
[root@localhost 123]# cat 111 222
abc
def
[root@localhost 123]# /bin/cp 111 222
[root@localhost 123]# cat 111
abc
```

```
[root@localhost 123]# cat 222
abc
```

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

命令: mv

`mv' 是move的简写。格式为 mv [选项] [源文件] [目标文件] 下面介绍几个常用的选项。

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

该命令有几种情况：

1） 目标文件是目录，而且目标文件不存在；

2） 目标文件是目录，而且目标文件存在；

3） 目标文件不是目录不存在；

4） 目标文件不是目录存在；

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

```
[root@localhost ~]# mkdir dira dirb
[root@localhost ~]# ls
anaconda-ks.cfg  dira  dirb  install.log  install.log.syslog
[root@localhost ~]# mv dira dirc
[root@localhost ~]# ls
anaconda-ks.cfg  dirb  dirc  install.log  install.log.syslog
```

目标文件为目录，并且目标目录不存在，相当于把 `dira' 重命名为 `dirc'.

```
[root@localhost ~]# mv dirc dirb
[root@localhost ~]# ls
anaconda-ks.cfg  dirb  install.log  install.log.syslog
[root@localhost ~]# ls dirb
dirc
```

目标文件为目录，且目标目录存在，则会把 `dirc' 移动到 `dirb' 目录里。

```
[root@localhost ~]# touch filed
[root@localhost ~]# ls
anaconda-ks.cfg  dirb  filed  install.log  install.log.syslog
[root@localhost ~]# mv filed filee
[root@localhost ~]# ls
anaconda-ks.cfg  dirb  filee  install.log  install.log.syslog
[root@localhost ~]# mv filee dirb
[root@localhost ~]# ls
anaconda-ks.cfg  dirb  install.log  install.log.syslog
[root@localhost ~]# ls dirb
dirc  filee
```

目标文件不是目录，且不存在，则会重命名文件。

8.3 几个和文档相关的命令

命令: cat

比较常用的一个命令，即查看一个文件的内容并显示在屏幕上，后面可以不加任何选项直接跟文件名，阿铭介绍两个常用的选项：

-n：查看文件时，把行号也显示到屏幕上。

```
[root@localhost ~]# echo  '111111111' > dirb/filee
[root@localhost ~]# echo  '222222222' >> dirb/filee
[root@localhost ~]# cat dirb/filee
111111111
222222222
[root@localhost ~]# cat -n dirb/filee
     1  111111111
     2  222222222
```

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

-A：显示所有东西出来，包括特殊字符

```
[root@localhost ~]# cat -A dirb/filee
111111111$
222222222$
```

命令: tac

和 `cat' 一样，用来把文件的内容显示在屏幕上，只不过是先显示最后一行，然后是倒数第二行，最后显示的是第一行。

```
 [root@localhost ~]# tac dirb/filee
 222222222
 111111111
```

命令: more

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

命令: less

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

命令: head

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

```
[root@localhost ~]# head /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
```

```
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
[root@localhost ~]# head -n 1 /etc/passwd
root:x:0:0:root:/root:/bin/bash
[root@localhost ~]# head -n2 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
```

`-n'后可以有空格也可以无空格。

命令: tail

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

```
[root@localhost ~]# head -n2 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
[root@localhost ~]# tail /etc/passwd
nobody:x:99:99:Nobody:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
saslauth:x:499:76:'Saslauthd user':/var/empty/saslauth:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
abrt:x:173:173::/etc/abrt:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
tcpdump:x:72:72::/:/sbin/nologin
[root@localhost ~]# tail -n2 /etc/passwd
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
tcpdump:x:72:72::/:/sbin/nologin
```

-f：动态显示文件的最后十行，如果文件是不断增加的，则用-f 选项。如：`tail -f /var/log/messages` 该选项特别特别常用，请熟记。

8.4 文件的所属主以及所属组

一个linux目录或者文件，都会有一个所属主和所属组。所属主，即文件的拥有者，而所属组，即该文件所属主所在的一个组。Linux这样设置文件属性的目的是为了文件的安全。例如，test文件的所属主是user0 而test1文件的所属主是user1，那么user1是不能查看test文件的，相应的user0也不能查看test1文件。有时我们也会有这样的需求，让一个文件同时让user0和user1来查看，这怎么实现呢？

这时 `所属组' 就派上用场了。即，创建一个群组users，让user0和user1同属于users组，然后建立一个文件test2，且其所属组为users，那么user0和user1都可以访问test2文件。Linux文件属性不仅规定了所属主和所属组，还规定了所属主（user）、所属组(group)以及其他用户（others）对该文件的权限。您可以通过ls -l 来查看这些属性。

```
[root@localhost ~]# ls -l
总用量 40
-rw-------. 1 root root   980 5月   7 18:00 anaconda-ks.cfg
drwxr-xr-x. 3 root root  4096 5月  10 05:10 dirb
```

8.5 linux文件属性

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

第1列，包含的东西有该文件类型和所属主、所属组以及其他用户对该文件的权限。第一列共11位有的文件是10位，没有最后面的一位。 其中第一位用来描述该文件的类型。上例中，我们看到的类型有 `d', `-` ，其实除了这两种外还有 `l', `b', `c', `s' 等。

`d' 表示该文件为目录；

`-` 表示该文件为普通文件；

`l' 表示该文件为链接文件（linux file），上边提到的软链接即为该类型；

```
[root@localhost ~]# ls -l /etc/rc.local
lrwxrwxrwx. 1 root root 13 5月   7 17:54 /etc/rc.local -> rc.d/rc.local
```

上例中，第一列第一位是 `l' 表示该文件为链接文件，稍后阿铭详细介绍。

`b' 表示该文件为块设备，比如 /dev/sda 就是这样的文件。

`c' 表示该文件为串行端口设备，例如键盘、鼠标。

`s' 表示该文件为套接字文件（socket），用于进程间通信。

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

一个文件的属性为 `-rwxr-xr--.' ，它代表的意思是，该文件为普通文件，文件拥有者可读可写可执行，文件所属组对其可读不可写可执行，其他用户对其只可读。对于一个目录来讲，打开这个目录即为执行这个目录，所以任何一个目录必须要有x权限才能打开并查看该目录。例如一个目录的属性为 `drwxr--r--.' 其所属主为root，那么除了root外的其他用户是不能打开这个目录的。

关于第一列的最后一位的 .，阿铭要说一下，之前的CentOS 5 是没有这个点的，这主要是因为新版本的ls把selinux或者acl的属性加进来了，当文件或者目录只使用了selinux context的属性，这里是一个点。如果设置了acl，后面将是一个加号 `+'. 关于selinux 和 acl 阿铭不再详细介绍，您只要了解是怎么回事即可，如果您感兴趣可以网上搜索相关知识也可以通过阿铭在本章后面提供的扩展学习链接进行学习。

第2列，表示为链接占用的节点(inode), [1] 为目录时，通常与该目录底下还有多少目录有关系。

第3列，表示该文件的所属主。

第4列，表示该文件的所属组。

第5列，表示该文件的大小。

第6列、第7列和第8列为该文件的最近的修改日期，分别为月份日期以及时间，也就是所谓的mtime.

第9列，文件名。

8.6 更改文件的权限

1. 更改所属组 chgrp

语法：chgrp [组名] [文件名]

[1] inode 译成中文就是索引节点，它用来存放档案及目录的基本信息，包含时间信息、文档名、属主以及属组等。Inode是Unix操作系统中的一种数据结构，本质是结构体，inode是随文件系统创建时生成的，它的个数有限。在Linux下，可以通过 df -i 来查看各个分区的inode数量。

```
[root@localhost ~]# groupadd testgroup
[root@localhost ~]# touch test1
[root@localhost ~]# ls -l test1
-rw-r--r-- 1 root root 0 5月  10 08:41 test1
[root@localhost ~]# chgrp testgroup test1
[root@localhost ~]# ls -l test1
-rw-r--r-- 1 root testgroup 0 5月  10 08:41 test1
```

这里用到了 `groupadd' 命令，其含义为增加一个用户组。该命令在以后章节中做详细介绍，您只要知道它是用来增加用户组的即可。除了更改文件的所属组，还可以更改目录的所属组。

```
[root@localhost ~]# ls -l dirb/
总用量 8
drwxr-xr-x. 2 root root 4096 5月  10 05:08 dirc
-rw-r--r--. 1 root root   20 5月  10 05:37 filee
[root@localhost ~]# ls -ld dirb/
drwxr-xr-x. 3 root root 4096 5月  10 05:10 dirb/
[root@localhost ~]# chgrp testgroup dirb
[root@localhost ~]# ls -ld dirb/
drwxr-xr-x. 3 root testgroup 4096 5月  10 05:10 dirb/
[root@localhost ~]# ls -l dirb/
总用量 8
drwxr-xr-x. 2 root root 4096 5月  10 05:08 dirc
-rw-r--r--. 1 root root   20 5月  10 05:37 filee
```

`chgrp'命令也可以更改目录的所属组，但是只能更改目录本身，而目录下面的目录或者文件没有更改，要想级联更改子目录以及子文件，有个选项可以实现：

```
[root@localhost ~]# chgrp -R testgroup dirb
[root@localhost ~]# ls -l dirb
总用量 8
drwxr-xr-x. 2 root testgroup 4096 5月  10 05:08 dirc
-rw-r--r--. 1 root testgroup   20 5月  10 05:37 filee
```

`chgroup' 命令阿铭使用的不多，因为还有一个命令可以替代。

2. 更改文件的所属主 chown

语法： chown [-R] 账户名 文件名 chown [-R] 账户名：组名 文件名

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

```
[root@localhost ~]# mkdir test // 创建 'test' 目录
[root@localhost ~]# useradd user1 // 创建用户 'user1', 关于 'useradd' 命令会在后续章节介绍。
[root@localhost ~]# touch test/test2 // 在test目录下创建test2文件
[root@localhost ~]# chown user1 test
[root@localhost ~]# ls -l test
总用量 0
-rw-r--r-- 1 root root 0 5月  10 09:00 test2
[root@localhost ~]# ls -ld test   // test目录所属主已经由 'root' 改为 'user1'.
drwxr-xr-x 2 user1 root 4096 5月  10 09:00 test
[root@localhost ~]# ls -l test  // 但是test目录下的test2文件所属主依旧是 'root'.
总用量 0
-rw-r--r-- 1 root root 0 5月  10 09:00 test2
[root@localhost ~]# chown -R user1:testgroup test
[root@localhost ~]# ls -l test
总用量 0
-rw-r--r-- 1 user1 testgroup 0 5月  10 09:00 test2
```

`chown -R user1:testgroup' 把test目录以及目录下的文件都修改成所属主为user1, 所属组为testgroup.

3. 改变用户对文件的读写执行权限 chmod

在linux中为了方便更改这些权限，linux使用数字去代替rwx, 具体规则为 `r' 等于4, `w' 等于2, `x' 等于1, `-` 等于0. 举个例子: `-rwxrwx---` 用数字表示就是 `770', 具体是这样来的: `rwx' = 4+2+1=7; `rwx' = 4+2+1=7; `- - -` = 0+0+0=0.

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

`-R' 选项作用同chown，级联更改。

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

```
[root@localhost ~]# ls -ld test
drwxr-xr-x 2 user1 testgroup 4096 5月  10 09:00 test
[root@localhost ~]# ls -l test
总用量 0
-rw-r--r-- 1 user1 testgroup 0 5月  10 09:00 test2
[root@localhost ~]# chmod 750 test
[root@localhost ~]# ls -ld test
drwxr-x--- 2 user1 testgroup 4096 5月  10 09:00 test
[root@localhost ~]# ls -l test/test2
-rw-r--r-- 1 user1 testgroup 0 5月  10 09:00 test/test2
[root@localhost ~]# chmod 700 test/test2
[root@localhost ~]# chmod -R 700 test
[root@localhost ~]# ls -ld test
drwx------ 2 user1 testgroup 4096 5月  10 09:00 test
[root@localhost ~]# ls -l test
总用量 0
-rwx------ 1 user1 testgroup 0 5月  10 09:00 test2
```

如果您创建了一个目录，而该目录不想让其他人看到内容，则只需设置成 `rwxr-----` (740) 即可。'chmod' 还支持使用rwx的方式来设置权限。从之前的介绍中我们可以发现，基本上就九个属性分别是(1)user (2)group (3)others, 我们可以使用u, g, o 来代表它们三个的属性，此外， a 则代表 all 亦即全部。阿铭举例来介绍他们的用法：

```
[root@localhost ~]# chmod u=rwx,og=rx test/test2
[root@localhost ~]# ls -l test/test2
-rwxr-xr-x 1 user1 testgroup 0 5月  10 09:00 test/test2
```

这样可以把 `test/test2' 文件权限修改为 `rwxr-xr-x'. 另外还可以针对u, g, o, a增加或者减少某个权限（读，写，执行），例如：

```
[root@localhost ~]# chmod u-x test/test2
[root@localhost ~]# ls -l test
总用量 0
-rw-r-xr-x 1 user1 testgroup 0 5月  10 09:00 test2
[root@localhost ~]# chmod a-x test/test2
[root@localhost ~]# ls -l test/test2
-rw-r--r-- 1 user1 testgroup 0 5月  10 09:00 test/test2
[root@localhost ~]# chmod u+x test/test2
[root@localhost ~]# ls -l test/test2
-rwxr--r-- 1 user1 testgroup 0 5月  10 09:00 test/test2
```

命令: umask

上边也提到了默认情况下，目录权限值为755, 普通文件权限值为644, 那么这个值是由谁规定呢？追究其原因就涉及到了 `umask'.

umask语法： umask xxx （这里的xxx代表三个数字）

查看umask值只要输入 `umask' 然后回车。

```
[root@localhost ~]# umask
0022
```

umask预设是0022，其代表什么含义？先看一下下面的规则：

1）若用户建立为普通文件，则预设 `没有可执行权限', 只有'rw'两个权限。最大为666 (`-rw-rw-rw-`).

2）若用户建立为目录，则预设所有权限均开放，即777 (`drwxrwxrwx').

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

```
[root@localhost ~]# umask 002
[root@localhost ~]# mkdir test2
[root@localhost ~]# ls -ld test2
drwxrwxr-x 2 root root 4096 5月  10 09:44 test2
[root@localhost ~]# touch test3
[root@localhost ~]# ls -l test3
-rw-rw-r-- 1 root root 0 5月  10 09:45 test3
```

可以看到创建的目录权限默认变为775, 而文件默认权限变为664. 然后再把umask改回来。

```
[root@localhost ~]# umask 022
[root@localhost ~]# touch test4
[root@localhost ~]# ls -l test4
-rw-r--r-- 1 root root 0 5月  10 09:45 test4
```

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

4. 修改文件的特殊属性

命令：chattr

语法： chattr [+-=][ASaci [文件或者目录名]

`+-='：分别为增加、减少、设定

`A'：增加该属性后，文件或目录的atime将不可被修改；

`S'：增加该属性后，会将数据同步写入磁盘中；

`a'：增加该属性后，只能追加不能删除，非root用户不能设定该属性；

`c'：自动压缩该文件，读取时会自动解压；

`i'：增加后，使文件不能被删除、重命名、设定链接接、写入、新增数据；

```
[root@localhost ~]# chattr +i test2
[root@localhost ~]# touch test2/test1
touch: 无法创建'test2/test1': 权限不够
[root@localhost ~]# chattr -i test2
[root@localhost ~]# touch test2/test1
[root@localhost ~]# chattr +i test2
[root@localhost ~]# rm -f test2/test1
rm: 无法删除'test2/test1': 权限不够
```

对 `test2' 目录增加 `i' 权限后，即使是root账户也不能在 `test2' 里创建或删除test1文件。

```
[root@localhost ~]# chattr -i test2
[root@localhost ~]# touch test2/test3
[root@localhost ~]# ls test2
test1  test3
[root@localhost ~]# chattr +a test2
[root@localhost ~]# rm -f test2/test1
rm: 无法删除 'test2/test1': 不允许的操作
[root@localhost ~]# touch test2/test4
[root@localhost ~]# ls test2
test1  test3  test4
```

test2目录增加 `a' 权限后，只可以在里面创建文件，而不能删除文件。文件同样可以适用这些权限。

```
[root@localhost ~]# chattr +a test2/test1
[root@localhost ~]# echo '11111' > test2/test1
-bash: test2/test1: 不允许的操作
[root@localhost ~]# echo '11111' >> test2/test1
[root@localhost ~]# cat test2/test1
11111
[root@localhost ~]# chattr +i test2/test3
[root@localhost ~]# echo '11111' >> test2/test3
-bash: test2/test3: 权限不够
[root@localhost ~]# echo '11111' > test2/test3
-bash: test2/test3: 权限不够
[root@localhost ~]# rm -f test2/test3
rm: 无法删除'test2/test3': 权限不够
```

命令：lsattr

该命令用来读取文件或者目录的特殊权限，语法为 `lsattr [-aR] [文件/目录名]`

`-a'：类似与ls 的-a 选项，即连同隐藏文件一同列出；

`-R'：连同子目录的数据一同列出

```
[root@localhost ~]# lsattr test2
-----a-------e- test2/test1
----i--------e- test2/test3
-------------e- test2/test4
[root@localhost ~]# lsattr -aR test2
----i--------e- test2/.
-----a-------e- test2/test1
-------------e- test2/..
----i--------e- test2/test3
-------------e- test2/test4
```

8.7 在linux下搜一个文件

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

1. `which' 用来查找可执行文件的绝对路径。

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

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

语法： whereis [-bmsu] [文件名称]

`-b'：只找binary 文件

`-m'：只找在说明文件manual路径下的文件

`-s'：只找source来源文件

`-u'：没有说明档的文件

说明：whereis 阿铭几乎很少用到，如果您感兴趣请深入研究。

3. `locate' 类似于'whereis'，也是通过查找预先生成的文件列表库来告诉用户要查找的文件在哪里。

后边直接跟文件名。如果您的linux没有这个命令，请安装软件包 `mlocate'，这个软件包在您的系统安装盘里，后缀名是RPM，随后介绍的find命令会告诉您如何查找这个包。如果您装的CentOS您可以使用这个命令来安装 yum install -y mlocate 前提是您的CentOS能连网。至于yum这个命令如何使用，到后续章节您自然会明白。如果您刚装上这个命令，初次使用会报错。

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

这是因为系统还没有生成那个文件列表库。您可以使用 updatedb 命令立即生成（更新）这个库。如果您的服务器上正跑着重要的业务，那么您最好不要去运行这个命令，因为一旦运行，服务器的压力会变大。这个数据库默认情况下每周更新一次。所以您用locate命令去搜索一个文件，正好是在两次更新时间段内，那您肯定是得不到结果的。您可以到/etc/updated.conf 去配置这个数据库生成（更新）的规则。'locate'所搜索到的文件列表，不管是目录名还是文件名，只要包含我们要搜索的关键词，都会列出来，所以'locate'不适合精准搜索，这个命令阿铭使用的也并不多，您只要明白有这么一个工具即可，用到时再去深究其用法吧。

4. `find' 这个搜索工具是阿铭用的最多的一个，所以请您务必要熟悉它。

语法：find [路径] [参数] 下面介绍几个阿铭经常用的参数

`-atime +n/-n'：访问或执行时间大于/小于n天的文件

`-ctime +n/-n'：写入、更改inode属性（例如更改所有者、权限或者链接）时间大于/小于n天的文件

`-mtime +n/-n'：写入时间大于/小于n天的文件

```
[root@localhost ~]# find /tmp/ -mtime -1
/tmp/
/tmp/.ICE-unix
/tmp/test
[root@localhost ~]# find /tmp/ -atime +10
[root@localhost ~]# find /tmp/ -atime +1
/tmp/yum.log
/tmp/.bash_history
```

看到这里，您对这三个time是不是有些晕了，那阿铭就先给您介绍一下这三个time属性。

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

`stat' 命令可用来列出文件的 atime、ctime 和 mtime。

```
[root@localhost ~]# stat test/test2
  File: 'test/test2'
  Size: 0               Blocks: 0          IO Block: 4096   普通空文件
Device: 803h/2051d      Inode: 261657      Links: 1
```

```
Access: (0744/-rwxr--r--)  Uid: (  500/   user1)   Gid: (  500/testgroup)
Access: 2013-05-10 09:00:36.092000531 +0800
Modify: 2013-05-10 09:00:36.092000531 +0800
Change: 2013-05-10 09:30:58.788996594 +0800
```

atime不一定在访问文件之后被修改，因为：使用ext3文件系统的时候，如果在mount的时候使用了noatime参数那么就不会更新atime的信息。总之，這三個 time stamp 都放在 inode 中。若 mtime，atime 修改inode 就一定會改，既然 inode 改了，那 ctime 也就跟着要改了。

阿铭继续'find'常用选项：

`-name filename' 直接查找该文件名的文件，这个选项使用很多。

```
[root@localhost ~]# find . -name test2
./test/test2
./test2
```

`-type filetype' 通过文件类型查找。文件类型在前面部分已经简单介绍过，相信您已经大体上了解了。filetype 包含了 f, b, c, d, l, s 等。

```
[root@localhost ~]# find /tmp/ -type d
/tmp/
/tmp/.ICE-unix
/tmp/test
[root@localhost ~]# find /tmp/ -type f
/tmp/yum.log
/tmp/.bash_history
/tmp/ip.txt
```

8.8 linux的文件系统简介

搞计算机的应该都知道windows的系统格式化硬盘时会指定格式，fat 或者 ntfs。而linux的文件系统格式为Ext3，或者Ext4 。早期的linux使用Ext2格式，目前的linux都使用了Ext3, 而CentOS6已经使用了Ext4. Ext2文件系统虽然是高效稳定的。但是，随着Linux系统在关键业务中的应用，Linux文件系统的弱点也渐渐显露出来了，因为Ext2文件系统是非日志文件系统。这在关键行业的应用是一个致命的弱点。Ext3文件系统是直接从Ext2文件系统发展而来，Ext3文件系统带有日志功能，可以跟踪记录文件系统的变化，并将变化内容写入日志，写操作首先是对日志记录文件进行操作，若整个写操作由于某种原因 (如系统掉电) 而中断，系统重启时，会根据日志记录来恢复中断前的写操作，而且这个过程费时极短。目前Ext3文件系统已经非常稳定可靠。它完全兼容Ext2文件系统。用户可以平滑地过渡到一个日志功能健全的文件系统中来。这实际上了也是ext3日志文件系统初始设计的初衷。

而现在我们使用的Ext4文件系统，较Ext3文件系统又有很多好的特性，最明显的特征是，Ext4支持的最大文件系统容量和单个最大文件大小，详细的区别阿铭不再介绍，这需要您到网上搜一下以丰富您的知识点。

Linux文件系统在windows中是不能识别的，但是在linux系统中您可以挂载的windows的文件系统，linux目前支持MS-DOS，VFAT，FAT，BSD等格式，如果您使用的是Redhat或者CentOS，那么请不要妄图挂载NTFS格式的分区到linux下，因为它不支持NTFS。虽然有些版本的linux支持NTFS，但不建议使用，因为目前的技术还不成熟。但有时，的确会有这方面的需求，您可以安装ntfs-3g软件包来解决这个问题。

Ext3文件系统为早期版本Redhat/CentOS默认使用的文件系统，目前Ext4为默认文件系统，除了Ext3/Ext4文件系统外，有些linux发行版例如SuSE默认的文件系统为reiserFS, Ext3 独特的优点就是易于转换，很容易在 Ext2 和 Ext3 之间相互转换，而具有良好的兼容性，其它优点 ReiserFS 都有，而且还比它做得更好。如高效的磁盘空间利用和独特的搜寻方式都是Ext3 所不具备的，速度上它也不能和 ReiserFS相媲美，在实际使用过程中，reiserFS 也更加安全高效，据说反删除功能也不错。

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

8.9 linux文件类型

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

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

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

3）链接文件(link)：`ls -l` 查看第一个属性为 `l', 类似windows下的快捷方式。这种文件在linux中很常见，而且阿铭在日常的系统运维工作中用的很多，所以您要特意留意一下这种类型的文件。在后续章节阿铭会介绍。

4）设备(device)：与系统周边相关的一些档案，通常都集中在 /dev 这个目录之下! 通常又分为两种：块(block)设备 ：就是一些储存数据，以提供系统存取的接口设备，简单的说就是硬盘。例如您的一号硬盘的代码是 /dev/sda1, 第一个属性为 `b'；另一种是字符(character)设备 ：是一些串行端口的接口设备，例如键盘、鼠标等等，第一个属性为 `c'.

Linux文件后缀名

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

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

8.10 Linux的链接文件

上面阿铭多次提到链接文件的概念，相信您早就想明白到底什么是链接文件。下面阿铭就来介绍一下这个链接文件的知识点。链接文件分为两种，硬链接(hard link)和软链接(symbolic link)。两种链接的本质区别关键点在于inode.

Hard Links：当系统要读取一个文件时，就会先去读inode table，然后再去根据inode中的信息到块区域去将数据取出来。而hard link 是直接再建立一个inode链接到文件放置的块区域。也就是说，进行hard link 的时候实际上该文件内容没有任何变化，只是增加了一个指到这个文件的inode, hard link 有两个限制：(1) 不能跨文件系统，因为不同的文件系统有不同的inode table; (2) 不能链接目录。

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

么b就会把读取的动作转发到a上，这样就读取到了文件a。所以，当您删除文件a时，文件b并不会被删除，但是再读取b时，会提示无法打开文件。而，当您删除b时，a是不会有任何影响的。

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

命令: ln

语法：ln [-s] [来源文件] [目的文件]

ln 常用的选项就一个 `-s', 如果不加就是建立硬链接，加上就建立软链接。

```
[root@localhost ~]# mkdir 123
[root@localhost ~]# cd 123
[root@localhost 123]# cp /etc/passwd ./
[root@localhost 123]# ll
总用量 4
-rw-r--r-- 1 root root 1097 5月  10 17:08 passwd
[root@localhost 123]# du -sk
8       .
[root@localhost 123]# ln passwd passwd-hard
[root@localhost 123]# ll
总用量 8
-rw-r--r-- 2 root root 1097 5月  10 17:08 passwd
-rw-r--r-- 2 root root 1097 5月  10 17:08 passwd-hard
[root@localhost 123]# du -sk
8       .
```

上例中的 `ll' 命令等同于 `ls -l', 请使用 `which' 命令查看一下。做了硬链接后，虽然两个文件大小都为 `1097', 但是目录的大小并没有变化。

```
[root@localhost 123]# ll
总用量 4
-rw-r--r-- 1 root root 1097 5月  10 17:08 passwd-hard
[root@localhost 123]# rm -f passwd
[root@localhost 123]# du -sk
8       .
```

删除源文件passwd, 空间依旧不变。这说明硬链接只是复制了一份inode信息。

```
[root@localhost ~]# ln 123 456
ln: "123": 不允许将硬链接指向目录
```

硬链接不能用于目录。

```
[root@localhost ~]# mkdir 456
[root@localhost ~]# cd 456
[root@localhost 456]# cp /etc/passwd ./
[root@localhost 456]# ln -s passwd  passwd-soft
[root@localhost 456]# ll
总用量 4
-rw-r--r-- 1 root root 1097 5月  10 17:18 passwd
lrwxrwxrwx 1 root root    6 5月  10 17:19 passwd-soft -> passwd

[root@localhost 456]# head -n1 passwd-soft
root:x:0:0:root:/root:/bin/bash
[root@localhost 456]# head -n1 passwd
root:x:0:0:root:/root:/bin/bash
[root@localhost 456]# rm -f passwd
```

```
[root@localhost 456]# head -n1 passwd-soft
head: 无法打开"passwd-soft" 读取数据: 没有那个文件或目录
[root@localhost 456]# ll
总用量 0
lrwxrwxrwx 1 root root 6 5月  10 17:19 passwd-soft -> passwd
```

如果删除掉源文件，则软链接文件不能读取了，而且使用 `ll' 查看发现颜色也变了。

```
[root@localhost ~]# ln -s 456 789
[root@localhost ~]# ls -ld 456 789
drwxr-xr-x 2 root root 4096 5月  10 17:22 456
lrwxrwxrwx 1 root root    3 5月  10 17:29 789 -> 456
```

目录是可以软链接的。

知识面扩展学习：http://www.lishiming.net/thread-5406-1-1.html

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

EIGHT

第八章 LINUX系统用户及用户组管理

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

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

9.1 认识/etc/passwd和/etc/shadow

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

```
[root@localhost ~]# cat /etc/passwd | head
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
```

您是不是对上面的命令有点不知所以，''head'' 前面的 ``|'' 我们叫做管道符，它的作用是把前面的命令的输出再输入给后面的命令。管道符会在后面章节中提及，这个符号用的也是蛮多的，请掌握它的用法。

`/etc/passwd' 由 `:' 分割成7个字段，每个字段的具体含义是:

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

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

3）这个数字代表用户标识号，也叫做uid。系统识别用户身份就是通过这个数字来的，0就是root，也就是说您可以修改test用户的uid为0，那么系统会认为root和test为同一个账户。通常uid的取值范围是0~65535(但实际上已经可以支持到4294967294)，0是超级用户（root）的标识号，1~499由系统保留，作为管理账号，普通用户的标识号从500开始，如果我们自定义建立一个普通用户，您会看到该账户的标识号是大于或等于500的。

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

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

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

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

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

```
[root@localhost ~]# cat /etc/shadow |head -n 3
root:$6$Wo0kPkgm$OAp0Wl2AsaE4ei4YVbxo3DIU5OBSYxn1y7qxB5Jns70Yk91AvzElsR5GmoGCC8DUXkKzK7vyiV8wXNeaWNm861:15832:0:99999:7:
bin:*:15628:0:99999:7:::
daemon:*:15628:0:99999:7:::
```

每个字段的含义是：

1）用户名，跟/etc/passwd对应。

2）用户密码，这个才是该账号的真正的密码，不过这个密码已经加密过了，但是有些黑客还是能够解密的。所以，该文件属性设置为000，但是root账户是可以访问或更改的。

```
[root@localhost ~]# ls -l /etc/shadow
---------- 1 root root 719 5月  10 09:02 /etc/shadow
```

3）上次更改密码的日期，这个数字是这样计算得来的，距离1970年1月1日到上次更改密码的日期，例如上次更改密码的日期为2012年1月1日，则这个值就是 `365 x (2012-1970) + (2012-1970)/4 + 1 = 15341'. 因为如果是闰年，则有366天。

4）要过多少天才可以更改密码，默认是0，即不限制。

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

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

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

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

9）作为保留用的，没有什么意义。

9.2 新增/删除用户和用户组

1. 新增一个组

   命令：groupadd

   语法：groupadd [-g GID] groupname

```
[root@localhost ~]# groupadd grptest1
[root@localhost ~]# tail -n1 /etc/group
grptest1:x:502:
```

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

```
[root@localhost ~]# groupadd -g 511 grptest2
[root@localhost ~]# tail -n2 /etc/group
grptest1:x:502:
grptest2:x:511:
```

``-g'' 选项可以自定义gid.

2. 删除组

命令：groupdel

```
[root@localhost ~]# groupdel grptest2
[root@localhost ~]# tail -n3 /etc/group
testgroup:x:500:
user1:x:501:
grptest1:x:502:
```

该命令没有特殊选项，但有一种情况不能删除组：

```
[root@localhost ~]# groupdel user1
groupdel: cannot remove the primary group of user 'user1'
```

这是因为user1组中包含user1账户，只有删除user1账户后才可以删除该组。

3. 增加账户

命令：useradd

语法：useradd [-u UID] [-g GID] [-d HOME] [-M] [-s]

`-u' 自定义UID

`-g' 使其属于已经存在的某个组，后面可以跟组id, 也可以跟组名

`-d' 自定义用户的家目录

`-M' 不建立家目录

`-s' 自定义shell

```
[root@localhost ~]# useradd test10
[root@localhost ~]# tail -n1 /etc/passwd
test10:x:500:503::/home/test10:/bin/bash
[root@localhost ~]# tail -n1 /etc/group
test10:x:503:
```

`useradd' 不加任何选项直接跟用户名，则会创建一个跟用户名同样名字的组。

```
[root@localhost ~]# useradd -u510 -g 513 -M -s /sbin/nologin user11
useradd: group '513' does not exist
[root@localhost ~]# useradd -u510 -g 502 -M -s /sbin/nologin user11
[root@localhost ~]# useradd -u511 -g grptest1 user12
[root@localhost ~]# tail -n2 /etc/passwd
user11:x:510:502::/home/user11:/sbin/nologin
user12:x:511:502::/home/user12:/bin/bash
[root@localhost ~]# tail -n2 /etc/group
grptest1:x:502:
test10:x:503:
```

`-g' 选项后面跟一个不存在的gid会报错，提示该组不存在。刚刚上面说过 `-M' 选项加上后则不建立用户家目录，但是在/etc/passwd文件中仍然有这个字段。但是您使用 `ls /home/user11` 查看一下会提示该目录不存在。所以 `-M' 选项的作用只是不创建那个目录。

```
[root@localhost ~]# ls /home/user11
ls: 无法访问/home/user11: 没有那个文件或目录
```

4. 删除账户

   命令：userdel

   语法：userdel [-r] username

```
[root@localhost ~]# ls -ld /home/user12
drwx------ 3 user12 grptest1 4096 5月  11 07:12 /home/user12
[root@localhost ~]# userdel user12
[root@localhost ~]# ls -ld /home/user12
drwx------ 3 511 grptest1 4096 5月  11 07:12 /home/user12
[root@localhost ~]# ls -ld /home/test10/
drwx------ 3 test10 test10 4096 5月  11 07:09 /home/test10/
[root@localhost ~]# userdel  -r test10
[root@localhost ~]# ls -ld /home/test10/
ls: 无法访问/home/test10/: 没有那个文件或目录
```

`-r' 选项的作用只有一个，就是删除账户的时候连带账户的家目录一起删除。

9.3 chfn 更改用户的finger （不常用）

阿铭几乎没有用过这个功能，只简单介绍一下即可，而您也许了解一下即可。前面内容中提到了findger，即在/etc/passwd文件中的第5个字段中所显示的信息，那么如何去设定这个信息呢？

```
[root@localhost ~]# chfn user11
Changing finger information for user11.
Name []: user11
Office []: user11's office
Office Phone []: 12345678
Home Phone []: 123456789

Finger information changed.
[root@localhost ~]# grep 'user11' /etc/passwd
user11:x:510:502:user11,user11's office,12345678,123456789:/home/user11:/sbin/nologin
```

`chfn' 命令可以修改用户的findger信息，比如name, office, office phone 以及 Home phone.修改完后，就会在/etc/passwd文件中的user11的那一行第五个字段中看到相关信息了，默认是空的。 在本例中，阿铭使用了 ``grep'' 命令，它是用来过滤指定关键词的行，阿铭会在以后的章节中详细介绍它的用法。

9.4 创建/修改一个用户的密码

命令：passwd

语法：passwd [username]

等创建完账户后，默认是没有设置密码的，虽然没有密码，但该账户同样登录不了系统。只有设置好密码后方可登录系统。为用户创建密码时，为了安全起见，请尽量设置复杂一些。您可以按照这样的规则来设置密码：

1. 长度大于10个字符；
2. 密码中包含大小写字母数字以及特殊字符 `*', `&', `%' 等；

3. 不规则性（不要出现root, happy, love, linux, 7758520, 111111等等单词或者数字）；
4. 不要带有自己名字、公司名字、自己电话、自己生日等。

```
[root@localhost ~]# passwd
更改用户 root 的密码 。
新的 密码：
重新输入新的 密码：
passwd： 所有的身份验证令牌已经成功更新。
```

``passwd'' 后面不加username则是修改当前账户的密码。如果您登陆的是root账户，后面可以跟普通账户的名字，意思是修改指定账户的密码。

```
[root@localhost ~]# passwd user11
更改用户 user11 的密码 。
新的 密码：
重新输入新的 密码：
passwd： 所有的身份验证令牌已经成功更新。
```

只有root才可以修该其他账户的密码，普通账户只能修改自己的密码，其他账户的密码是不可以修改的。

命令：mkpasswd

这个命令阿铭经常用来生成密码，省的自己去想。默认您的Linux是没有这个命令的，需要安装一个包``expect'', 如果您的CentOS可以上网，请使用命令 `yum install -y expect` 即可完成安装。安装好后，输入命令：

```
[root@localhost ~]# mkpasswd
HXut8oy*8
```

生成的随机字符串就可以作为一个密码，只不过这个密码不容易记忆，没有关系，阿铭等会介绍一个小工具来帮您记录密码，而且很安全。

9.5 用户身份切换

Linux系统中，有时候普通用户有些事情是不能做的，除非是root用户才能做到。这时就需要临时切换到root身份来做事了。下面阿铭带您做一个小实验，创建 ``test'' 账户，并修改其密码，这样我们就可以使用test账户登陆Linux了。

```
[root@localhost ~]# useradd test
[root@localhost ~]# passwd test
更改用户 test 的密码 。
新的 密码：
重新输入新的 密码：
passwd： 所有的身份验证令牌已经成功更新。
```

然后用test账户登陆Linux.

```
login as: test
test@10.72.137.78's password:
[test@localhost ~]$ whoami
test
```

登陆后，使用 ``whoami'' 命令可以查看当前用户是谁。

命令su

语法：`su [-] username`

后面可以跟 `-` 也可以不跟，普通用户su不加username时就是切换到root用户，当然root用户同样可以su到普通用户。 `-` 这个字符的作用是，加上后会初始化当前用户的各种环境变量，关于环境变量这部分内容阿铭放在后面的章节中讲解。 下面阿铭做个简单的实验来说明加与不加 `-` 的区别：

```
[test@localhost ~]$ pwd
/home/test
[test@localhost ~]$ su
密码：
[root@localhost test]# pwd
/home/test
[root@localhost test]# exit
exit
[test@localhost ~]$ su -
密码：
[root@localhost ~]# pwd
/root
```

如果不加 `-` 切换到root账户下时，当前目录没有变化，而加上 `-` 切换到root账户后，当前目录为root账户的家目录，这跟直接登陆root账户是一样的。当用root切换普通用户时，是不需要输入密码的。这也体现了root用户至高无上的权利。

命令：sudo

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

默认root能够sudo是因为这个文件中有一行 ``root ALL=(ALL) ALL'' 在该行下面加入 ``test ALL=(ALL) ALL'' 就可以让test用户拥有了sudo的权利。使用 ``visudo'' 命令编辑/etc/sudoers配置文件，其实它的操作方法和前面阿铭介绍的 ``vi'' 命令使用方法是一样的，按 `i' 进入编辑模式，编辑完成后，按 ``Esc'' ，再输入 '':wq'' 完成保存。

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

此时可以验证一下test账户的权限了。

```
[root@localhost ~]# su test
[test@localhost root]$ ls
ls: 无法打开目录.: 权限不够
[test@localhost root]$ sudo ls

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for test:
123  456  789  anaconda-ks.cfg  dirb  install.log  install.log.syslog  test  test1  test2  test3
```

由于切换到test账户后的当前目录依旧是在/root 下，test账户没有任何权限，所以 `ls' 的时候提示说权限不够，然而使用 sudo ls 输入test账户自身的密码后就有权限了。初次使用sudo 时会有上面的一大段提示，而后再次使用sudo 命令则不再提示。

如果每增加一用户就设置一行，这样太麻>烦了。所以您可以这样设置。把 ``# %wheel ALL=(ALL) ALL''

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

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

配置文件/etc/sudoers包含了诸多配置项，可以使用命令 man sudoers 来获得帮助信息。下面阿铭介绍一个很实用的案例，我们的需求是把Linux服务器设置成这个样子：只允许使用普通账户登陆，而普通账户登录后，可以不输入密码就能sudo切换到root账户。下面但是阿铭的配置：

```
[root@localhost ~]# visudo
```

然后在文件的最后面加入三行：

```
User_Alias USER_SU = test, test1, aming
Cmnd_Alias SU = /bin/su
USER_SU ALL=(ALL) NOPASSWD: SU
```

保存配置文件后，使用test, test1, aming 三个账户登陆Linux后，执行命令 sudo su - 切换到root账户，获取root账户的所有权利。

```
[root@localhost ~]# su - test
[test@localhost ~]$ sudo su -
[root@localhost ~]# whoami
root
```

而不让root直接登陆，这个简单，设置一个非常复杂连自己都记不住的密码。不过这样也有一个问题，就是普通用户可以su到root，然后他再自己修改简单的密码就能直接root登陆了不是嘛？的确有这个问题，其实阿铭还有一个更好的办法，会在后面的扩展学习章节中介绍。

9.6 使用密码记录工具keepass来保存密码

在第三章，阿铭曾经给过您建议，密码不要保存在文档中，那样不安全，如果密码很多而且又很复杂，人的大脑是不可能很容易记住的，只能记录下来，如果不能记在文档中那记在哪里呢？阿铭介绍给您一款记录密码的软件，是在windows上用的哦！它就是keepass.

Keepass官网地址是: http://www.keepass.info 在官网keepass是这样被形容的: The free, open source, light-weight and easy-to-use password manager. 没错，这款软件是免费的、开源的、容易使用轻量级的密码管理工具。

我们下载最新版本的keepass，当前最新的发行版本为2.22, 下载地址：http:// downloads.sourceforge.net/keepass/KeePass-2.22-Setup.exe 下载后安装它。安装过程没有什么可说的，直接next一直到安装完成。

1. 安装好后，运行keepass，首先创建一个新的密码库文件。

点菜单栏 ``file'' 然后选择 ``new''，为密码库文件找一个安全的地方存放。接下来，为我们的密码文件创建一个 ``master password''，这个密码以后每次我们查看密码的时候都需要输入，输入正确后才可以查看，这样的设计也是为了安全。设置好密码后连续点两个 ``ok'' 完成创建密码库文件。

2. 增加一个group

鼠标选中左侧的 ``NewDatebase'', 点右键选择 ``Add Group'', 单击后创建新组，然后更改组名，比说叫做 ``test''. 当然组下面还可以创建组，方法一样的。

3. 创建一个Entry

鼠标左键先点一下刚才创建的组 ``test'' ， 然后在右侧空白处右键单击，选择 ``Add Entry''. 弹出一个会话窗口，Title 自定义，方便我们以后查找; User name 用来记录密码的用户是谁; Password 这个默认就存

在了，也可以更改，点一下后面的 ''...'' 图标可以查看密码的内容，再点一下变为不可见状态; URL 用来记录网址，方便我们跳转，比如这个密码为某个网站的某个会员的密码，那如果在这里填写了该网址地址，则可以直接跳转到那个网站，可以留空；Notes 用来写一些与这个密码相关的信息，方便我们记忆，可以留空。

4. 修改Entry信息

在右侧窗口，选中要修改的Entry那行，鼠标移动到Title区域，双击则会跳出一个会话窗口，我们可以更改Entry的各项信息。

5. 获取Entry密码

同样是在右侧窗口，选中要获取的密码那行，鼠标移动到Password区域，直接双击，就把密码复制到剪切板了，密码会在剪切板中保存12s，过期会失效，所以您应该在12s内把密码粘贴。

阿铭建议您最好再扩展学习一下: http://www.lishiming.net/thread-5409-1-1.html

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

NINE

第九章 LINUX磁盘管理

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

在日常的Linux管理工作中，这部分内容使用还是比较多的。

10.1 查看磁盘或者目录的容量

命令：df

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

```
[root@localhost ~]# df
文件系统                1K-块      已用      可用 已用% 挂载点
/dev/sda3             14347632   1490876  12127924  11% /
tmpfs                   163308         0    163308   0% /dev/shm
/dev/sda1                99150     26808     67222  29% /boot
```

``df'' 常用选项有 ``-i'' ``-h'' ``-k'' ``-m''等

``-i'' 查看inodes使用状况

```
文件系统              Inode  已用(I)  可用(I) 已用(I)%% 挂载点
/dev/sda3             912128   66195  845933    8% /
tmpfs                  40827       1   40826    1% /dev/shm
/dev/sda1              25688      38   25650    1% /boot
```

``-h'' 使用合适的单位显示，例如 `G'

```
文件系统              容量  已用  可用 已用%% 挂载点
/dev/sda3              14G  1.5G   12G  11% /
tmpfs                 160M     0  160M   0% /dev/shm
/dev/sda1              97M   27M   66M  29% /boot
```

``-k'', ``-m'' 分别以K, M 为单位显示

```
[root@localhost ~]# df -k
文件系统                1K-块      已用      可用 已用% 挂载点
/dev/sda3             14347632   1490880  12127920  11% /
tmpfs                   163308         0    163308   0% /dev/shm
/dev/sda1                99150     26808     67222  29% /boot
[root@localhost ~]# df -m
文件系统                1M-块      已用      可用 已用% 挂载点
/dev/sda3                14012      1456     11844  11% /
```

```
tmpfs                     160          0       160   0% /dev/shm
/dev/sda1                  97         27        66  29% /boot
```

简单介绍一下各列所表示的含义，其实如果您的Linux和阿铭的虚拟机一样也是中文显示的话，那么不用说太多，看字面意思就明白了。第一列是分区的名字，第二列为该分区总共的容量，第三列为已经使用了多少，第四列为还剩下多少，第五列为已经使用百分比，如果这个数值到达90%以上，那么您就应该关注了，磁盘分区满了可不是什么好事情，会引起系统崩溃的。最后一列为挂载点，您是否还记得，阿铭在装系统的时候，有说到这个词，''/dev/shm'' 为内存挂载点，如果您想把文件放到内存里，就可以放到/dev/shm/目录下。

命令：du

``du'' 用来查看某个目录或文件所占空间大小.

语法：du [-abckmsh] [文件或者目录名] 常用的参数有：

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

```
[root@localhost ~]# du dirb
4       dirb/dirc
12      dirb
[root@localhost ~]# du -a dirb
4       dirb/filee
4       dirb/dirc
12      dirb
```

如果du不指定单位的话，默认显示单位为K.

``-b'' 列出的值以bytes为单位输出。

``-k'' 以KB为单位输出，和默认不加任何选项的输出值是一样的。

``-m'' 以MB为单位输出

``-h'' 系统自动调节单位，例如文件太小可能就几K，那么就以K为单位显示，如果大到几G，则就以G为单位显示。

```
[root@localhost ~]# du -b /etc/passwd
1181    /etc/passwd
[root@localhost ~]# du -k /etc/passwd
4       /etc/passwd
[root@localhost ~]# du -m /etc/passwd
1       /etc/passwd
[root@localhost ~]# du -h /etc/passwd
4.0K    /etc/passwd
```

``-c'' 最后加总

```
[root@localhost ~]# du -c dirb
4       dirb/dirc
12      dirb
12      总用量
[root@localhost ~]# du dirb
4       dirb/dirc
12      dirb
```

``-s'' 只列出总和

```
[root@localhost ~]# du -s dirb
12      dirb
```

阿铭习惯用 du -sh filename 这样的形式。

10.2 磁盘的分区和格式化

阿铭经常做的事情就是拿一个全新的磁盘来分区并格式化。这也说明了作为一个linux系统管理员，对于磁盘的操作必须要熟练。所以请您认真学习该部分内容。在正式介绍Linux下分区工具之前，阿铭需要先给虚拟机添加一块磁盘，以便于我们做后续的实验，如果您也是使用vmware 虚拟机，请跟着阿铭一起来做吧。

1. 先关闭正在运行的Linux系统 `init 0`.
2. 到vmware的Linux虚拟机界面，点 ``Edit virtual machine settings'', 点一下左侧靠下面的 ``Add...'' 按钮。
3. 在左侧选中 ``Hard Disk'' 默认就是这一行，点右下角的 ``Next'', 继续点 ``Next''.
4. ``Virtual disk type'' 选择 IDE, 点 ``Next''
5. 继续点 ``Next'', ``Disk size'' 默认即可，最后点 ``Finish''.

命令：fdisk

fdisk 是Linux下硬盘的分区工具，是一个非常实用的命令，但是fdisk只能划分小于2T的分区。

语法：`fdisk [-l ] [设备名称]` 选项只有一个。

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

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

Disk /dev/sda: 17.2 GB, 17179869184 bytes
255 heads, 63 sectors/track, 2088 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00018d63

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1          13      102400   83  Linux
Partition 1 does not end on cylinder boundary.
/dev/sda2              13         274     2097152   82  Linux swap / Solaris
Partition 2 does not end on cylinder boundary.
/dev/sda3             274        2089    14576640   83  Linux

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

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

Disk /dev/sda: 17.2 GB, 17179869184 bytes
255 heads, 63 sectors/track, 2088 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00018d63

   Device Boot      Start         End      Blocks   Id  System
```

```
/dev/sda1   *           1          13      102400   83  Linux
Partition 1 does not end on cylinder boundary.
/dev/sda2              13         274     2097152   82  Linux swap / Solaris
Partition 2 does not end on cylinder boundary.
/dev/sda3             274        2089    14576640   83  Linux
```

可以看到刚才阿铭加的一块磁盘 /dev/sdb 的信息。

``fdisk'' 如果不加 ``-l'' 则进入另一个模式，在该模式下，可以对磁盘进行分区操作。

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

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
         switch off the mode (command 'c') and change display units to
         sectors (command 'u').

Command (m for help):
```

如果您输入 `m' 会列出常用的命令：

```
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)
```

如果您的英文好，我想您不难理解这些字母的功能。阿铭常用的有'p', `n', `d', `w', `q'.

``p'' 打印当前磁盘的分区情况。

```
Command (m for help): p

Disk /dev/sda: 17.2 GB, 17179869184 bytes
255 heads, 63 sectors/track, 2088 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00018d63

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1          13      102400   83  Linux
Partition 1 does not end on cylinder boundary.
/dev/sda2              13         274     2097152   82  Linux swap / Solaris
Partition 2 does not end on cylinder boundary.
/dev/sda3             274        2089    14576640   83  Linux
```

`n' 建立一个新的分区。

`w' 保存操作。

`q' 退出。

`d' 删除一个分区

下面阿铭会把刚才增加的磁盘/dev/sdb进行分区操作。先使用 `p' 命令看一下/dev/sdb的分区状况：

```
[root@localhost ~]# fdisk /dev/sdb
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0xf4121235.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
         switch off the mode (command 'c') and change display units to
         sectors (command 'u').

Command (m for help): p

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xf4121235

   Device Boot      Start         End      Blocks   Id  System

Command (m for help):
```

可以看到目前/dev/sdb没有任何分区，下面阿铭给它建立第一个分区：

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
```

使用 `n' 命令新建分区，它会提示是要 `e' (扩展分区) 还是 `p' (主分区) [1] 阿铭的选择是 `p'，于是输入 `p' 然后回车

```
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-1044, default 1): 1
Last cylinder, +cylinders or +size{K,M,G} (1-1044, default 1044): +1000M
```

输入 `p' 后，会提示分区数，这里阿铭写 `1', 因为是第一个分区，当然您也可以写 `2' 或 `3', 如果您直接回车的话，会继续提示您必须输入一个数字，接着又提示第一个柱面从哪里开始，默认是 `1', 您可以写一个其他的数字，不过这样就浪费了空间，所以还是写 `1' 吧，或者您直接回车也会按 `1' 处理，接着是让输入最后一个柱面的数值，也就是说您需要给这个分区分多大空间，关于柱面是多大阿铭不再细究，您只需要掌握阿铭教给您的方法即可，即写 ``+1000M'', 这样即方便又不容易出错。用 `p' 查看已经多出了一个分区：

[1] 磁盘分区有三种形式：主分区、扩展分区和逻辑分区。主分区最多可以有四个，如果想再多分分区，则需要先分三个主分区，然后第四个分为扩展分区，然后再把扩展分区分成若干个逻辑分区，逻辑分区最多可以分多少个？之前阿铭使用ide接口的磁盘尝试过(hda, hdb这样的磁盘)，最多可以分10个，至于scsi接口的磁盘(sda, sdb)最多可以分多少个，阿铭没有做实验，这留给您来做吧。

```
Command (m for help): p

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0600660a

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1         128     1028128+  83  Linux
```

继续上面的操作，一直创建主分区到4, 然后再一次创建分区的时候则会提示：

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

这是因为，在linux中最多只能创建4个主分区，那如果您想多创建几个分区如何做？很容易，在创建完第三个分区后，创建第四个分区时选择扩展分区。

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
e
Selected partition 4
First cylinder (385-1044, default 385):
Using default value 385
Last cylinder, +cylinders or +size{K,M,G} (385-1044, default 1044):
Using default value 1044

Command (m for help): p

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xef267349

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1         128     1028128+  83  Linux
/dev/sdb2             129         256     1028160   83  Linux
/dev/sdb3             257         384     1028160   83  Linux
/dev/sdb4             385        1044     5301450    5  Extended
```

扩展分区，在最后一列显示为 ``Extended'', 接下来继续创建分区：

```
Command (m for help): n
First cylinder (385-1044, default 385):
Using default value 385
Last cylinder, +cylinders or +size{K,M,G} (385-1044, default 1044): +1000M

Command (m for help): p

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
```

```
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xef267349

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1         128     1028128+  83  Linux
/dev/sdb2             129         256     1028160   83  Linux
/dev/sdb3             257         384     1028160   83  Linux
/dev/sdb4             385        1044     5301450    5  Extended
/dev/sdb5             385         512     1028128+  83  Linux
```

这时候再分区和以前有区别了，不再选择是主分区还是扩展分区了，而是直接定义大小。有一点阿铭要讲一下，当分完三个主分区后，第四个扩展分区需要把剩余的磁盘空间全部划分给扩展分区，不然的话剩余的空间会浪费，因为分完扩展分区后，再划分新的分区时是在已经划分的扩展分区里来分的。其中/dev/sdb4为扩展分区，这个分区是不可以格式化的，您可以把它看成是一个空壳子，能使用的为/dev/sdb5, 其中/dev/sdb5为/dev/sdb4的子分区，这个子分区叫做逻辑分区。如果您发现分区分的不合适，想删除掉某个分区怎么办？这就用到了 `d' 命令：

```
Command (m for help): d
Partition number (1-5): 1

Command (m for help): p

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x7b9a6af3

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb2             129         256     1028160   83  Linux
/dev/sdb3             257         384     1028160   83  Linux
/dev/sdb4             385        1044     5301450    5  Extended
/dev/sdb5             385         512     1028128+  83  Linux

Command (m for help): d
Partition number (1-5): 5

Command (m for help): p

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x7b9a6af3

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb2             129         256     1028160   83  Linux
/dev/sdb3             257         384     1028160   83  Linux
/dev/sdb4             385        1044     5301450    5  Extended

Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
l
First cylinder (385-1044, default 385):
```

```
Using default value 385
Last cylinder, +cylinders or +size{K,M,G} (385-1044, default 1044): +1000M

Command (m for help): p

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x7b9a6af3

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb2             129         256     1028160   83  Linux
/dev/sdb3             257         384     1028160   83  Linux
/dev/sdb4             385        1044     5301450    5  Extended
/dev/sdb5             385         512     1028128+  83  Linux

Command (m for help): d
Partition number (1-5): 4

Command (m for help): p

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x7b9a6af3

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb2             129         256     1028160   83  Linux
/dev/sdb3             257         384     1028160   83  Linux
```

输入 `d' 会提示要删除哪个分区，可以选择从 1-5 其中1-3是主分区(sdb1, sdb2, sdb3)，4是扩展分区(sdb4)，5是逻辑分区 [1] (sdb5)，如果输入5，则直接把逻辑分区sdb5删除掉，但是如果输入4的话，会把整个扩展分区sdb4干掉，当然也包含扩展分区里面的逻辑分区sdb5。在刚才的分区界面直接 Ctrl + C 退出来，这样刚刚的分区全部都取消了，咱们重新来做分区：

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

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
         switch off the mode (command 'c') and change display units to
         sectors (command 'u').

Command (m for help): p

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x7b9a6af3

   Device Boot      Start         End      Blocks   Id  System

Command (m for help): n
Command action
```

```
   e   extended
   p   primary partition (1-4)
e
Partition number (1-4): 1
First cylinder (1-1044, default 1): 1
Last cylinder, +cylinders or +size{K,M,G} (1-1044, default 1044): 1044

Command (m for help): p

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x7b9a6af3

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1        1044     8385898+   5  Extended

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

如果把第一个分区分为扩展分区，并且把全部空间都分给扩展分区的话，再继续分区的话，会提示的分区类型为主分区还是逻辑分区(logical), 用 `l' 表示逻辑分区，逻辑分区的id是从5开始的，因为前四个id为主分区或者扩展分区。既然阿铭把所有磁盘空间都分为了扩展分区，如果您在这里选择 `p' 则会报错：

```
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
p
Partition number (1-4): 2
No free sectors available
```

这是因为没有足够空间分给主分区了，那我们就分逻辑分区：

```
Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
l
First cylinder (1-1044, default 1): 1
Last cylinder, +cylinders or +size{K,M,G} (1-1044, default 1044): +1000M

Command (m for help): p

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x7b9a6af3

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1        1044     8385898+   5  Extended
/dev/sdb5               1         128     1028097   83  Linux
```

```
Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
l
First cylinder (129-1044, default 129): 129
Last cylinder, +cylinders or +size{K,M,G} (129-1044, default 1044): +1000M

Command (m for help): p

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x7b9a6af3

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1        1044     8385898+   5  Extended
/dev/sdb5               1         128     1028097   83  Linux
/dev/sdb6             129         256     1028128+  83  Linux
```

分区完后，需要输入 `w' 命令来保存我们的配置：

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

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

再使用 fdisk -l /dev/sdb 查看分区情况：

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

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x7b9a6af3

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1        1044     8385898+   5  Extended
/dev/sdb5               1         128     1028097   83  Linux
/dev/sdb6             129         256     1028128+  83  Linux
```

通过以上操作，相信您也学会了用fdisk来分区了吧。但阿铭要提醒您，不要闲着没事分区玩儿，这操作的危险性是很高的，一不留神就把服务器上的数据全部给分没有了。所以在您执行分区操作的时候，请保持百分之二百的细心，切记切记！

10.3 格式化磁盘分区

命令：mke2fs, mkfs.ext2, mkfs.ext3, mkfs.ext4

当用man查询这四个命令的帮助文档时，您会发现我们看到了同一个帮助文档，这说明四个命令是一样的。mke2fs常用的选项有：

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

`-i' 设定inode的大小

`-N' 设定inode数量，有时使用默认的inode数不够用，所以要自定设定inode数量。

`-c' 在格式化前先检测一下磁盘是否有问题，加上这个选项后会非常慢

`-L' 预设该分区的标签label

`-j' 建立ext3格式的分区，如果使用mkfs.ext3 就不用加这个选项了

`-t' 用来指定什么类型的文件系统，可以是ext2, ext3 也可以是 ext4.

```
[root@localhost ~]# mke2fs -t ext4 /dev/sdb5
mke2fs 1.41.12 (17-May-2010)
文件系统标签=
操作系统:Linux
块大小=4096 (log=2)
分块大小=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
64256 inodes, 257024 blocks
12851 blocks (5.00%) reserved for the super user
第一个数据块=0
Maximum filesystem blocks=264241152
8 block groups
32768 blocks per group, 32768 fragments per group
8032 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

正在写入inode表: 完成
Creating journal (4096 blocks): 完成
Writing superblocks and filesystem accounting information: 完成

This filesystem will be automatically checked every 24 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

指定文件系统格式为ext4, 该命令等同于 mkfs.ext4 /dev/sdb5. 目前CentOS 6 默认文件系统格式为ext4, 所以以后您遇到需要格式磁盘分区的时候，直接指定格式为ext4即可，但早期的版本CentOS 5 是使用ext3作为默认的文件系统的，所以您可以根据操作系统的版本来决定格式化什么格式的文件系统。在上面的例子中，您是否有注意到一些指标呢？其中一个指标是 ``块大小=4096'' 这里涉及到一个 ``块'' 的概念，磁盘在被格式化的时候会预先规定好每一个块的大小，然后再把所有的空间分割成一个一个的小块，存数据的时候也是一个块一个块的去写入。所以如果您的磁盘存的都是特别小特别小的文件，比如说1k或者2k，那么建议在格式化磁盘的时候指定块数值小一点。ext文件系统默认块大小为4096也就是4k. 在格式化的时候，可以指定块大小为1024, 2048, 4096(它们是成倍增加的)，虽然格式化的时候可以指定块大小超过4096，但是一旦超过4096则不能正常挂载，如何指定块大小？

```
[root@localhost ~]# mke2fs -t ext4 -b 8192 /dev/sdb5
Warning: blocksize 8192 not usable on most systems.
mke2fs 1.41.12 (17-May-2010)
mke2fs: 8192-byte blocks too big for system (max 4096)
无论如何也要继续? (y,n) y
Warning: 8192-byte blocks too big for system (max 4096), forced to continue
文件系统标签=
操作系统:Linux
块大小=8192 (log=3)
分块大小=8192 (log=3)
Stride=0 blocks, Stripe width=0 blocks
64256 inodes, 128512 blocks
```

```
6425 blocks (5.00%) reserved for the super user
第一个数据块=0
Maximum filesystem blocks=134201344
2 block groups
65528 blocks per group, 65528 fragments per group
32128 inodes per group
Superblock backups stored on blocks:
        65528

正在写入inode表: 完成
Creating journal (4096 blocks): 完成
Writing superblocks and filesystem accounting information: 完成

This filesystem will be automatically checked every 28 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

指定块大小为8192会提示，块值设置太大了，我们直接输入 `y' 强制格式化，您还可以尝试指定更大的数字。

```
[root@localhost ~]# mke2fs -t ext4 -L TEST -b 8192 /dev/sdb5
```

可以使用 `-L' 来指定标签。标签会在挂载磁盘的时候使用，另外也可以写到配置文件里，稍后阿铭介绍。关于格式化的这一部分，阿铭建议您除非有需求，否则不需要指定块大小，也就是说，您只需要记住这两个选项： `-t' 和 `-L' 即可。

命令：e2label

用来查看或修改分区的标签，阿铭很少使用，您只要了解一下即可。

```
[root@localhost ~]# e2label /dev/sdb5
TEST
[root@localhost ~]# e2label /dev/sdb5 TEST123
[root@localhost ~]# e2label /dev/sdb5
TEST123
```

10.4 挂载/卸载磁盘

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

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

命令：mount

如果不加任何选项，直接运行 ``mount'' 命令，会显示如下信息：

```
[root@localhost ~]# mount
/dev/sda3 on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw)
```

```
/dev/sda1 on /boot type ext4 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
```

这个命令可以查看当前系统已经挂载的所有分区，以及分区文件系统的类型，挂载点和一些选项等信息，所以您如果想知道某个分区的文件系统类型直接用该命令查看即可。下面我们先建立一个空目录，然后在目录里建一个空白文档。

```
[root@localhost ~]# mkdir /newdir
[root@localhost ~]# touch /newdir/newfile.txt
[root@localhost ~]# ls /newdir/newfile.txt
/newdir/newfile.txt
```

然后把刚才格式化的 /dev/sdb5 挂载到 /newdir 上。

```
[root@localhost ~]# mount /dev/sdb5 /newdir/
mount: wrong fs type, bad option, bad superblock on /dev/sdb5,
       missing codepage or helper program, or other error
       In some cases useful info is found in syslog - try
       dmesg | tail  or so
```

不能完成挂载，根据提示可以查看一下错误信息：

```
[root@localhost ~]# dmesg |tail
eth0: no IPv6 routers present
 sdb: sdb1 < sdb5 >
 sdb:
 sdb: sdb1 < sdb5 sdb6 >
EXT4-fs (sdb5): bad block size 8192
EXT4-fs (sdb5): bad block size 8192
EXT4-fs (sdb5): bad block size 8192
EXT4-fs (sdb5): bad block size 8192
EXT4-fs (sdb5): mounted filesystem with ordered data mode. Opts:
EXT4-fs (sdb5): bad block size 8192
```

可以看到，我的/dev/sdb5指定的块值8192不合法，所以只能重新格式化磁盘。

```
[root@localhost ~]# mke2fs -t ext4 -L TEST  /dev/sdb5
```

使用默认块值即可。我们继续挂载sdb5:

```
[root@localhost ~]# mount /dev/sdb5 /newdir/
[root@localhost ~]# ls /newdir/
lost+found
[root@localhost ~]# df -h
文件系统              容量  已用  可用 已用%% 挂载点
/dev/sda3              14G  1.5G   12G  11% /
tmpfs                 160M     0  160M   0% /dev/shm
/dev/sda1              97M   27M   66M  29% /boot
/dev/sdb5             989M   18M  921M   2% /newdir
```

把 /dev/sdb5 挂载到 /newdir 后，原来在 /newdir 下的 newfile.txt 被覆盖了，通过 `df -h` 可以看到刚刚挂载的分区，我们也可以使用LABEL的方式挂载分区：

```
[root@localhost ~]# umount /newdir/
[root@localhost ~]# df -h
文件系统              容量  已用  可用 已用%% 挂载点
/dev/sda3              14G  1.5G   12G  11% /
tmpfs                 160M     0  160M   0% /dev/shm
/dev/sda1              97M   27M   66M  29% /boot
```

```
[root@localhost ~]# mount LABEL=TEST /newdir
[root@localhost ~]# df -h
文件系统              容量  已用  可用 已用% 挂载点
/dev/sda3              14G  1.5G   12G  11% /
tmpfs                 160M     0  160M   0% /dev/shm
/dev/sda1              97M   27M   66M  29% /boot
/dev/sdb5             989M   18M  921M   2% /newdir
```

本例中用到了 ``umount'' 命令，这个是用来卸载磁盘分区的，稍后阿铭介绍。mount 命令常用的选项有：'-a', `-t', `-o'. 在讲 `-a' 选项前，我们有必要先了解一下这个文件 /etc/fstab.

```
[root@localhost ~]# cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Tue May  7 17:51:27 2013
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=95297b81-538d-4d96-870a-de90255b74f5 /                       ext4    defaults        1 1
UUID=a593ff68-2db7-4371-8d8c-d936898e9ac9 /boot                   ext4    defaults        1 2
UUID=ff042a91-b68f-4d64-9759-050c51dc9e8b swap                    swap    defaults        0 0
tmpfs                   /dev/shm                tmpfs   defaults        0 0
devpts                  /dev/pts                devpts  gid=5,mode=620  0 0
sysfs                   /sys                    sysfs   defaults        0 0
proc                    /proc                   proc    defaults        0 0
```

这个文件是系统启动时，需要挂载的各个分区。第一列就是分区的标识，可以写分区的LABEL，也可以写分区的UUID(等会阿铭会着重讲一下这个概念)，当然也可以写分区名(/dev/sda1)；第二列是挂载点；第三列是分区的格式；第四列则是mount的一些挂载参数，等下会详细介绍一下有哪些参数，一般情况下，直接写defaults即可；第五列的数字表示是否被dump备份，是的话这里就是1，否则就是0；第六列是开机时是否自检磁盘。1，2都表示检测，0表示不检测，在Redhat/CentOS中，这个1，2还有个说法，/ 分区必须设为1，而且整个fstab中只允许出现一个1，这里有一个优先级的说法。1比2优先级高，所以先检测1，然后再检测2，如果有多个分区需要开机检测那么都设置成2吧，1检测完了后会同时去检测2。下面该说说第四列中常用到的参数了。

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

``auto/noauto''：开机自动挂载/不自动挂载；

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

``ro''：按只读权限挂载 ；

``rw''：按可读可写权限挂载 ；

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

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

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

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

``grquota''：启动群组磁盘配额模式；

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

```
LABEL=TEST              /newdir                 ext4    defaults        0 0
```

然后卸载掉刚才我们已经挂载的/dev/sdb5

```
[root@localhost ~]# umount /dev/sdb5
[root@localhost ~]# df -h
文件系统              容量  已用  可用 已用%% 挂载点
/dev/sda3              14G  1.5G   12G  11% /
tmpfs                 160M     0  160M   0% /dev/shm
/dev/sda1              97M   27M   66M  29% /boot
```

使用 df -h 查看已经成功卸载 /dev/sdb5 下面执行命令 mount -a

```
[root@localhost ~]# mount -a
[root@localhost ~]# df -h
文件系统              容量  已用  可用 已用%% 挂载点
/dev/sda3              14G  1.5G   12G  11% /
tmpfs                 160M     0  160M   0% /dev/shm
/dev/sda1              97M   27M   66M  29% /boot
/dev/sdb5             989M   18M  921M   2% /newdir
```

此时，多出来一个 /dev/sdb5 挂载到了 /newfir 下。这就是 mount -a 命令执行的结果，这个 `-a' 选项会把/etc/fstab中出现的所有磁盘分区挂载上。

```
[root@localhost ~]# umount /newdir
[root@localhost ~]# mount -t ext4 /dev/sdb5 /newdir
[root@localhost ~]# df -h
文件系统              容量  已用  可用 已用%% 挂载点
/dev/sda3              14G  1.5G   12G  11% /
tmpfs                 160M     0  160M   0% /dev/shm
/dev/sda1              97M   27M   66M  29% /boot
/dev/sdb5             989M   18M  921M   2% /newdir
```

`-t' 选项用来指定挂载的分区类型，默认不指定会自动识别。

`-o' 选项用来指定挂载的分区有哪些特性，即上面 ``/etc/fatab'' 配置文件中第四列的那些。阿铭经常这样使用这个 `-o' 选项：

```
[root@localhost ~]# mkdir /newdir/dir1
[root@localhost ~]# mount -o remount,ro,sync,noauto /dev/sdb5 /newdir
[root@localhost ~]# mkdir /newdir/dir2
mkdir: 无法创建目录 "/newdir/dir2": 只读文件系统
```

由于指定了 `ro' 参数，所以该分区只读了。通过 mount 命令也可以看到 /dev/sdb5 有 `ro' 选项

```
[root@localhost ~]# mount
/dev/sda3 on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw)
/dev/sda1 on /boot type ext4 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
/dev/sdb5 on /newdir type ext4 (ro,sync)
```

下面阿铭重新挂载，让它恢复读写。

```
[root@localhost ~]# mount -o remount /dev/sdb5 /newdir
[root@localhost ~]# mkdir /newdir/dir2
[root@localhost ~]# ls /newdir/
dir1  dir2  lost+found
```

命令：blkid

阿铭在日常的运维工作中遇到过这样的情况，一台服务器上新装了两块磁盘，磁盘a（在服务器上显示为sdc）和磁盘b（在服务器上显示为sdd），有一次把这两块磁盘都拔掉了，然后再重新插上，重启机器，结果磁盘编号调换了，a变成了sdd，b变成了sdc（这是因为把磁盘插错了插槽），问题来了。通过上边的学习，您挂载磁盘是通过/dev/hdb1 这样的分区名字来挂载的，如果先前加入到了/etc/fstab 中，结果系统启动后则会挂载错分区。那么怎么样避免这样的情况发生？

这就用到了UUID，可以通过 blkid 命令获取各分区的UUID:

```
/dev/sda1: UUID="a593ff68-2db7-4371-8d8c-d936898e9ac9" TYPE="ext4"
/dev/sda2: UUID="ff042a91-b68f-4d64-9759-050c51dc9e8b" TYPE="swap"
/dev/sda3: UUID="95297b81-538d-4d96-870a-de90255b74f5" TYPE="ext4"
/dev/sdb5: LABEL="TEST" UUID="c61117ca-9176-4d0b-be4d-1b0f434359a7" TYPE="ext4"
/dev/sdb6: UUID="c271cb5a-cb46-42f4-9eb4-d2b1a5028e18" SEC_TYPE="ext2" TYPE="ext3"
```

这样可以获得全部磁盘分区的UUID，如果格式化的时候指定了 LABEL 则该命令也会显示LABEL值，甚至连文件系统类型也会显示。当然这个命令后面也可以指定哪个分区：

```
[root@localhost ~]# blkid /dev/sdb5
/dev/sdb5: LABEL="TEST" UUID="c61117ca-9176-4d0b-be4d-1b0f434359a7" TYPE="ext4"
```

获得UUID后，如何使用它呢？

```
[root@localhost ~]# umount /newdir
[root@localhost ~]# mount UUID="c61117ca-9176-4d0b-be4d-1b0f434359a7" /newdir
[root@localhost ~]# df -h
文件系统              容量  已用  可用 已用%% 挂载点
/dev/sda3              14G  1.5G   12G  11% /
tmpfs                 160M     0  160M   0% /dev/shm
/dev/sda1              97M   27M   66M  29% /boot
/dev/sdb5             989M   18M  921M   2% /newdir
```

也可以把下面这行写到 /etc/fstab 中

```
UUID=c61117ca-9176-4d0b-be4d-1b0f434359a7              /newdir                 ext4    defaults        0 0
```

如果想让某个分区开机后就自动挂载，有两个办法可以实现：

1. 在 /etc/fstab 中添加一行，如上例中那行；
2. 把挂载命令写到 /etc/rc.d/rc.local 文件中去，阿铭会经常把想要开机启动的命令加到这个文件中。系统启动完后会执行这个文件中的命令，所以只要您想开机后运行什么命令统统写入到这个文件下面吧，直接放到最后面即可，阿铭把挂载的命令放到该文件的最后一行了：

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

touch /var/lock/subsys/local
mount UUID="c61117ca-9176-4d0b-be4d-1b0f434359a7" /newdir
```

以上两种方法，任选其一，阿铭介绍第二种方法其实也是教给您一个小知识，如何让一些操作行为随系统启动而自动执行。另外，阿铭需要给您一个小建议，那就是挂载磁盘分区的时候，尽量使用UUID或者LABEL这两种方法。

命令：umount

在上面的小实验中，阿铭多次用到这个命令，这个命令也简单的很，后边可以跟挂载点，也可以跟分区名(/dev/hdb1), 但是不可以跟LABEL和UUID.

```
[root@localhost ~]# umount /dev/sdb5
[root@localhost ~]# mount UUID="c61117ca-9176-4d0b-be4d-1b0f434359a7" /newdir
[root@localhost ~]# umount /newdir
[root@localhost ~]# mount UUID="c61117ca-9176-4d0b-be4d-1b0f434359a7" /newdir
```

umount 命令有一个非常有用的选项那就是 `-l', 有时候您会遇到不能卸载的情况：

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

这是因为当前目录为要卸载的分区上，解决办法有两种，一是到其他目录，二是使用 `-l' 选项：

```
[root@localhost newdir]# umount -l /newdir
[root@localhost newdir]# df -h
文件系统              容量  已用  可用 已用% 挂载点
/dev/sda3              14G  1.5G   12G  11% /
tmpfs                 160M     0  160M   0% /dev/shm
/dev/sda1              97M   27M   66M  29% /boot
```

10.5 建立一个swap文件增加虚拟内存

从装系统时就接触过这个swap了，它类似与windows的虚拟内存，分区的时候一般大小为内存的2倍，如果您的内存超过8G，那么您分16G似乎是没有必要了。分16G足够日常交换了。然而，还会有虚拟内存不够用的情况发生。如果真遇到了，莫非还要重新给磁盘分区？当然不能，那我们就增加一个虚拟的磁盘出来。基本的思路就是：建立swapfile -> 格式化为swap格式 -> 启用该虚拟磁盘。

```
[root@localhost ~]# dd if=/dev/zero of=/tmp/newdisk bs=4k count=102400
记录了102400+0 的读入
记录了102400+0 的写出
419430400字节(419 MB)已复制, 2.59193 秒, 162 MB/秒
```

``dd'' 这个命令阿铭经常用到，所以请您也要掌握它的使用方法，其实也不难，用 ``if'' 指定源，基本上除了 ``/dev/zero'' 外基本上不会写别的，而/dev/zero 是UNIX系统特有的一个文件，它可以提供源源不断的 ``0'', 关于它的其他信息请您在网上查一下资料。 ``of'' 指定目标文件， ``bs'' 定义块的大小， ``count'' 定义块的数量，这两个参数的多少决定了目标文件的大小，目标文件大小 = bs x count. 阿铭用dd建了一个大小为400M的文件，然后格式化成swap格式：

```
[root@localhost ~]# mkswap -f /tmp/newdisk
Setting up swapspace version 1, size = 409596 KiB
no label, UUID=29832cab-04b9-4083-a667-9a5795a5d490
```

格式化完后，就可以挂载上使用了：

```
[root@localhost ~]# free -m
          total       used       free     shared    buffers     cached
Mem:           318        314          4          0          5        278
```

```
-/+ buffers/cache:          30        288
Swap:         2047          0       2047
[root@localhost ~]# swapon /tmp/newdisk
[root@localhost ~]# free -m
             total       used       free     shared    buffers     cached
Mem:           318        314          4          0          5        278
-/+ buffers/cache:         31        287
Swap:         2447          0       2447
```

前后对比swap分区多了400M空间。其中 ``free'' 这个命令用来查看内存使用情况， ``-m'' 表示以M为单位显示，阿铭会在后面介绍该命令。

10.6 磁盘配额

磁盘配合其实就是给每个用户分配一定的磁盘额度，只允许他使用这个额度范围内的磁盘空间。在linux系统中，是多用户多任务的环境，所以会有很多人共用一个磁盘的情况。针对每个用户去限定一定量的磁盘空间是有必要的，这样才显得公平。随着硬件成本的降低，服务器上的磁盘资源似乎不再刻意的去限制了，所以磁盘配额也就可有可无了，但是您也需要了解一下这部分内容，用到时必须会操作。

在linux中，用来管理磁盘配额的东西就是quota了。如果您的linux上没有quota，则需要您安装这个软件包 quota-3.13-5.el5.RPM （其实版本是多少无所谓了，关键是这个软件包）。quota在实际应用中是针对整个分区进行限制的。比如，如果我们限制了/dev/sdb1这个分区，而/dev/sdb1 是挂载在/home 目录下的，那么/home 所有目录都会受到限制。

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

命令：quota

``quota'' 用来显示某个组或者某个使用者的限额。

语法：`quota [-guvs] [user,group]`

``-g'' 显示某个组的限额

``-u'' 显示某个用户的限额

``-v'' 显示的意思

``-s'' 选择inod或硬盘空间来显示

命令：quotacheck

``quotacheck'' 用来扫描某一个磁盘的quota空间。

语法：`quotacheck [-auvg] /path`

``-a'' 扫描所有已经mount的具有quota支持的磁盘

``-u'' 扫描某个使用者的文件以及目录

``-g'' 扫描某个组的文件以及目录

``-v'' 显示扫描过程

``-m'' 强制进行扫描

命令：edquota

``edquota'' 用来编辑某个用户或者组的quota值。

语法：`edquota [-u user] [-g group] [-t]`

``-u'' 编辑某个用户的quota

``-g'' 编辑某个组的quota

``-t'' 编辑宽限时间

``-p'' 拷贝某个用户或组的quota到另一个用户或组

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

``Filesystem'' 磁盘分区，如/dev/sdb5

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

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

``inodes'' 目前使用掉的inode的状态，不用修改。

命令：quotaon

``quotaon'' 用来启动quota，在编辑好quota后，需要启动才能是quota生效

语法：quotaon [-a] [-uvg directory]

``-a'' 全部设定的quota启动

``-u'' 启动某个用户的quota

``-g'' 启动某个组的quota

``-s'' 显示相关信息

命令：quotaoff

``quotaoff'' 用来关闭quota, 该命令常用只有一种情况 quotaoff -a 关闭全部的quota.

以上讲了很多quota的相关命令，那么接下来阿铭教您如何在实践应用中去做这个磁盘配额。整个执行过程如下：

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

```
文件系统                1K-块      已用      可用 已用% 挂载点
/dev/sda3             14347632   1899376  11719424  14% /
tmpfs                   163308         0    163308   0% /dev/shm
/dev/sda1                99150     26808     67222  29% /boot
```

阿铭的linux系统中，/home并没有单独占用一个分区。所以需要把/home目录挂载在一个单独的分区下，因为quota是针对分区来限额的。下面阿铭把 /dev/sdb5 挂载到/home 目录下， 编辑 /etc/fstab 把刚才添加的那行修改为：

```
UUID=c61117ca-9176-4d0b-be4d-1b0f434359a7              /home                   ext4    defaults        0 0
```

保存 /etc/fstab 后，运行 mount -a 命令挂载全部的分区。

```
[root@localhost ~]# mount -a
[root@localhost ~]# df -h
文件系统              容量  已用  可用 已用%% 挂载点
/dev/sda3              14G  1.9G   12G  14% /
tmpfs                 160M     0  160M   0% /dev/shm
/dev/sda1              97M   27M   66M  29% /boot
/dev/sdb5             989M   18M  921M   2% /home
```

此时的 /home 为一个单独分区了。

1. 建立测试账户

首先建立一个test用户，则同时建立了一个test组。其中uid和gid都为511 ，然后又建立一个test1账号，使其加入test组，查看/etc/passwd文件发现test和test1用户的gid都为511.

```
[root@localhost ~]# useradd test
[root@localhost ~]# grep test /etc/passwd
test:x:511:511::/home/test:/bin/bash
[root@localhost ~]# useradd -g 511 test1
[root@localhost ~]# grep test1 /etc/passwd
test1:x:512:511::/home/test1:/bin/bash
```

2. 打开磁盘的quota功能

默认linux并没有对任何分区做quota的支持，所以需要我们手动打开磁盘的quota功能，您是否记得，在前面内容中分析/etc/fstab文件的第四列时讲过这个quota选项（usrquota, grpquota），没错，要想打开这个磁盘的quota支持就是需要修改这个第四列的。用vi编辑/etc/fstab 编辑刚才加的那一行，如下:

```
UUID=c61117ca-9176-4d0b-be4d-1b0f434359a7     /home         ext4    defaults,usrquota,grpquota     0 0
```

保存 /etc/fstab 后，重新挂载/home分区。

```
[root@localhost ~]# umount /home/
[root@localhost ~]# mount -a
[root@localhost ~]# mount
/dev/sda3 on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw)
/dev/sda1 on /boot type ext4 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
/dev/sdb5 on /home type ext4 (rw,usrquota,grpquota)
```

使用 mount 命令可以查看到 /home 分区已经加上了 ``usrquota,grpquota'' 两个配额相关的参数。

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

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

```
[root@localhost ~]# quotacheck -augv
```

可能会有一些错误信息，不要管它。看一看您的/home分区下是否多了两个文件(aquota.group, aquota.user)

```
[root@localhost ~]# ll /home/
总用量 44
-rw------- 1 root  root  7168 5月  12 02:07 aquota.group
-rw------- 1 root  root  8192 5月  12 02:07 aquota.user
drwxr-xr-x 2 root  root  4096 5月  12 00:11 dir1
drwx------ 2 root  root 16384 5月  11 23:18 lost+found
drwx------ 3 test  test  4096 5月  12 01:59 test
drwx------ 3 test1 test  4096 5月  12 02:00 test1
```

如果有了，则可以进入下一步了。

4. 启动quota配额

```
[root@localhost ~]# quotaon -av
/dev/sdb5 [/home]: group quotas turned on
/dev/sdb5 [/home]: user quotas turned on
```

5. 编辑用户磁盘配额

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

```
[root@localhost ~]# edquota -u test
```

将下面内容

```
/dev/sdb5                       20          0          0          5        0        0
```

修改为：

```
/dev/sdb5                       20          20000          30000          5        0        0
```

其中单位是Kb，所以soft 值大约为20Mb，hard值为30Mb，保存这个文件，保存的方式跟vi一个文件的方式一样的。下面将test的配额复制给test1.

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

下面继续设定宽限时间：

```
[root@localhost ~]# edquota -t
```

将7days 改为 1days

```
/dev/sdb5                     1days                  1days
```

下面查看一下test以及test1用户的配额吧。

```
[root@localhost ~]# quota -uv test test1
Disk quotas for user test (uid 511):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
      /dev/sdb5      20   20000   30000               5       0       0
Disk quotas for user test1 (uid 512):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
      /dev/sdb5      20   20000   30000               5       0       0
```

6. 编辑组磁盘配额

```
[root@localhost ~]# edquota -g test
```

修改为：

```
/dev/sdb5                       40          40000          50000         10        0        0
```

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

```
[root@localhost ~]# quota -gv test
Disk quotas for group test (gid 511):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
      /dev/sdb5      40   40000   50000              10       0       0
```

7. 设定开机启动

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

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

阿铭建议您最好再扩展学习一下: http://www.lishiming.net/thread-5424-1-1.html

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

TEN

第十章 文本编辑工具VIM

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

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

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

也许您刚刚安装的CentOS系统上没有这个命令，请这样安装它 `yum install -y vim-enhanced`

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

1. 一般模式： 当您vim filename 编辑一个文件时，一进入该文件就是一般模式了。在这个模式下，您可以做的操作有，上下移动光标；删除某个字符；删除某行；复制、粘贴一行或者多行。
2. 编辑模式：一般模式下，是不可以修改某一个字符的，只能到编辑模式了。从一般模式进入编辑模式，只需您按一个键即可（i, I, a, A, o, O, r, R）。当进入编辑模式时，会在屏幕的最下一行出现“INSERT或REPLACE”的字样。从编辑模式回到一般模式只需要按一下键盘左上方的ESC键即可。
3. 命令模式：在一般模式下，输入 ''':'' 或者 ``/'' 即可进入命令模式。在该模式下，您可以搜索某个字符或者字符串，也可以保存、替换、退出、显示行号等等。

下面阿铭教您如何在一个空白文档中写入一段文字，然后保存。

```
[root@localhost ~]# vim test.txt
```

输入vim test.txt直接回车进入一般模式。然后按 ``i'' 字母进入编辑模式，在窗口的左下角会显示 ``-- 插入 --'' 或者 ``-- INSERT --'' 这说明进入插入模式，可以编辑文档。下面阿铭随便写一段文字：

```
This is a test file.
And this is the first time to using "vim".
It's easy to use "vim".
I like to using it, do you like it?
```

如果您编辑完了，想保存的话，需要先按一下键盘左上角的 ``Esc'' 键，此时 ``-- 插入 --'' 或者 ``-- INSERT --'' 消失，然后输入 ''':wq'' 回车就会保存刚才的文字了。

```
This is a test file.
And this is the first time to using "vim".
It's easy to use "vim".
I like to using it, do you like it?
~
```

```
~
:wq
```

这时，看一下test.txt文档的内容吧:

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

其实 ``vim'' 为全键盘操作的编辑器，所以在各个模式下都有很多功能键。下面列举一下，其中阿铭认为常用的会用红色标出，需要您多加练习，另外不常用的您也需要知道。

一般模式下移动光标	
h 或向左方向键	光标向左移动一个字符
j 或者向下方向键	光标向下移动一个字符
K 或者向上方向键	光标向上移动一个字符
l 或者向右方向键	光标向右移动一个字符
Ctrl + f 或者 pageUP 键	屏幕向前移动一页
Ctrl + b 或者 pageDOWN 键	屏幕向后移动一页
Ctrl + d	屏幕向前移动半页
Ctrl + u	屏幕向后移动半页
+	光标移动到非空格符的下一列
-	光标移动到非空格符的上一列
n 空格（n 是数字）	按下数字 n 然后按空格，则光标向右移动 n 个字符，如果该行字符数小于 n，则光标继续从下行开始向右移动，一直到 n
0（数字 0）或者 Shift+6	移动到本行行首
Shift+4	即'$'移动到本行行尾
H	光标移动到当前屏幕的最顶行
M	光标移动到当前屏幕的中央那一行
L	光标移动到当前屏幕的最底行
G	光标移动到文本的最末行
nG（n 是数字）	移动到该文本的第 n 行
gg	移动带该文本的首行
n 回车（n 是数字）	光标向下移动 n 行

一般模式下查找与替换	
/word	向光标之后寻找一个字符串名为 word 的字符串，当找到第一个 word 后，按"n"继续搜后一个
?word	想光标之前寻找一个字符串名为 word 的字符串，当找到第一个 word 后，按"n"继续搜前一个
:n1,n2s/word1/word2/g	在 n1 和 n2 行间查找 word1 这个字符串并替换为 word2，你也可以把"/"换成"#"
:1,$s/word1/word2/g	从第一行到最末行，查找 word1 并替换成 word2
:1,$s/word1/word2/gc	加上 c 的作用是，在替换前需要用户确认
一般模式下删除、复制粘贴	
x,X	x 为向后删除一个字符，X 为向前删除一个字符
nx（n 为数字）	向后删除 n 个字符
dd	删除光标所在的那一行
ndd（n 为数字）	删除光标所在的向下 n 行
d1G	删除光标所在行到第一行的所有数据
dG	删除光标所在行到末行的所有数据
yy	复制光标所在的那行
nyy	复制从光标所在行起向下 n 行
p,P	p 复制的数据从光标下一行粘贴，P 则从光标上一行粘贴
y1G	复制光标所在行到第一行的所有数据
yG	复制光标所在行到末行的所有数据
J	讲光标所在行与下一行的数据结合成同一行
u	还原过去的操作
进入编辑模式	
i	在当前字符前插入字符
I	在当前行行首插入字符
a	在当前字符后插入字符
A	在当前行行末插入字符
o	在当前行下插入新的一行
O	在当前行上插入新的一行
r	替换光标所在的字符，只替换一次
R	一直替换光标所在的字符，一直到按下 ESC

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

暂时就讲这么多了。如果您能全部掌握，那您一定是vim高手啦。如果您觉得太多，只要记住阿铭标红部分即可，其他的用时再过来查就ok啦。下面阿铭给您留一个小作业，希望您能认真完成！

1. 请把/etc/init.d/iptables 复制到/root/目录下，并重命名为test.txt
2. 用vim打开test.txt并设置行号
3. 分别向下、向右、向左、向右移动5个字符
4. 分别向下、向上翻两页
5. 把光标移动到第49行
6. 让光标移动到行末，再移动到行首
7. 移动到test.txt文件的最后一行
8. 移动到文件的首行
9. 搜索文件中出现的 iptables 并数一下一共出现多少个
10. 把从第一行到第三行出现的iptables 替换成iptable
11. 还原上一步操作
12. 把整个文件中所有的iptables替换成iptable
13. 把光标移动到25行，删除字符 ``$''
14. 还原上一步操作
15. 删除第50行
16. 还原上一步操作
17. 删除从37行到42行的所有内容

18. 还原上一步操作
19. 复制48行并粘贴到52行下面
20. 还原上一步操作（按两次u）
21. 复制从37行到42行的内容并粘贴到44行上面
22. 还原上一步操作（按两次u）
23. 把37行到42行的内容移动到19行下面
24. 还原上一步操作（按两次u）
25. 光标移动到首行，把/bin/sh 改成 /bin/bash
26. 在第一行下面插入新的一行，并输入''# Hello!''
27. 保存文档并退出

阿铭建议您最好再扩展学习一下: http://www.lishiming.net/thread-5434-1-1.html

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

ELEVEN

第十一章 文档的压缩与打包

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

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

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

.gz gzip 压缩工具压缩的文件

.bz2 bzip2 压缩工具压缩的文件

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

.tar.gz 可以理解为先用tar打包，然后再gzip压缩

.tar.bz2 同上，先用tar打包，然后再bzip2压缩

12.1 gzip压缩工具

语法： gzip [-d#] filename 其中#为1-9的数字

``-d'' : 解压缩时使用

``-#'' : 压缩等级，1压缩最差，9压缩最好，6为默认

```
[root@localhost ~]# [ -d test ] && rm -rf test
[root@localhost ~]# mkdir test
[root@localhost ~]# mv test.txt test
[root@localhost ~]# cd test
[root@localhost test]# ls
test.txt
[root@localhost test]# gzip test.txt
[root@localhost test]# ls
test.txt.gz
```

您对第一条命令也许很陌生，其实这是两条命令，[ ] 内是一个判断，''-d test'' 判断test目录是否存在，'&&' 为一个连接命令符号，当前面的命令执行成功后，后面的命令才执行。关于这两块内容阿铭会在后面详细讲解。gzip 后面直接跟文件名，就在当前目录下把该文件压缩了，而原文件也会消失。

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

``gzip -d'' 后面跟压缩文件，会解压压缩文件。gzip 是不支持压缩目录的。

```
[root@localhost ~]# gzip test
gzip: test is a directory -- ignored
[root@localhost ~]# ls test
test/  test1  test2/ test3
[root@localhost ~]# ls test
test.txt
```

至于 ``-#'' 选项，平时很少用，使用默认压缩级别足够了。

12.2 bzip2压缩工具

语法： `bzip2 [-dz] filename`

bzip2 只有两个选项需要您掌握。

``-d'' : 解压缩

``-z'' : 压缩

压缩时，可以加 ``-z'' 也可以不加，都可以压缩文件，''-d'' 则为解压的选项：

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

bzip2 同样也不可以压缩目录：

```
[root@localhost test]# cd ..
[root@localhost ~]# bzip2 test
bzip2: Input file test is a directory.
```

12.3 tar压缩工具

tar 本身为一个打包工具，可以把目录打包成一个文件，它的好处是它把所有文件整合成一个大文件整体，方便拷贝或者移动。

语法：`tar [-zjxcvfpP] filename` tar 命令有多个选项，其中不常用的阿铭做了标注。

``-z'' : 同时用gzip压缩

``-j'' : 同时用bzip2压缩

``-x'' : 解包或者解压缩

``-t'' : 查看tar包里面的文件

``-c'' : 建立一个tar包或者压缩文件包

``-v''：可视化

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

``-p''：使用原文件的属性，压缩前什么属性压缩后还什么属性。（不常用）

``-P''：可以使用绝对路径。（不常用）

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

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

首先在test目录下建立test111目录，然后在test111目录下建立test2.txt, 并写入 ``nihao'' 到test2.txt中，接着是用tar把test111打包成test111.tar. 请记住 ``-f'' 参数后跟的是打包后的文件名, 然后再是要打包的目录或者文件。tar 打包后，原文件不会消失，而依旧存在。在上例中，阿铭使用一个特殊的符号 ''!$'' 它表示上一条命令的最后一个参数，比如在本例中，它表示''test111/test2.txt''. tar 不仅可以打包目录也可以打包文件，打包的时候也可以不加 ``-v'' 选项表示，表示不可视化。

```
[root@localhost test]# rm -f test111.tar
[root@localhost test]# tar -cf test.tar test111 test.txt.bz2
[root@localhost test]# ls
test111  test.tar  test.txt.bz2
```

删除原来的test111目录，然后解包test.tar，不管是打包还是解包，原来的文件是不会删除的，而且它会覆盖当前已经存在的文件或者目录。

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

打包的同时使用gzip压缩

tar命令非常好用的一个功能就是可以在打包的时候直接压缩，它支持gzip压缩和bzip2压缩。

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

``-tf'' 可以查看包或者压缩包的文件列表：

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

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

``-zxvf'' 用来解压.tar.gz的压缩包

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

打包的同时使用bzip2压缩

和gzip压缩不同的是，这里使用 ``-cjvf'' 选项来压缩

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

同样可以使用 ``-tf'' 来查看压缩包文件列表

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

解压.tar.bz2 的压缩包也很简单

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

下面介绍一下 `--exclude` 这个选项的使用，因为在日常的管理工作中您也许会用到它。

```
[root@localhost test]# tar -cvf test111.tar --exclude test3.txt test111
test111/
test111/test4.txt
test111/test2.txt
test111/test5/
```

请注意上条命令中，test111.tar 是放到了 `--exclude` 选项的前面。除了可以排除文件，也可以排除目录：

```
[root@localhost test]# rm -f test111.tar
[root@localhost test]# tar -cvf test111.tar --exclude test5 test111
test111/
test111/test4.txt
test111/test3.txt
test111/test2.txt
```

阿铭建议您最好再扩展学习一下: http://www.lishiming.net/thread-5435-1-1.html

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

TWEVEL

第十二章 安装RPM包或者安装源码包

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

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

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

13.1 RPM工具

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

如果您的光驱中还有系统安装盘的话，我们可以通过 `mount /dev/cdrom /mnt` 命令把光驱挂载到/mnt目录下，那么您会在/mnt/Packages目录下看到很多.rpm的文件，这就是RPM包了。

```
[root@localhost ~]# mount /dev/cdrom /mnt/
mount: block device /dev/sr0 is write-protected, mounting read-only
[root@localhost ~]# ls /mnt/
CentOS_BuildTag  Packages                    RPM-GPG-KEY-CentOS-Security-6
EULA             RELEASE-NOTES-en-US.html    RPM-GPG-KEY-CentOS-Testing-6
GPL              repodata                    TRANS.TBL
images           RPM-GPG-KEY-CentOS-6
isolinux         RPM-GPG-KEY-CentOS-Debug-6
[root@localhost ~]# ls /mnt/Packages/|head
389-ds-base-1.2.11.15-11.el6.i686.rpm
389-ds-base-libs-1.2.11.15-11.el6.i686.rpm
abrt-2.0.8-15.el6.centos.i686.rpm
abrt-addon-ccpp-2.0.8-15.el6.centos.i686.rpm
abrt-addon-kerneloops-2.0.8-15.el6.centos.i686.rpm
abrt-addon-python-2.0.8-15.el6.centos.i686.rpm
abrt-cli-2.0.8-15.el6.centos.i686.rpm
abrt-desktop-2.0.8-15.el6.centos.i686.rpm
```

```
abrt-gui-2.0.8-15.el6.centos.i686.rpm
abrt-libs-2.0.8-15.el6.centos.i686.rpm
```

每一个rpm包的名称都由 - 和 . 分成了若干部分。就拿 ``abrt-cli-2.0.8-15.el6.centos.i686.rpm'' 这个包来解释一下， ``abrt-cli'' 为包名， ``2.0.8'' 则为版本信息， ``15.el6.centos'' 为发布版本号， ``i686'' 为运行平台。其中运行平台常见的有i386, i586, i686, x86_64 ，需要您注意的是cpu目前是分32位和64位的，i386,i586和i686都为32位平台，x86_64则代表为64位的平台。另外有些rpm包并没有写具体的平台而是noarch，这代表这个rpm包没有硬件平台限制。例如 ``alacarte-0.10.0-1.fc6.noarch.rpm''. 下面介绍一下rpm常用的命令。

1. 安装一个rpm包

```
[root@localhost ~]# rpm -ivh /mnt/Packages/libjpeg-turbo-devel-1.2.1-1.el6.i686.rpm
Preparing...                ########################################### [100%]
   1:libjpeg-turbo-devel    ########################################### [100%]
```

``-i'' : 安装的意思

``-v'' : 可视化

``-h'' : 显示安装进度

另外在安装一个rpm包时常用的附带参数有：

--force : 强制安装，即使覆盖属于其他包的文件也要安装

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

2. 升级一个rpm包

命令 rpm -Uvh filename

``-U'' : 即升级的意思

3. 卸载一个rpm包

命令 rpm -e filename

这里的filename是通过rpm的查询功能所查询到的，稍后会作介绍。

```
[root@localhost ~]# rpm -qa |grep libjpeg-turbo-devel
libjpeg-turbo-devel-1.2.1-1.el6.i686
[root@localhost ~]# rpm -e libjpeg-turbo-devel
```

卸载时后边跟的filename和安装时的是有区别的，安装时是把一个存在的文件作为参数，而卸载时只需要包名即可。

4. 查询一个包是否安装

命令 rpm -q rpm包名 (这里的包名，是不带有平台信息以及后缀名的)

```
[root@localhost ~]# rpm -q libjpeg-turbo-devel
package libjpeg-turbo-devel is not installed
[root@localhost ~]# rpm -ivh /mnt/Packages/libjpeg-turbo-devel-1.2.1-1.el6.i686.rpm
Preparing...                ########################################### [100%]
   1:libjpeg-turbo-devel    ########################################### [100%]
[root@localhost ~]# rpm -q libjpeg-turbo-devel
libjpeg-turbo-devel-1.2.1-1.el6.i686
```

我们可以使用 rpm -qa 查询当前系统所有安装过的rpm包，限于篇幅，阿铭只列出前十个。

```
[root@localhost ~]# rpm -qa |head
plymouth-core-libs-0.8.3-27.el6.centos.i686
xml-common-0.6.3-32.el6.noarch
sgpio-1.2.0.10-5.el6.i686
iso-codes-3.16-2.el6.noarch
gnome-vfs2-2.24.2-6.el6.i686
libX11-common-1.5.0-4.el6.noarch
curl-7.19.7-35.el6.i686
ca-certificates-2010.63-3.el6_1.5.noarch
cups-libs-1.4.2-48.el6_3.3.i686
kbd-misc-1.15-11.el6.noarch
```

5. 得到一个已安装rpm包的相关信息

命令 rpm -qi 包名 （同样不需要加平台信息与后缀名）

```
[root@localhost ~]# rpm -qi libjpeg-turbo-devel
Name        : libjpeg-turbo-devel          Relocations: (not relocatable)
Version     : 1.2.1                             Vendor: CentOS
Release     : 1.el6                          Build Date: 2013年02月22日 星期五 06时49分08秒
Install Date: 2013年05月13日 星期一 01时37分48秒      Build Host: c6b9.bsys.dev.centos.org
Group       : Development/Libraries         Source RPM: libjpeg-turbo-1.2.1-1.el6.src.rpm
Size        : 321085                           License: wxWidgets
Signature   : RSA/SHA1, 2013年02月24日 星期日 01时53分55秒, Key ID 0946fca2c105b9de
Packager    : CentOS BuildSystem <http://bugs.centos.org>
URL         : http://sourceforge.net/projects/libjpeg-turbo
Summary     : Headers for the libjpeg-turbo library
Description :
This package contains header files necessary for developing programs which
will manipulate JPEG files using the libjpeg-turbo library.
```

6. 列出一个rpm包安装的文件

命令 rpm -ql 包名

```
[root@localhost ~]# rpm -ql libjpeg-turbo-devel
/usr/include/jconfig.h
/usr/include/jerror.h
/usr/include/jmorecfg.h
/usr/include/jpeglib.h
/usr/lib/libjpeg.so
/usr/share/doc/libjpeg-turbo-devel-1.2.1
/usr/share/doc/libjpeg-turbo-devel-1.2.1/coderules.txt
/usr/share/doc/libjpeg-turbo-devel-1.2.1/example.c
/usr/share/doc/libjpeg-turbo-devel-1.2.1/jconfig.txt
/usr/share/doc/libjpeg-turbo-devel-1.2.1/libjpeg.txt
/usr/share/doc/libjpeg-turbo-devel-1.2.1/structure.txt
```

通过上面的命令可以看出文件 ``/usr/lib/libjpeg.so'' 是通过安装 ``libjpeg-turbo-devel'' 这个rpm包得来的。那么反过来如何通过一个文件去查找是由安装哪个rpm包得来的？

7. 列出某一个文件属于哪个rpm包

命令 rpm -qf 文件的绝对路径

```
[root@localhost ~]# rpm -qf /usr/lib/libjpeg.so
libjpeg-turbo-devel-1.2.1-1.el6.i686
```

13.2 yum工具

在前面的章节中，阿铭多次提到yum工具，今天终于该讲它了。这个工具比rpm工具好用多了，当然前提是您使用的linux系统是支持yum的。yum最大的优势在于可以联网去下载所需要的rpm包，然后自动安装，在这个工程中如果要安装的rpm包有依赖关系，yum会帮您解决掉这些依赖关系依次安装所有rpm包。下面阿铭介绍常用的yum 命令。

1. 列出所有可用的rpm包 ``yum list''

```
[root@localhost ~]# yum list |head -n 20
Loaded plugins: fastestmirror, security
Loading mirror speeds from cached hostfile
 * base: mirrors.btte.net
 * extras: mirrors.btte.net
 * updates: mirrors.btte.net
Installed Packages
ConsoleKit.i686                         0.4.1-3.el6               @anaconda-CentOS-201303020136.i386/6.4
ConsoleKit-libs.i686                    0.4.1-3.el6               @anaconda-CentOS-201303020136.i386/6.4
ConsoleKit-x11.i686                     0.4.1-3.el6               @anaconda-CentOS-201303020136.i386/6.4
GConf2.i686                             2.28.0-6.el6              @anaconda-CentOS-201303020136.i386/6.4
MAKEDEV.i686                            3.24-6.el6                @anaconda-CentOS-201303020136.i386/6.4
ORBit2.i686                             2.14.17-3.2.el6_3         @anaconda-CentOS-201303020136.i386/6.4
abrt.i686                               2.0.8-15.el6.cento        @anaconda-CentOS-201303020136.i386/6.4
abrt-addon-ccpp.i686                    2.0.8-15.el6.cento        @anaconda-CentOS-201303020136.i386/6.4
abrt-addon-kerneloops.i686              2.0.8-15.el6.cento        @anaconda-CentOS-201303020136.i386/6.4
abrt-addon-python.i686                  2.0.8-15.el6.cento        @anaconda-CentOS-201303020136.i386/6.4
abrt-cli.i686                           2.0.8-15.el6.centos       @anaconda-CentOS-201303020136.i386/6.4
abrt-libs.i686                          2.0.8-15.el6.centos       @anaconda-CentOS-201303020136.i386/6.4
abrt-tui.i686                           2.0.8-15.el6.centos       @anaconda-CentOS-201303020136.i386/6.4
acl.i686                                2.2.49-6.el6              @anaconda-CentOS-201303020136.i386/6.4
```

限于篇幅，阿铭只列举出来前14个包信息。从上例中可以看到有 ``mirrors.btte.net'' 信息出现，这是在告诉用户，它是从mirrors.btte.net这里下载到的rpm包资源。从上面的例子中您还可以看到最左侧是rpm包名字，中间是版本信息，最右侧是安装信息，如果安装了就显示类似 ``@anaconda-CentOS'', ``@base'' 或者 ``@extras'', 他们前面都会有一个 ``@'' 符号，这很好区分。未安装则显示base或者extras, 如果是该rpm包已安装但需要升级则显示updates. 如果您看的仔细会发现，''yum list'' 会先列出已经安装的包(Installed Packages) 然后再列出可以安装的包(Available Packages)

2. 搜索一个rpm包

命令 `yum search [相关关键词]`

```
[root@localhost ~]# yum search vim
Loaded plugins: fastestmirror, security
Loading mirror speeds from cached hostfile
 * base: mirrors.btte.net
 * extras: mirrors.btte.net
 * updates: mirrors.btte.net
============================================ N/S Matched: vim =============================================
vim-X11.i686 : The VIM version of the vi editor for the X Window System
vim-common.i686 : The common files needed by any version of the VIM editor
vim-enhanced.i686 : A version of the VIM editor which includes recent enhancements
vim-minimal.i686 : A minimal version of the VIM editor

  Name and summary matches only, use "search all" for everything.
```

除了这样搜索外，阿铭常用的是利用grep来过滤

```
[root@localhost ~]# yum list |grep 'vim'
vim-common.i686                   2:7.2.411-1.8.el6          @anaconda-CentOS-201303020136.i386/6.4
vim-enhanced.i686                 2:7.2.411-1.8.el6          @anaconda-CentOS-201303020136.i386/6.4
vim-minimal.i686                  2:7.2.411-1.8.el6          @anaconda-CentOS-201303020136.i386/6.4
vim-X11.i686                      2:7.2.411-1.8.el6          base
```

我们同样可以找到相应的rpm包。

3. 安装一个rpm包

命令 yum install [-y] [rpm包名]

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

```
[root@localhost ~]# yum install vim-X11
Loaded plugins: fastestmirror, security
Loading mirror speeds from cached hostfile
 * base: mirrors.btte.net
 * extras: mirrors.btte.net
 * updates: mirrors.btte.net
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package vim-X11.i686 2:7.2.411-1.8.el6 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package          Arch          Version                      Repository     Size
================================================================================
Installing:
 vim-X11          i686          2:7.2.411-1.8.el6            base          975 k

Transaction Summary
================================================================================
Install       1 Package(s)

Total download size: 975 k
Installed size: 2.1 M
Is this ok [y/N]: y
Downloading Packages:
vim-X11-7.2.411-1.8.el6.i686.rpm                          | 975 kB     00:03
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : 2:vim-X11-7.2.411-1.8.el6.i686                               1/1
  Verifying  : 2:vim-X11-7.2.411-1.8.el6.i686                               1/1

Installed:
  vim-X11.i686 2:7.2.411-1.8.el6

Complete!
```

4. 卸载一个rpm包

命令 yum remove [-y] [rpm包名]

```
[root@localhost ~]# yum remove vim-X11
Loaded plugins: fastestmirror, security
Setting up Remove Process
Resolving Dependencies
--> Running transaction check
---> Package vim-X11.i686 2:7.2.411-1.8.el6 will be erased
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package          Arch          Version                    Repository      Size
================================================================================
Removing:
 vim-X11          i686          2:7.2.411-1.8.el6          @base          2.1 M

Transaction Summary
================================================================================
Remove        1 Package(s)

Installed size: 2.1 M
Is this ok [y/N]: y
Downloading Packages:
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Erasing    : 2:vim-X11-7.2.411-1.8.el6.i686                               1/1
  Verifying  : 2:vim-X11-7.2.411-1.8.el6.i686                               1/1

Removed:
  vim-X11.i686 2:7.2.411-1.8.el6

Complete!
```

卸载和安装一样，也可以直接加上 ``-y'' 选项来省略掉和用户交互的步骤。在这里阿铭要提醒一下，卸载某个rpm包一定要看清楚了，不要连其他重要的rpm包一起卸载了，以免影响正常的业务。

4. 升级一个rpm包

命令 yum update [-y] [rpm包]

```
[root@localhost ~]# yum update libselinux
Loaded plugins: fastestmirror, security
Loading mirror speeds from cached hostfile
 * base: mirrors.btte.net
 * extras: mirrors.btte.net
 * updates: mirrors.btte.net
Setting up Update Process
Resolving Dependencies
--> Running transaction check
---> Package libselinux.i686 0:2.0.94-5.3.el6 will be updated
--> Processing Dependency: libselinux = 2.0.94-5.3.el6 for package: libselinux-utils-2.0.94-5.3.el6.i686
---> Package libselinux.i686 0:2.0.94-5.3.el6_4.1 will be an update
--> Running transaction check
---> Package libselinux-utils.i686 0:2.0.94-5.3.el6 will be updated
---> Package libselinux-utils.i686 0:2.0.94-5.3.el6_4.1 will be an update
--> Finished Dependency Resolution
```

```
Dependencies Resolved

================================================================================
 Package                Arch       Version                  Repository     Size
================================================================================
Updating:
 libselinux             i686       2.0.94-5.3.el6_4.1       updates       108 k
Updating for dependencies:
 libselinux-utils       i686       2.0.94-5.3.el6_4.1       updates        81 k

Transaction Summary
================================================================================
Upgrade       2 Package(s)

Total download size: 189 k
Is this ok [y/N]: y
Downloading Packages:
(1/2): libselinux-2.0.94-5.3.el6_4.1.i686.rpm            | 108 kB     00:00
(2/2): libselinux-utils-2.0.94-5.3.el6_4.1.i686.rpm      |  81 kB     00:00
--------------------------------------------------------------------------------
Total                                           1.0 MB/s | 189 kB     00:00
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Updating   : libselinux-2.0.94-5.3.el6_4.1.i686                          1/4
  Updating   : libselinux-utils-2.0.94-5.3.el6_4.1.i686                    2/4
  Cleanup    : libselinux-utils-2.0.94-5.3.el6.i686                        3/4
  Cleanup    : libselinux-2.0.94-5.3.el6.i686                              4/4
  Verifying  : libselinux-utils-2.0.94-5.3.el6_4.1.i686                    1/4
  Verifying  : libselinux-2.0.94-5.3.el6_4.1.i686                          2/4
  Verifying  : libselinux-2.0.94-5.3.el6.i686                              3/4
  Verifying  : libselinux-utils-2.0.94-5.3.el6.i686                        4/4

Updated:
  libselinux.i686 0:2.0.94-5.3.el6_4.1

Dependency Updated:
  libselinux-utils.i686 0:2.0.94-5.3.el6_4.1

Complete!
```

以上介绍了如何使用yum搜索、安装、卸载以及升级一个rpm包，如果您掌握了这些那么您就已经可以解决日常工作中遇到的与rpm包相关问题了。当然yum工具还有好多其他好用的命令，阿铭不再列举出来，如果您感兴趣就去man 一下吧。除此之外，阿铭还会教您一些关于yum的小应用。

13.3 使用本地的光盘来制作一个yum源

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

a）挂载光盘

```
[root@localhost ~]# mount /dev/cdrom /mnt
```

b）删除/etc/yum.repos.d目录所有的repo文件

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

c） 创建新文件dvd.repo

```
[root@localhost ~]# vim /etc/yum.repos.d/dvd.repo
```

加入以下内容：

```
[dvd]
name=install dvd
baseurl=file:///mnt
enabled=1
gpgcheck=0
```

d） 刷新 repos 生成缓存

```
[root@localhost ~]# yum makecache
```

然后就可以使用yum命令安装您所需要的软件包了

13.4 利用yum工具下载一个rpm包

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

a）首先要安装 yum-downloadonly

```
[root@localhost ~]# yum install -y yum-plugin-downloadonly.noarch
```

如果您的CentOS是5.x版本，则需要安装yum-downloadonly.noarch这个包。

b）下载一个rpm包而不安装

```
[root@localhost ~]# yum install 包名 -y --downloadonly
```

这样虽然下载了，但是并没有保存到我们想要的目录下，那么如何指定目录呢？

c）下载到指定目录

```
[root@localhost ~]# yum install 包名 -y --downloadonly --downloaddir=/usr/local/src
```

下面阿铭下载一个rpm包：

```
[root@localhost ~]# yum install -y yum-presto.noarch  --downloadonly --downloaddir=/usr/local/src/
Loaded plugins: downloadonly, fastestmirror, security
Loading mirror speeds from cached hostfile
 * base: mirrors.btte.net
 * extras: mirrors.btte.net
 * updates: mirrors.btte.net
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package yum-presto.noarch 0:0.6.2-1.el6 will be installed
--> Processing Dependency: deltarpm >= 3.4-2 for package: yum-presto-0.6.2-1.el6.noarch
--> Running transaction check
---> Package deltarpm.i686 0:3.5-0.5.20090913git.el6 will be installed
--> Finished Dependency Resolution
```

```
Dependencies Resolved

================================================================================
 Package          Arch         Version                        Repository  Size
================================================================================
Installing:
 yum-presto       noarch       0.6.2-1.el6                    base        32 k
Installing for dependencies:
 deltarpm         i686         3.5-0.5.20090913git.el6        base        73 k

Transaction Summary
================================================================================
Install       2 Package(s)

Total download size: 105 k
Installed size: 257 k
Downloading Packages:
--------------------------------------------------------------------------------
Total                                           43 MB/s | 105 kB     00:00


exiting because --downloadonly specified
[root@localhost ~]# ls /usr/local/src/
deltarpm-3.5-0.5.20090913git.el6.i686.rpm  yum-presto-0.6.2-1.el6.noarch.rpm
```

13.5 安装源码包

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

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

安装一个源码包，通常需要三个步骤：

1）./configure

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

2） make

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

3） make install

安装步骤，生成相关的软件存放目录和配置文件的过程。

上面介绍的3步并不是所有的源码包软件都一样的，阿铭以前也曾经遇到过，安装步骤并不是这样，也就是说源码包的安装并非具有一定的标准安装步骤。这就需要您拿到源码包解压后，然后进入到目录找相关的帮助文档，通常会以INSTALL或者README为文件名。所以，您一定要去看一下。下面阿铭会编译安装一个源码包来帮您更深刻的去理解如何安装源码包。

1. 下载一个源码包

下载源码包一定要去官方站点去下载，不要在网上随便下载，那样很不安全。因为您下载到的源码包很有可能是被人修改过的。

```
[root@localhost src]# cd /usr/local/src/
[root@localhost src]# wget http://apache.etoak.com/httpd/httpd-2.2.24.tar.bz2
```

阿铭提供的下载地址为apache官方网站上提供的一个镜像，下载速度还可以。在下载之前，阿铭进入到了 ``/usr/local/src'' 目录，这是因为阿铭习惯把源码包都放到这个目录下，这样做的好处是，方便自己和其他管理员维护，所以阿铭给您一个建议，以后下载的源码包都统一放到这个目录下吧。

2. 解压源码包

```
[root@localhost src]# tar jxvf httpd-2.2.24.tar.bz2
```

3. 配置相关的选项，并生成Makefile

```
[root@localhost src]# cd httpd-2.2.24
[root@localhost httpd-2.2.24]# ./configure --help |less
`configure' configures this package to adapt to many kinds of systems.

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

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

Defaults for the options are specified in brackets.

Configuration:
  -h, --help              display this help and exit
      --help=short        display options specific to this package
      --help=recursive    display the short help of all the included packages
  -V, --version           display version information and exit
  -q, --quiet, --silent   do not print `checking ...' messages
      --cache-file=FILE   cache test results in FILE [disabled]
  -C, --config-cache      alias for `--cache-file=config.cache'
  -n, --no-create         do not create output files
      --srcdir=DIR        find the sources in DIR [configure dir or `..']
```

后面的内容阿铭省略掉了，阿铭使用 ./configure --help 命令查看可以使用的选项。一般常用的有 --prefix=PREFIX 这个选项的意思是定义软件包安装到哪里。到这里，阿铭再提一个小小的建议，通常源码包都是安装在/usr/local/目录下的。比如，我们把apache安装在/usr/local/apache2下，那么这里就应该这样写 --prefix=/usr/local/apache2 其他还有好多选项，如果您有耐心可以挨个去看一看都有什么作用。

```
[root@localhost httpd-2.2.24]# ./configure --prefix=/usr/local/apache2
checking for chosen layout... Apache
checking for working mkdir -p... yes
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking target system type... i686-pc-linux-gnu

Configuring Apache Portable Runtime library ...

checking for APR... reconfig
configuring package in srclib/apr now
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking target system type... i686-pc-linux-gnu
```

```
Configuring APR library
Platform: i686-pc-linux-gnu
checking for working mkdir -p... yes
APR Version: 1.4.6
checking for chosen layout... apr
checking for gcc... no
checking for cc... no
checking for cl.exe... no
configure: error: in `/usr/local/src/httpd-2.2.24/srclib/apr':
configure: error: no acceptable C compiler found in $PATH
See `config.log' for more details
configure failed for srclib/apr
```

不幸的是，阿铭一开始就报错了，因为没有gcc编译器，需要先安装一下。

```
[root@localhost httpd-2.2.24]# yum install -y gcc
```

由于gcc依赖的包很多，所以安装时间会长一些。安装完后，再继续上面的步骤。

tcode:

```
[root@localhost httpd-2.2.24]# ./configure --prefix=/usr/local/apache2
```

验证这一步是否成功的命令是：

```
[root@localhost httpd-2.2.24]# echo $?
0
```

返回值如果是 ``0'' 则执行成功，否则就是没有成功。此时就成功生成 Makefile 了。

```
[root@localhost httpd-2.2.24]# ls -l Makefile
-rw-r--r-- 1 root root 8954 5月  13 12:02 Makefile
```

4. 进行编译

```
[root@localhost httpd-2.2.24]# make
-bash: make: command not found
```

又发生错误了，提示 ``make'' 命令没有发现，解决办法是安装make工具。

```
[root@localhost httpd-2.2.24]# yum install -y make
```

继续make

```
[root@localhost httpd-2.2.24]# make
Making all in srclib
make[1]: Entering directory `/usr/local/src/httpd-2.2.24/srclib'
Making all in apr
make[2]: Entering directory `/usr/local/src/httpd-2.2.24/srclib/apr'
make[3]: Entering directory `/usr/local/src/httpd-2.2.24/srclib/apr'
/bin/sh /usr/local/src/httpd-2.2.24/srclib/apr/libtool --silent --mode=compile gcc -g -O2 -pthread   -DHAVE_CONFIG_H -DL
```

编译的时候，就会出现类似这么多乱七八糟的信息，编译的时间比较长，CPU使用率会很高，这是因为CPU高速计算，编译完后，再使用 echo $? 验证一下是否正常成功。

```
[root@localhost httpd-2.2.24]# echo $?
0
```

如果是0的话，就可以执行最后一步了。

5. 安装

```
[root@localhost httpd-2.2.24]# make install
Making install in srclib
make[1]: Entering directory `/usr/local/src/httpd-2.2.24/srclib'
Making install in apr
make[2]: Entering directory `/usr/local/src/httpd-2.2.24/srclib/apr'
make[3]: Entering directory `/usr/local/src/httpd-2.2.24/srclib/apr'
make[3]: Nothing to be done for `local-all'.
make[3]: Leaving directory `/usr/local/src/httpd-2.2.24/srclib/apr'
```

当然您也可以使用 echo $? 看看有没有正确安装，执行完这一步，则会在 ``/usr/local/apache2'' 目录下增加了很多目录。

```
[root@localhost httpd-2.2.24]# ls /usr/local/apache2/
bin    cgi-bin  error   icons    lib   man     modules
build  conf     htdocs  include  logs  manual
```

到此，apache源码的安装就完成了，其实在日常的源码安装工作中，并不是谁都像阿铭这样顺利完成安装的，遇到错误不能完成安装的情况是很多的。通常都是因为缺少某一个库文件导致的。这就需要您仔细琢磨报错信息或者查看当前目录下的 ``config.log'' 去得到相关的信息。另外，如果自己不能解决那就去网上google一下吧，通常您会得到想要的答案。

阿铭建议您最好再扩展学习一下： http://www.lishiming.net/thread-5436-1-1.html

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

THIRTEEN

第十三章 学习 SHELL脚本之前的基础知识

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

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

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

14.1 什么是shell

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

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

1. 记录命令历史

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

与命令历史有关的有一个有意思的字符那就是 `!' 了。常用的有这么几个应用：

1） !! 连续两个 `!', 表示执行上一条指令；

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

2） !n 这里的n是数字，表示执行命令历史中第n条指令，例如 !1002 表示执行命令历史中第1002个命令；

```
[root@localhost ~]# history |grep 1002
 1002  pwd
 1015  history |grep 1002
[root@localhost ~]# !1002
pwd
/root
```

history 命令如果未改动过环境变量，默认可以把最近1000条命令历史打印出来。

3）!字符串 （字符串大于等于1），例如 !pw 表示执行命令历史中最近一次以 `pw' 为开头的指令。

```
[root@localhost ~]# !pw
pwd
/root
```

2. 指令和文件名补全

最开始阿铭就介绍过这个功能了，记得吗？它就是按tab键，它可以帮您补全一个指令，也可以帮您补全一个路径或者一个文件名。连续按两次tab键，系统则会把所有的指令或者文件名都列出来。

3. 别名

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

系统预设的alias指令也就这几个而已，您也可以自定义您想要的指令别名。alias语法很简单， alias [命令别名]=['具体的命令']

```
[root@localhost ~]# alias aming='pwd'
[root@localhost ~]# aming
/root
[root@localhost ~]# unalias aming
[root@localhost ~]# aming
bash: aming: command not found
```

使用 unalias 命令别名 就可以把设置的别名给解除了。

4. 通配符

在bash下，可以使用 * 来匹配零个或多个字符，而用 ？匹配一个字符。

```
[root@localhost ~]# ls -d test*
test1.txt  test2  test3  test.pl  test.txt
[root@localhost ~]# ls -d test?
test2  test3
```

5. 输入输出重定向

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

6. 管道符

前面已经提过过管道符 ``|'', 就是把前面的命令运行的结果丢给后面的命令。

7. 作业控制

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

```
[root@localhost ~]# vi test1.txt
testtestssttstst
```

阿铭使用 ``vi'' 编辑test1.txt, 随便输入一些内容，按 ``ESC'' 后， 使用 ``Ctrl + z'' 使任务暂停：

```
[root@localhost ~]# vi test1.txt

[1]+  Stopped                 vi test1.txt
```

可以看到提示 ``vi test1.txt'' 已经停止了，然后使用fg命令恢复它，此时又进入刚才的 ``vi'' 窗口了。再次使其暂停，然后输入 jobs, 可以看到在被暂停或者在后台运行的任务:

```
[root@localhost ~]# jobs
[1]+  Stopped                 vi test1.txt
```

如果想把暂停的任务丢在后台跑起来，就使用bg命令:

```
[root@localhost ~]# bg
[1]+ vi test1.txt &

[1]+  Stopped                 vi test1.txt
```

但是 vi 似乎并不支持在后台运行，那阿铭换一个其他的命令：

```
[root@localhost ~]# vmstat 1 > /tmp/1.log
^Z
[2]+  Stopped                 vmstat 1 > /tmp/1.log
[root@localhost ~]# jobs
[1]-  Stopped                 vi test1.txt
[2]+  Stopped                 vmstat 1 > /tmp/1.log
[root@localhost ~]# bg 2
[2]+ vmstat 1 > /tmp/1.log &
```

在上面的例子中，又有一个新的知识点需要您知道，那就是多个被暂停的任务会有编号，使用 jobs 命令可以看到两个任务，那么使用bg或者fg的时候，就需要在后面加一个编号了，阿铭使用 bg 2 把第二个被暂停的任务丢到后台跑起来了，丢入后台需要使用在命令后边加一个 & 符号，中间有个空格。本例中的 vmstat 1 这个是用来观察系统状态的一个命令，后面章节阿铭再介绍。

丢到后台的任务如何关掉呢？如果您没有退出刚才的shell, 那么先使用 fg 编号 把任务调到前台，然后使用 ``Ctrl + c'' 结束任务：

```
[root@localhost ~]# fg 2
vmstat 1 > /tmp/1.log
^C
```

另一种情况则是，关闭到当前的shell, 再次打开另一个shell时，使用jobs命令并不会显示在后台运行或者被暂停的任务，要想停掉它的话，则需要先知道其pid, 然后使用kill命令杀死那个进程。

```
[root@localhost ~]# vmstat 1 > /tmp/1.log &
[1] 9433
[root@localhost ~]# ps aux |grep vmstat
root      9433  0.0  0.0   6180   516 pts/2    S    09:57   0:00 vmstat 1
root      9435  0.0  0.0 103308   848 pts/2    S+   09:58   0:00 grep vmstat
```

使用 & 把任务丢入后台运行，它会显示pid信息，如果忘记这个pid，我们还可以使用 ps aux 命令找到那个进程，关于 ps 这个命令阿铭会在后面讲解的。想结束掉该进程，需要使用 kill 命令：

```
[root@localhost ~]# kill 9433
[1]+  已终止               vmstat 1 > /tmp/1.log
```

kill命令语法很简单，直接在后面加pid即可，如果遇到杀不死的进程时，可以在kill 后面加一个选项：`kill -9 [pid]`

14.2 变量

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

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

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

```
[root@localhost ~]# env
HOSTNAME=localhost.localdomain
TERM=xterm
SHELL=/bin/bash
HISTSIZE=1000
SSH_CLIENT=10.72.137.107 50947 22
SSH_TTY=/dev/pts/0
USER=root
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=01;05;37;41:su=
MAIL=/var/spool/mail/root
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin
PWD=/root
LANG=zh_CN.UTF-8
HISTCONTROL=ignoredups
SHLVL=1
HOME=/root
LOGNAME=root
SSH_CONNECTION=10.72.137.107 50947 10.72.137.159 22
LESSOPEN=|/usr/bin/lesspipe.sh %s
G_BROKEN_FILENAMES=1
_=/bin/env
```

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

PATH 决定了shell将到哪些目录中寻找命令或程序 HOME 当前用户主目录 HISTSIZE 历史记录数 LOGNAME 当前用户的登录名 HOSTNAME 指主机的名称 SHELL 前用户Shell类型 LANG 语言相关的环境变量，多语言可以修改此环境变量 MAIL 当前用户的邮件存放目录 PWD 当前目录

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

```
[root@localhost ~]# set
BASH=/bin/bash
BASHOPTS=checkwinsize:cmdhist:expand_aliases:extquote:force_fignore:hostcomplete:interactive_comments:login_shell:progco
BASH_ALIASES=()
```

```
BASH_ARGC=()
BASH_ARGV=()
BASH_CMDS=()
BASH_LINENO=()
BASH_SOURCE=()
BASH_VERSINFO=([0]="4" [1]="1" [2]="2" [3]="1" [4]="release" [5]="i386-redhat-linux-gnu")
BASH_VERSION='4.1.2(1)-release'
COLORS=/etc/DIR_COLORS
COLUMNS=168
DIRSTACK=()
EUID=0
GROUPS=()
G_BROKEN_FILENAMES=1
HISTCONTROL=ignoredups
HISTFILE=/root/.bash_history
HISTFILESIZE=1000
HISTSIZE=1000
HOME=/root
HOSTNAME=localhost.localdomain
HOSTTYPE=i386
ID=0
IFS=$' \t\n'
LANG=zh_CN.UTF-8
LESSOPEN='|/usr/bin/lesspipe.sh %s'
LINES=44
LOGNAME=root
LS_COLORS='rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=01;05;37;41:su
MACHTYPE=i386-redhat-linux-gnu
MAIL=/var/spool/mail/root
MAILCHECK=60
OLDPWD=/root
OPTERR=1
OPTIND=1
OSTYPE=linux-gnu
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin
PIPESTATUS=([0]="0")
PPID=27377
PROMPT_COMMAND='printf "\033]0;%s@%s:%s\007" "${USER}" "${HOSTNAME%%.*}" "${PWD/#$HOME/~}"'
PS1='[\u@\h \W]\$ '
PS2='> '
PS4='+ '
PWD=/root
SHELL=/bin/bash
SHELLOPTS=braceexpand:emacs:hashall:histexpand:history:interactive-comments:monitor
SHLVL=1
SSH_CLIENT='10.72.137.107 50947 22'
SSH_CONNECTION='10.72.137.107 50947 10.72.137.159 22'
SSH_TTY=/dev/pts/0
TERM=xterm
UID=0
USER=root
_=-
colors=/etc/DIR_COLORS
```

set不仅可以显示系统预设的变量，也可以连同用户自定义的变量显示出来。

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

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

虽然您可以自定义变量，但是该变量只能在当前shell中生效。

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

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

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

1） 要想系统内所有用户登录后都能使用该变量

需要在 ``/etc/profile'' 文件最末行加入 `export myname=Aming` 然后运行 `source /etc/profile` 就可以生效了。此时再运行bash命令或者直接 `su - test` 账户可以看到效果。

```
[root@localhost ~]# echo "export myname=Aming" >> /etc/profile
[root@localhost ~]# source !$
source /etc/profile
[root@localhost ~]# bash
[root@localhost ~]# echo $myname
Aming
[root@localhost ~]# exit
exit
[root@localhost ~]# su - test
[test@localhost ~]$ echo $myname
Aming
```

2）只想让当前用户使用该变量

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

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

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

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

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

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

4. 如果变量内容中需要用到其他命令运行结果则可以使用反引号;

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

5. 变量内容可以累加其他变量的内容，需要加双引号;

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

在这里如果您不小心把双引号加错为单引号，将得不到您想要的结果

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

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

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

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

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

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

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

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

```
[root@localhost ~]# export
declare -x G_BROKEN_FILENAMES="1"
declare -x HISTCONTROL="ignoredups"
declare -x HISTSIZE="1000"
declare -x HOME="/root"
declare -x HOSTNAME="localhost.localdomain"
declare -x LANG="zh_CN.UTF-8"
```

```
declare -x LESSOPEN="|/usr/bin/lesspipe.sh %s"
declare -x LOGNAME="root"
declare -x LS_COLORS="rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=01;
declare -x MAIL="/var/spool/mail/root"
declare -x OLDPWD
declare -x PATH="/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin"
declare -x PWD="/root"
declare -x SHELL="/bin/bash"
declare -x SHLVL="3"
declare -x SSH_CLIENT="10.72.137.107 50947 22"
declare -x SSH_CONNECTION="10.72.137.107 50947 10.72.137.159 22"
declare -x SSH_TTY="/dev/pts/0"
declare -x TERM="xterm"
declare -x USER="root"
declare -x abc="123"
declare -x myname="\$LOGNAMEAming"
```

在最后面连同我们自定义的变量都被声明了。

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

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

14.3 系统环境变量与个人环境变量的配置文件

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

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

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

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

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

除了两个系统级别的配置文件外，每个用户的主目录下还有几个这样的隐藏文件：

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

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

.bash_history ：记录命令历史用的。

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

14.4 linux shell中的特殊符号

您在学习linux的过程中，也许您已经接触过某个特殊符号，例如”*”，它是一个通配符号，代表零个或多个字符或数字。下面阿铭就说一说常用到的特殊字符。

1. * 代表零个或多个任意字符。

```
[root@localhost ~]# ls -d test*
test  test1  test2  test3
```

2. ? 只代表一个任意的字符

```
[root@localhost ~]# touch testa
[root@localhost ~]# touch testb.txt
[root@localhost ~]# ls -d test?
test1  test2  test3  testa
```

不管是数字还是字母，只要是一个都能匹配出来。

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

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

4. \ 脱意字符，将后面的特殊符号（例如''*'' ）还原为普通字符。

```
[root@localhost ~]# ls -d test\*
ls: 无法访问test*: 没有那个文件或目录
```

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

```
[root@localhost ~]# cat testb.txt |wc -l
0
```

wc -l 用来计算一个文档有多少行。在这里阿铭一下子列出来很多对您陌生的命令，其实这些命令在日常的处理文档工作中非常实用，所以阿铭需要先简单介绍一下它们，如果您记不住没有关系，以后用到的时候再过来查或者直接用man来查询帮助文档。

命令：cut

用来截取某一个字段

语法： cut -d '分隔字符' [-cf] n 这里的n是数字

-d ：后面跟分隔字符，分隔字符要用单引号括起来

-c ：后面接的是第几个字符

-f ：后面接的是第几个区块

```
[root@localhost ~]# cat /etc/passwd |cut -d ':' -f 1 |head -n5
root
bin
daemon
adm
lp
```

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

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

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

```
[root@localhost ~]# head -n2 /etc/passwd|cut -c1,3,10
ro0
bn:
```

命令：sort

sort 用做排序

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

-t 分隔符 ：作用跟cut的-d一个意思

-n ：使用纯数字排序

-r ：反向排序

-u ：去重复

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

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

如果sort不加任何选项，则从首字符向后，依次按ASCII码值进行比较，最后将他们按升序输出。

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

-t 后面跟分隔符，-k后面跟数字，表示对第几个区域的字符串排序，-n 则表示使用纯数字排序

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

-k3,5 表示从第3到第5区域间的字符串排序，-r表示反向排序

命令：wc

用于统计文档的行数、字符数、词数，常用的选项为：

-l ：统计行数

-m ：统计字符数

-w ：统计词数

```
[root@localhost ~]# wc /etc/passwd
  27   37 1220 /etc/passwd
[root@localhost ~]# wc -l /etc/passwd
27 /etc/passwd
[root@localhost ~]# wc -m /etc/passwd
1220 /etc/passwd
[root@localhost ~]# wc -w /etc/passwd
37 /etc/passwd
```

wc 不跟任何选项，直接跟文档，则会把行数、词数、字符数依次输出。

命令：uniq

去重复的行，阿铭最常用的选项只有一个：

-c ：统计重复的行数，并把行数写在前面

```
[root@localhost ~]# vim testb.txt
```

把下面的内容写入testb.txt, 保存。

```
111
222
111
333
```

使用uniq 的前提是需要先给文件排序，否则不管用。

```
[root@localhost ~]# uniq testb.txt
111
222
111
333
[root@localhost ~]# sort testb.txt |uniq
111
222
333
[root@localhost ~]# sort testb.txt |uniq -c
      2 111
      1 222
      1 333
```

命令：tee

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

```
[root@localhost ~]# echo "aaaaaaaaaaaaaaaaaaaaaaaaa" |tee testb.txt
aaaaaaaaaaaaaaaaaaaaaaaaa
[root@localhost ~]# cat testb.txt
aaaaaaaaaaaaaaaaaaaaaaaaa
```

tee 常用语管道符 ``|'' 后。

命令：tr

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

-d ：删除某个字符，-d 后面跟要删除的字符

-s ：把重复的字符去掉

最常用的就是把小写变大写: tr `[a-z]' `[A-Z]'

```
[root@localhost ~]# head -n2 /etc/passwd |tr '[a-z]' '[A-Z]'
ROOT:X:0:0:ROOT:/ROOT:/BIN/BASH
BIN:X:1:1:BIN:/BIN:/SBIN/NOLOGIN
```

当然替换一个字符也是可以的。

```
[root@localhost ~]# grep 'root' /etc/passwd |tr 'r' 'R'
Root:x:0:0:Root:/Root:/bin/bash
opeRatoR:x:11:0:opeRatoR:/Root:/sbin/nologin
```

不过替换、删除以及去重复都是针对一个字符来讲的，有一定局限性。如果是针对一个字符串就不再管用了，所以阿铭建议您只需简单了解这个tr即可，以后您还会学到更多可以实现针对字符串操作的工具。

命令：split

切割文档，常用选项：

-b ：依据大小来分割文档，单位为byte

```
[root@localhost ~]# mkdir split_dir
[root@localhost ~]# cd !$
cd split_dir
[root@localhost split_dir]# cp /etc/passwd ./
[root@localhost split_dir]# split -b500 passwd
[root@localhost split_dir]# ls
passwd  xaa  xab  xac
```

如果split不指定目标文件名，则会以xaa xab... 这样的文件名来存取切割后的文件。当然我们也可以指定目标文件名：

```
[root@localhost split_dir]# split  -b500 passwd  123
[root@localhost split_dir]# ls
123aa  123ab  123ac  passwd
```

-l ：依据行数来分割文档

```
[root@localhost split_dir]# rm -f 123a*
[root@localhost split_dir]# split -l10 passwd
[root@localhost split_dir]# wc -l *
  27 passwd
  10 xaa
  10 xab
  7 xac
  54 总用量
```

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

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

```
ls testb.txt
testb.txt
```

`!$' 表示上条命中中最后一个变量（总之就是上条命令中最后出现的那个东西）例如上边命令最后是testb.txt那么在当前命令下输入!$则代表testb.txt.

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

```
[root@localhost ~]# ls -d test*;  touch test111; ls -d test*
test  test1  test2  test3  testa  testb.txt
test  test1  test111  test2  test3  testa  testb.txt
```

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

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

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

```
[root@localhost ~]# sleep 30 &
[1] 3260
[root@localhost ~]# jobs
[1]+  Running                 sleep 30 &
```

10. >, >>, 2>, 2>> 前面讲过重定向符号> 以及>> 分别表示取代和追加的意思，然后还有两个符号就是这里的2> 和 2>> 分别表示错误重定向和错误追加重定向，当我们运行一个命令报错时，报错信息会输出到当前的屏幕，如果想重定向到一个文本里，则要用2>或者2>>

```
[root@localhost ~]# ls aaaa
ls: 无法访问aaaa: 没有那个文件或目录
[1]+  Done                    sleep 30
[root@localhost ~]# ls aaaa
ls: 无法访问aaaa: 没有那个文件或目录
[root@localhost ~]# ls aaaa 2> /tmp/error
[root@localhost ~]# cat /tmp/error
ls: 无法访问aaaa: 没有那个文件或目录
[root@localhost ~]# ls aaaa 2>> /tmp/error
[root@localhost ~]# cat /tmp/error
ls: 无法访问aaaa: 没有那个文件或目录
ls: 无法访问aaaa: 没有那个文件或目录
```

11. [ ] 中括号，中间为字符组合，代表中间字符中的任意一个。

```
[root@localhost ~]# ls -d test*
test  test1  test111  test2  test3  testa  testb.txt
[root@localhost ~]# ls -d test[1-3]
test1  test2  test3
[root@localhost ~]# ls -d test[1a3]
test1  test3  testa
[root@localhost ~]# ls -d test[0-9]
test1  test2  test3
```

```
[root@localhost ~]# ls -d test[0-9a-z]
test1  test2  test3  testa
```

12. && 与 ||

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

1. command1；command2
2. command1 && command2
3. command1 || command2

使用 '';'' 时，不管command1是否执行成功都会执行command2；

使用 ``&&'' 时，只有command1执行成功后，command2才会执行，否则command2不执行；

使用 ``||'' 时，command1执行成功后command2 不执行，否则去执行command2，总之command1和command2总有一条命令会执行。

在做实验前，阿铭想把所有的 test* 删除掉，可是删除的时候，却提示说权限不够，下面是阿铭排除问题的过程：

```
[root@localhost ~]# rm -rf test*
rm: 无法删除"test2/test1": 权限不够
rm: 无法删除"test2/test3": 权限不够
rm: 无法删除"test2/test4": 权限不够
[root@localhost ~]# ls test*
test1  test3  test4
[root@localhost ~]# lsattr test*
-----a-------e- test2/test1
----i--------e- test2/test3
-------------e- test2/test4
[root@localhost ~]# chattr -a test2/test1
[root@localhost ~]# chattr -i test2/test3
[root@localhost ~]# rm -rf test*
rm: 无法删除"test2/test1": 权限不够
rm: 无法删除"test2/test3": 权限不够
rm: 无法删除"test2/test4": 权限不够
[root@localhost ~]# ls test*
test1  test3  test4
[root@localhost ~]# ls -ld test*
drwxrwxr-x 2 root root 4096 5月  10 10:12 test2
[root@localhost ~]# ls -l test2/*
-rw-r--r-- 1 root root 6 5月  10 10:20 test2/test1
-rw-r--r-- 1 root root 0 5月  10 10:11 test2/test3
-rw-r--r-- 1 root root 0 5月  10 10:12 test2/test4
[root@localhost ~]# lsattr test2/*
-------------e- test2/test1
-------------e- test2/test3
-------------e- test2/test4
[root@localhost ~]# lsattr test2
-------------e- test2/test1
-------------e- test2/test3
-------------e- test2/test4
[root@localhost ~]# lsattr -d test2
----i--------e- test2
[root@localhost ~]# chattr -i test2/
[root@localhost ~]# rm -rf test2/
```

如果您之前跟着阿铭做过同样的实验，相信您也会出现同样的问题的。接下来阿铭要通过做实验来说明 ``&&'' 与 ``||'' 这两个特殊符号的作用：

```
[root@localhost ~]# touch test1 test3
[root@localhost ~]# ls test2 && touch test2
ls: 无法访问test2: 没有那个文件或目录
[root@localhost ~]# ls test2
ls: 无法访问test2: 没有那个文件或目录
[root@localhost ~]# ls test2 || touch test2
ls: 无法访问test2: 没有那个文件或目录
[root@localhost ~]# ls test*
test1  test2  test3
```

阿铭建议您最好再扩展学习一下： http://www.lishiming.net/thread-5437-1-1.html

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

FOURTEEN

第十四章 正则表达式

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

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

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

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

15.1 grep / egrep

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

语法： `grep [-cinvABC] 'word' filename`

-c ：打印符合要求的行数

-i ：忽略大小写

-n ：在输出符合要求的行的同时连同行号一起输出

-v ：打印不符合要求的行

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

-B ：后跟一个数字，例如 –B2 则表示打印符合要求的行以及上面两行

-C ：后跟一个数字，例如 –C2 则表示打印符合要求的行以及上下各两行

```
[root@localhost ~]# grep -A2 'halt' /etc/passwd
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
```

把包含 `halt' 的行以及这行下面的两行都打印出。

```
[root@localhost ~]# grep -B2 'halt' /etc/passwd
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
```

把包含 `halt' 的行以及这行上面的两行都打印出。

```
[root@localhost ~]# grep -C2 'halt' /etc/passwd
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
```

把包含 `halt' 的行以及这行上面和下面的各两行都打印出。

下面阿铭举几个典型实例帮您更深刻的理解grep.

1. 过滤出带有某个关键词的行并输出行号

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

2. 过滤不带有某个关键词的行，并输出行号

```
[root@localhost ~]# grep -nv 'nologin' /etc/passwd
1:root:x:0:0:root:/root:/bin/bash
6:sync:x:5:0:sync:/sbin:/bin/sync
7:shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
8:halt:x:7:0:halt:/sbin:/sbin/halt
26:test:x:511:511::/home/test:/bin/bash
27:test1:x:512:511::/home/test1:/bin/bash
```

3. 过滤出所有包含数字的行

```
[root@localhost ~]# grep '[0-9]' /etc/inittab
# upstart works, see init(5), init(8), and initctl(8).
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
id:3:initdefault:
```

4. 过滤出所有不包含数字的行

```
[root@localhost ~]# grep -v '[0-9]' /etc/inittab
# inittab is only used by upstart for the default runlevel.
#
# ADDING OTHER CONFIGURATION HERE WILL HAVE NO EFFECT ON YOUR SYSTEM.
#
# System initialization is started by /etc/init/rcS.conf
#
# Individual runlevels are started by /etc/init/rc.conf
#
# Ctrl-Alt-Delete is handled by /etc/init/control-alt-delete.conf
#
```

```
# Terminal gettys are handled by /etc/init/tty.conf and /etc/init/serial.conf,
# with configuration in /etc/sysconfig/init.
#
# For information on how to write upstart event handlers, or how
#
# Default runlevel. The runlevels used are:
#
```

5. 把所有以 `#' 开头的行去除

```
[root@localhost ~]# grep -v '^#' /etc/inittab
id:3:initdefault:
```

6. 去除所有空行和以 `#' 开头的行

```
[root@localhost ~]# grep -v '^#' /etc/crontab |grep -v '^$'
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/
```

在正则表达式中， ``^'' 表示行的开始， ``$'' 表示行的结尾，那么空行则可以用 ``^$'' 表示，如何打印出不以英文字母开头的行呢？

```
[root@localhost ~]# vim test.txt
[root@localhost ~]# cat test.txt
123
abc
456

abc2323
#laksdjf
Allllllll
```

阿铭先在test.txt中写几行字符串，用来做实验。

```
[root@localhost ~]# grep '^[^a-zA-Z]' test.txt
123
456
#laksdjf
[root@localhost ~]# grep '[^a-zA-Z]' test.txt
123
456
abc2323
#laksdjf
```

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

7. 过滤任意一个字符与重复字符

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

. 表示任意一个字符，上例中，就是把符合r与o之间有两个任意字符的行过滤出来， * 表示零个或多个前面的字符。

```
[root@localhost ~]# grep 'ooo*' /etc/passwd
root:x:0:0:root:/root:/bin/bash
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
```

`ooo*' 表示oo, ooo, oooo ... 或者更多的 `o' 现在您是否想到了 `.*' 这个组合表示什么意义？

```
[root@localhost ~]# grep '.*' /etc/passwd |wc -l
27
[root@localhost ~]# wc -l /etc/passwd
27 /etc/passwd
```

`.*' 表示零个或多个任意字符，空行也包含在内。

8. 指定要过滤字符出现的次数

```
[root@localhost ~]# grep 'o\{2\}' /etc/passwd
root:x:0:0:root:/root:/bin/bash
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
```

这里用到了{ }，其内部为数字，表示前面的字符要重复的次数。上例中表示包含有两个o 即 `oo' 的行。注意，{ }左右都需要加上脱意字符 `\', 另外，使用{ }我们还可以表示一个范围的，具体格式是 `{n1,n2}' 其中n1<n2，表示重复n1到n2次前面的字符，n2还可以为空，则表示大于等于n1次。

上面部分讲的grep，另外阿铭常常用到egrep这个工具，简单点讲，后者是前者的扩展版本，我们可以用egrep完成grep不能完成的工作，当然了grep能完成的egrep完全可以完成。如果您嫌麻烦，egrep了解一下即可，因为grep的功能已经足够可以胜任您的日常工作了。下面阿铭介绍egrep不用于grep的几个用法。为了试验方便，阿铭把test.txt 编辑成如下内容：

```
rot:x:0:0:/rot:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
operator:x:11:0:operator:/rooot:/sbin/nologin
roooot:x:0:0:/roooot:/bin/bash
11111111111111111111111111111
aaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

1. 筛选一个或一个以上前面的字符

```
[root@localhost ~]# egrep 'o+' test.txt
rot:x:0:0:/rot:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
operator:x:11:0:operator:/rooot:/sbin/nologin
roooot:x:0:0:/roooot:/bin/bash
[root@localhost ~]# egrep 'oo+' test.txt
operator:x:11:0:operator:/root:/sbin/nologin
operator:x:11:0:operator:/rooot:/sbin/nologin
roooot:x:0:0:/roooot:/bin/bash
[root@localhost ~]# egrep 'ooo+' test.txt
operator:x:11:0:operator:/rooot:/sbin/nologin
roooot:x:0:0:/roooot:/bin/bash
```

和grep 不同的是，egrep这里是使用’+’的。

2. 筛选零个或一个前面的字符

```
[root@localhost ~]# egrep 'o?' test.txt
rot:x:0:0:/rot:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
operator:x:11:0:operator:/rooot:/sbin/nologin
roooot:x:0:0:/roooot:/bin/bash
11111111111111111111111111111
aaaaaaaaaaaaaaaaaaaaaaaaaaaaa
[root@localhost ~]# egrep 'ooo?' test.txt
operator:x:11:0:operator:/root:/sbin/nologin
operator:x:11:0:operator:/rooot:/sbin/nologin
roooot:x:0:0:/roooot:/bin/bash
[root@localhost ~]# egrep 'oooo?' test.txt
operator:x:11:0:operator:/rooot:/sbin/nologin
roooot:x:0:0:/roooot:/bin/bash
```

3. 筛选字符串1或者字符串2

```
[root@localhost ~]# egrep 'aaa|111|ooo' test.txt
operator:x:11:0:operator:/rooot:/sbin/nologin
roooot:x:0:0:/roooot:/bin/bash
11111111111111111111111111111
aaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

4. egrep中( )的应用

```
[root@localhost ~]# egrep 'r(oo)|(at)o' test.txt
operator:x:11:0:operator:/root:/sbin/nologin
operator:x:11:0:operator:/rooot:/sbin/nologin
roooot:x:0:0:/roooot:/bin/bash
```

用( )表示一个整体，例如(oo)+就表示1个 `oo' 或者多个 `oo'

```
[root@localhost ~]# egrep '(oo)+' test.txt
operator:x:11:0:operator:/root:/sbin/nologin
operator:x:11:0:operator:/rooot:/sbin/nologin
roooot:x:0:0:/roooot:/bin/bash
```

15.2 sed工具的使用

grep工具的功能其实还不够强大，grep实现的只是查找功能，而它却不能实现把查找的内容替换掉。以前用vim的时候，可以查找也可以替换，但是只局限于在文本内部来操作，而不能输出到屏幕上。sed工具以及下面要讲的awk工具就能实现把替换的文本输出到屏幕上的功能了，而且还有其他更丰富的功能。sed和awk都是流式编辑器，是针对文档的行来操作的。

1. 打印某行

sed -n 'n'p filename 单引号内的n是一个数字，表示第几行:

```
[root@localhost ~]# sed -n '2'p /etc/passwd
bin:x:1:1:bin:/bin:/sbin/nologin
```

要想把所有行都打印出来可以使用 sed -n '1,$'p filename

```
[root@localhost ~]# sed -n '1,$'p test.txt
rot:x:0:0:/rot:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
```

```
operator:x:11:0:operator:/rooot:/sbin/nologin
roooot:x:0:0:/roooooot:/bin/bash
11111111111111111111111111111111
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

也可以指定一个区间:

```
[root@localhost ~]# sed -n '1,3'p test.txt
rot:x:0:0:/rot:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
operator:x:11:0:operator:/rooot:/sbin/nologin
```

2. 打印包含某个字符串的行

```
[root@localhost ~]# sed -n '/root/'p test.txt
operator:x:11:0:operator:/root:/sbin/nologin
```

grep中使用的特殊字符，如 ^ $. * 等同样也能在sed中使用

```
[root@localhost ~]# sed -n '/^1/'p test.txt
11111111111111111111111111111111
[root@localhost ~]# sed -n '/in$/'p test.txt
operator:x:11:0:operator:/root:/sbin/nologin
operator:x:11:0:operator:/rooot:/sbin/nologin
[root@localhost ~]# sed -n '/r..o/'p test.txt
operator:x:11:0:operator:/root:/sbin/nologin
operator:x:11:0:operator:/rooot:/sbin/nologin
roooot:x:0:0:/roooooot:/bin/bash
[root@localhost ~]# sed -n '/ooo*/'p test.txt
operator:x:11:0:operator:/root:/sbin/nologin
operator:x:11:0:operator:/rooot:/sbin/nologin
roooot:x:0:0:/roooooot:/bin/bash
```

3. -e可以实现多个行为

```
[root@localhost ~]# sed -e '1'p -e '/111/'p -n test.txt
rot:x:0:0:/rot:/bin/bash
11111111111111111111111111111111
```

4. 删除某行或者多行

```
[root@localhost ~]# sed '1'd test.txt
operator:x:11:0:operator:/root:/sbin/nologin
operator:x:11:0:operator:/rooot:/sbin/nologin
roooot:x:0:0:/roooooot:/bin/bash
11111111111111111111111111111111
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
[root@localhost ~]# sed '1,3'd test.txt
roooot:x:0:0:/roooooot:/bin/bash
11111111111111111111111111111111
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
[root@localhost ~]# sed '/oot/'d test.txt
rot:x:0:0:/rot:/bin/bash
11111111111111111111111111111111
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

`d' 这个字符就是删除的动作了，不仅可以删除指定的单行以及多行，而且还可以删除匹配某个字符的行，另外还可以删除从某一行一直到文档末行。

5. 替换字符或字符串

```
[root@localhost ~]# sed '1,2s/ot/to/g' test.txt
rto:x:0:0:/rto:/bin/bash
operator:x:11:0:operator:/roto:/sbin/nologin
operator:x:11:0:operator:/rooot:/sbin/nologin
roooot:x:0:0:/roooot:/bin/bash
11111111111111111111111111111
aaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

上例中的 `s' 就是替换的命令， `g' 为本行中全局替换，如果不加 `g' 只换该行中出现的第一个。除了可以使用 `/' 作为分隔符外，还可以使用其他特殊字符例如 `#' 或者 `@' 都没有问题。

```
[root@localhost ~]# sed 's#ot#to#g' test.txt
rto:x:0:0:/rto:/bin/bash
operator:x:11:0:operator:/roto:/sbin/nologin
operator:x:11:0:operator:/rooto:/sbin/nologin
roooto:x:0:0:/rooooto:/bin/bash
11111111111111111111111111111
aaaaaaaaaaaaaaaaaaaaaaaaaaaaa
[root@localhost ~]# sed 's@ot@to@g' test.txt
rto:x:0:0:/rto:/bin/bash
operator:x:11:0:operator:/roto:/sbin/nologin
operator:x:11:0:operator:/rooto:/sbin/nologin
roooto:x:0:0:/rooooto:/bin/bash
11111111111111111111111111111
aaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

现在思考一下，如何删除文档中的所有数字或者字母？

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

aaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

[0-9]表示任意的数字。这里您也可以写成[a-zA-Z]甚至[0-9a-zA-Z]

```
[root@localhost ~]# sed 's/[a-zA-Z]//g' test.txt
::0:0:/://
::11:0::/://
::11:0::/://
::0:0:/://
11111111111111111111111111111
```

6. 调换两个字符串的位置

```
[root@localhost ~]# sed 's/\(rot\)\(.*\)\(bash\)/\3\2\1/' test.txt
bash:x:0:0:/rot:/bin/rot
operator:x:11:0:operator:/root:/sbin/nologin
operator:x:11:0:operator:/rooot:/sbin/nologin
roooot:x:0:0:/roooot:/bin/bash
11111111111111111111111111111
aaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

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

```
[root@localhost ~]# sed 's/^.*$/123&/' test.txt
123rot:x:0:0:/rot:/bin/bash
123operator:x:11:0:operator:/root:/sbin/nologin
123operator:x:11:0:operator:/rooot:/sbin/nologin
123roooot:x:0:0:/roooot:/bin/bash
12311111111111111111111111111111
123aaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

7. 直接修改文件的内容

```
[root@localhost ~]# sed -i 's/ot/to/g' test.txt
[root@localhost ~]# cat test.txt
rto:x:0:0:/rto:/bin/bash
operator:x:11:0:operator:/roto:/sbin/nologin
operator:x:11:0:operator:/rooto:/sbin/nologin
roooto:x:0:0:/rooooto:/bin/bash
11111111111111111111111111111
aaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

这样就可以直接更改test.txt文件中的内容了。由于这个命令可以直接把文件修改，所以在修改前最好先复制一下文件以免改错。

sed常用到的也就上面这些了，只要您多加练习就能熟悉它了。为了能让您更加牢固的掌握sed的应用，阿铭留几个练习题，希望您能认真完成。

1. 把/etc/passwd 复制到/root/test.txt，用sed打印所有行
2. 打印test.txt的3到10行
3. 打印test.txt 中包含 `root' 的行
4. 删除test.txt 的15行以及以后所有行
5. 删除test.txt中包含 `bash' 的行
6. 替换test.txt 中 `root' 为 `toor'
7. 替换test.txt中 `/sbin/nologin' 为 `/bin/login'
8. 删除test.txt中5到10行中所有的数字
9. 删除test.txt 中所有特殊字符（除了数字以及大小写字母）
10. 把test.txt中第一个单词和最后一个单词调换位置
11. 把test.txt中出现的第一个数字和最后一个单词替换位置
12. 把test.txt 中第一个数字移动到行末尾
13. 在test.txt 20行到末行最前面加 `aaa:'

阿铭希望您能尽量多想一想，答案只是用来作为参考的。

sed习题答案

```
1.  /bin/cp /etc/passwd  /root/test.txt ;  sed -n '1,$'p test.txt
2.  sed -n '3,10'p test.txt
3.  sed -n '/root/'p test.txt
4.  sed '15,$'d  test.txt
5.  sed '/bash/'d test.txt
6.  sed 's/root/toor/g' test.txt
7.  sed 's#sbin/nologin#bin/login#g' test.txt
8.  sed '5,10s/[0-9]//g' test.txt
9.  sed 's/[^0-9a-zA-Z]//g' test.txt
```

```
10.  sed 's/\(^[a-zA-Z][a-zA-Z]*\)\([^a-zA-Z].*\)\([^a-zA-Z]\)\([a-zA-Z][a-zA-Z]*$\)/\4\2\3\1/' test.txt
11.  sed 's#\([^0-9][^0-9]*\)\([0-9][0-9]*\)\([^0-9].*\)\([^a-zA-Z]\)\([a-zA-Z][a-zA-Z]*$\)#\1\5\3\4\2#' test.txt
12.  sed 's#\([^0-9][^0-9]*\)\([0-9][0-9]*\)\([^0-9].*$\)#\1\3\2#' test.txt
13.  sed 's/^.*$/&aaa/' test.txt
```

15.3 awk工具的使用

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

1. 截取文档中的某个段

```
[root@localhost ~]# head -n2 /etc/passwd |awk -F ':' '{print $1}'
root
bin
```

解释一下，-F 选项的作用是指定分隔符，如果不加-F指定，则以空格或者tab为分隔符。 Print为打印的动作，用来打印出某个字段。$1为第一个字段，$2为第二个字段，依次类推，有一个特殊的那就是$0，它表示整行。

```
[root@localhost ~]# head -n2 test.txt |awk -F':' '{print $0}'
rto:x:0:0:/rto:/bin/bash
operator:x:11:0:operator:/roto:/sbin/nologin
```

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

```
[root@localhost ~]# head -n2 test.txt |awk -F':' '{print $1"#"$2"#"$3"#"$4}'
rto#x#0#0
operator#x#11#0
```

2. 匹配字符或字符串

```
[root@localhost ~]# awk '/oo/' test.txt
operator:x:11:0:operator:/rooto:/sbin/nologin
roooto:x:0:0:/rooooto:/bin/bash
```

跟sed很类似吧，不过还有比sed更强大的匹配。

```
[root@localhost ~]# awk -F ':' '$1 ~/oo/' test.txt
roooto:x:0:0:/rooooto:/bin/bash
```

可以让某个段去匹配，这里的’~’就是匹配的意思，继续往下看

```
[root@localhost ~]# awk -F ':' '/root/ {print $1,$3} /test/ {print $1,$3}' /etc/passwd
root 0
operator 11
test 511
test1 512
```

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

3. 条件操作符

```
[root@localhost ~]# awk -F ':' '$3=="0"' /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

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

```
[root@localhost ~]# awk -F ':' '$3>="500"' /etc/passwd
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
tcpdump:x:72:72::/:/sbin/nologin
user11:x:510:502:user11,user11's office,12345678,123456789:/home/user11:/sbin/nologin
test:x:511:511::/home/test:/bin/bash
test1:x:512:511::/home/test1:/bin/bash
```

在上面的例子中，阿铭本想把pid大于等于500的行打印出，但是结果并不是我们的预期，这是因为awk把所有的数字当作字符来对待了，就跟上一章中提到的 sort 排序原理一样。

```
[root@localhost ~]# awk -F ':' '$7!="/sbin/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
test:x:511:511::/home/test:/bin/bash
test1:x:512:511::/home/test1:/bin/bash
```

!= 为不匹配，除了针对某一个段的字符进行逻辑比较外，还可以两个段之间进行逻辑比较。

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

另外还可以使用 && 和 || 表示 ``并且'' 和 ``或者'' 的意思。

```
[root@localhost ~]# awk -F ':' '$3>"5" && $3<"7"' /etc/passwd
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
user11:x:510:502:user11,user11's office,12345678,123456789:/home/user11:/sbin/nologin
test:x:511:511::/home/test:/bin/bash
test1:x:512:511::/home/test1:/bin/bash
```

也可以是或者

```
[root@localhost ~]# awk -F ':' '$3>"5" || $7=="/bin/bash"' /etc/passwd
root:x:0:0:root:/root:/bin/bash
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
```

```
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
tcpdump:x:72:72::/:/sbin/nologin
user11:x:510:502:user11,user11's office,12345678,123456789:/home/user11:/sbin/nologin
test:x:511:511::/home/test:/bin/bash
test1:x:512:511::/home/test1:/bin/bash
```

4. awk的内置变量

   awk常用的变量有：

   NF ：用分隔符分隔后一共有多少段

   NR ：行数

```
[root@localhost ~]# head -n3 /etc/passwd | awk -F ':' '{print NF}'
7
7
7
[root@localhost ~]# head -n3 /etc/passwd | awk -F ':' '{print $NF}'
/bin/bash
/sbin/nologin
/sbin/nologin
```

   NF 是多少段，而$NF是最后一段的值，而NR则是行号。

```
[root@localhost ~]# head -n3 /etc/passwd | awk -F ':' '{print NR}'
1
2
3
```

   我们可以使用行号作为判断条件：

```
[root@localhost ~]# awk 'NR>20' /etc/passwd
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
abrt:x:173:173::/etc/abrt:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
tcpdump:x:72:72::/:/sbin/nologin
user11:x:510:502:user11,user11's office,12345678,123456789:/home/user11:/sbin/nologin
test:x:511:511::/home/test:/bin/bash
test1:x:512:511::/home/test1:/bin/bash
```

   也可以配合段匹配一起使用：

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

5. awk中的数学运算

   awk可以把段值更改：

```
[root@localhost ~]# head -n 3 /etc/passwd |awk -F ':' '$1="root"'
root x 0 0 root /root /bin/bash
root x 1 1 bin /bin /sbin/nologin
root x 2 2 daemon /sbin /sbin/nologin
```

awk还可以对各个段的值进行数学运算：

```
[root@localhost ~]# head -n2 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
[root@localhost ~]# head -n2 /etc/passwd |awk -F ':' '{$7=$3+$4}'
[root@localhost ~]# head -n2 /etc/passwd |awk -F ':' '{$7=$3+$4; print $0}'
root x 0 0 root /root 0
bin x 1 1 bin /bin 2
```

当然还可以计算某个段的总和

```
[root@localhost ~]# awk -F ':' '{(tot=tot+$3)}; END {print tot}' /etc/passwd
2891
```

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

```
[root@localhost ~]# awk -F ':' '{if ($1=="root") print $0}' /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

基本上，正则表达的内容就这些了。但是阿铭要提醒您一下，阿铭介绍的这些仅仅是最基本的东西，并没有提啊深入的去讲sed和awk，但是完全可以满足日常工作的需要，有时候也许您会碰到比较复杂的需求，如果真遇到了就去请教一下google吧。下面出几道关于awk的练习题，希望您要认真完成。

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

awk习题答案

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

阿铭建议您最好再扩展学习一下： http://www.lishiming.net/thread-5438-1-1.html

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

FIFTEEN

第十五章 SHELL脚本

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

终于到shell 脚本这章了，在以前阿铭卖了好多关子说shell脚本怎么怎么重要，确实shell脚本在linux系统管理员的运维工作中非常非常重要。下面阿铭就带您正式进入shell脚本的世界吧。

到现在为止，您明白什么是shell脚本吗？如果明白最好了，不明白也没有关系，相信随着学习的深入您就会越来越了解到底什么是shell脚本。首先它是一个脚本，并不能作为正式的编程语言。因为是跑在linux的shell中，所以叫shell脚本。说白了，shell脚本就是一些命令的集合。举个例子，我想实现这样的操作：

1）进入到/tmp/目录；

2）列出当前目录中所有的文件名；

3）把所有当前的文件拷贝到/root/目录下；

4）删除当前目录下所有的文件。

简单的4步在shell窗口中需要您敲4次命令，按4次回车。这样是不是很麻烦？当然这4步操作非常简单，如果是更加复杂的命令设置需要几十次操作呢？那样的话一次一次敲键盘会很麻烦。所以不妨把所有的操作都记录到一个文档中，然后去调用文档中的命令，这样一步操作就可以完成。其实这个文档呢就是shell脚本了，只是这个shell脚本有它特殊的格式。

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

在正式写shell脚本之前阿铭先给您一条建议：凡是自定义的脚本建议放到/usr/local/sbin/目录下，这样做的目的是，一来可以更好的管理文档；二来以后接管您的管理员都知道自定义脚本放在哪里，方便维护。

16.1 shell脚本的基本结构以及如何执行

下面请跟着阿铭写第一个您的shell脚本吧:

```
[root@localhost ~]# cd /usr/local/sbin/
[root@localhost sbin]# vim first.sh
#! /bin/bash

## This is my first shell script.
## Writen by Aming 2013-05-24.
```

```
date
echo "Hello world!"
```

Shell脚本通常都是以.sh 为后缀名的，这个并不是说不带.sh这个脚本就不能执行，只是大家的一个习惯而已。所以，以后您发现了.sh为后缀的文件那么它可能是一个shell脚本了。test.sh中第一行要以 ``#! /bin/bash'' 开头，它代表的意思是，该文件使用的是bash语法。如果不设置该行，虽然您的shell脚本也可以执行，但是这不符合规范。 # 表示注释，在前面讲过的。后面跟一些该脚本的相关注释内容以及作者和创建日期或者版本等等。当然这些注释并非必须的，如果您懒的很，可以省略掉，但是阿铭不建议省略。因为随着工作时间的逐渐过渡，您写的shell脚本也会越来越多，如果有一天您回头查看自己写过的某个脚本时，很有可能忘记该脚本是用来干什么的以及什么时候写的。所以写上注释是有必要的。另外系统管理员并非只有您一个，如果是其他管理员查看您的脚本，他看不懂岂不是很郁闷。下面该运行一下这个脚本了:

```
[root@localhost sbin]# sh first.sh
2013年 05月 24日 星期五 18:58:02 CST
Hello world!
```

其实shell脚本还有一种执行方法就是:

```
[root@localhost sbin]# ./first.sh
-bash: ./first.sh: 权限不够
[root@localhost sbin]# chmod +x first.sh
[root@localhost sbin]# ./first.sh
2013年 05月 24日 星期五 18:58:51 CST
Hello world!
```

要想使用该种方法运行shell脚本，前提是脚本本身有执行权限，所以需要给脚本加一个 `x' 权限。另外使用sh命令去执行一个shell脚本的时候是可以加-x选项来查看这个脚本执行过程的，这样有利于我们调试这个脚本哪里出了问题:

```
[root@localhost sbin]# sh -x first.sh
+ date
2013年 05月 24日 星期五 20:00:11 CST
+ echo 'Hello world!'
Hello world!
```

本例中有一个命令 date 之前阿铭从未介绍过，这个命令在shell脚本中使用非常频繁，阿铭有必要向您介绍一下它的用法。

命令：date

阿铭举几个比较实用的例子来带您掌握它的用法:

```
[root@localhost sbin]# date +"%Y-%m-%d %H:%M:%S"
2013-05-24 19:41:01
```

date在脚本中最常用的几个用法：

data +%Y 以四位数字格式打印年份

date +%y 以两位数字格式打印年份

date +%m 月份

date +%d 日期

date +%H 小时

date +%M 分钟

date +%S 秒

date +%w 星期，如果结果显示0 则表示周日

有时在脚本中会用到一天前的日期:

```
[root@localhost sbin]# date -d "-1 day" +%d
23
```

或者一小时前:

```
[root@localhost sbin]# date -d "-1 hour" +%H
18
```

甚至1分钟前:

```
[root@localhost sbin]# date -d "-1 min" +%M
50
```

16.2 shell脚本中的变量

在shell脚本中使用变量显得我们的脚本更加专业更像是一门语言，变量的作用当然不是为了专业。如果您写了一个长达1000行的shell脚本，并且脚本中出现了某一个命令或者路径几百次。突然您觉得路径不对想换一下，那岂不是要更改几百次？您固然可以使用批量替换的命令，但也是很麻烦，并且脚本显得臃肿了很多。变量的作用就是用来解决这个问题的。

```
[root@localhost sbin]# cat variable.sh
#! /bin/bash

## In this script we will use variables.
## Writen by Aming 2013-05-24.

d=`date +%H:%M:%S`
echo "The script begin at $d."
echo "Now we'll sleep 2 seconds."
sleep 2
d1=`date +%H:%M:%S`
echo "The script end at $d1."
```

脚本中使用到了反引号，您是否还记得它的作用？ `d' 和 `d1' 在脚本中作为变量出现，定义变量的格式为 变量名=变量的值 当在脚本中引用变量时需要加上 `$' 符号，这跟前面讲的在shell中自定义变量是一致的。下面看看脚本执行结果吧:

```
[root@localhost sbin]# sh variable.sh
The script begin at 20:16:57.
Now we'll sleep 2 seconds.
The script end at 20:16:59.
```

下面我们用shell计算两个数的和:

```
[root@localhost sbin]# cat sum.sh
#! /bin/bash

## For get the sum of tow numbers.
## Aming 2013-05-24.

a=1
b=2
sum=$[$a+$b]
```

```
echo "$a+$b=$sum"
```

数学计算要用[ ]括起来并且外头要带一个 `$' 脚本结果为:

```
[root@localhost sbin]# sh sum.sh
1+2=3
```

Shell脚本还可以和用户交互:

```
[root@localhost sbin]# cat read.sh
#! /bin/bash

## Using 'read' in shell script.
## Aming 2013-05-24.

read -p "Please input a number: " x
read -p "Please input another number: " y
sum=$[$x+$y]
echo "The sum of the two numbers is: $sum"
```

read 命令就是用在这样的地方，用于和用户交互，把用户输入的字符串作为变量值。脚本执行过程如下:

```
[root@localhost sbin]# sh read.sh
Please input a number: 2
Please input another number: 10
The sum of the two numbers is: 12
```

我们不妨加上 -x 选项再来看看这个执行过程:

```
[root@localhost sbin]# sh -x read.sh
+ read -p 'Please input a number: ' x
Please input a number: 22
+ read -p 'Please input another number: ' y
Please input another number: 13
+ sum=35
+ echo 'The sum of the two numbers is: 35'
The sum of the two numbers is: 35
```

有时候我们会用到这样的命令 `/etc/init.d/iptables restart` 前面的/etc/init.d/iptables文件其实就是一个shell脚本，为什么后面可以跟一个 ``restart''? 这里就涉及到了shell脚本的预设变量。实际上，shell脚本在执行的时候后边是可以跟参数的，而且还可以跟多个。

```
[root@localhost sbin]# cat option.sh
#! /bin/bash

sum=$[$1+$2]
echo "sum=$sum"
```

执行结果为:

```
[root@localhost sbin]# sh -x option.sh 1 2
+ sum=3
+ echo sum=3
sum=3
```

在脚本中，您会不会奇怪，哪里来的$1和$2，这其实就是shell脚本的预设变量，其中$1的值就是在执行的时候输入的1，而$2的值就是执行的时候输入的$2，当然一个shell脚本的预设变量是没有限制的，这回您明白了吧。另外还有一个$0，不过它代表的是脚本本身的名字。不妨把脚本修改一下:

```
[root@localhost sbin]# cat option.sh
#! /bin/bash

echo "$1 $2 $0"
```

执行结果:

```
[root@localhost sbin]# sh option.sh  1 2
1 2 option.sh
```

16.3 shell脚本中的逻辑判断

如果您学过C或者其他语言，相信您不会对 if 陌生，在shell脚本中我们同样可以使用 if 逻辑判断。在shell中if判断的基本语法为：

1）不带else

```
if  判断语句; then
    command
fi
```

例如:

```
[root@localhost sbin]# cat if1.sh
#! /bin/bash

read -p "Please input your score: " a
if ((a<60)); then
    echo "You didn't pass the exam."
fi
```

在if1.sh中出现了 ((a<60)) 这样的形式，这是shell脚本中特有的格式，用一个小括号或者不用都会报错，请记住这个格式。执行结果为:

```
[root@localhost sbin]# sh if1.sh
Please input your score: 90
[root@localhost sbin]# sh if1.sh
Please input your score: 33
You didn't pass the exam.
```

2）带有else

```
if  判断语句  ; then
    command
else
    command
fi
```

例如:

```
[root@localhost sbin]# cat if2.sh
#! /bin/bash

read -p "Please input your score: " a
if ((a<60)); then
     echo "You didn't pass the exam."
else
```

```
     echo "Good! You passed the exam."
fi
```

执行结果:

```
[root@localhost sbin]# sh if2.sh
Please input your score: 80
Good! You passed the exam.
[root@localhost sbin]# sh if2.sh
Please input your score: 25
You didn't pass the exam.
```

和上一例唯一区别的地方是，如果输入大于等于60的数字会有所提示。

3）带有elif

```
if  判断语句一  ; then
    command
elif  判断语句二; then
    command
else
    command
fi
```

例如:

```
[root@localhost sbin]# cat if3.sh
#! /bin/bash

 read -p "Please input your score: " a
 if ((a<60)); then
         echo "You didn't pass the exam."
 elif ((a>=60)) && ((a<85)); then
         echo "Good! You pass the exam."
 else
         echo "very good! Your socre is very high!"
 fi
```

这里的 && 表示 ``并且'' 的意思，当然也可以使用 || 表示 ``或者'' 执行结果为:

```
[root@localhost sbin]# sh if3.sh
Please input your score: 90
very good! Your socre is very high!
[root@localhost sbin]# sh if3.sh
Please input your score: 60
Good! You pass the exam.
```

以上只是简单的介绍了if语句的结构。在判断数值大小除了可以用 (( )) 的形式外，还可以使用 [ ] 但是就不能使用>, < , = 这样的符号了，要使用 -lt （小于），-gt （大于），-le （小于等于），-ge （大于等于），-eq （等于），-ne （不等于）。下面阿铭就以命令行的形式简单比较一下，不再写shell脚本了。

```
[root@localhost sbin]# a=10; if [ $a -lt 5 ]; then echo ok; fi
[root@localhost sbin]# a=10; if [ $a -gt 5 ]; then echo ok; fi
ok
[root@localhost sbin]# a=10; if [ $a -ge 10 ]; then echo ok; fi
ok
[root@localhost sbin]# a=10; if [ $a -eq 10 ]; then echo ok; fi
ok
[root@localhost sbin]# a=10; if [ $a -ne 10 ]; then echo ok; fi
```

再看看if中使用 && 和 ||的情况:

```
[root@localhost sbin]# a=10; if [ $a -lt 1 ] || [ $a -gt 5 ]; then echo ok; fi
ok
[root@localhost sbin]# a=10; if [ $a -gt 1 ] || [ $a -lt 10 ]; then echo ok; fi
ok
```

shell 脚本中if还经常判断关于档案属性，比如判断是普通文件还是目录，判断文件是否有读写执行权限等。常用的也就几个选项：

-e ：判断文件或目录是否存在

-d ：判断是不是目录，并是否存在

-f ：判断是否是普通文件，并存在

-r ：判断文档是否有读权限

-w ：判断是否有写权限

-x ：判断是否可执行

使用if判断时，具体格式为：

```
if [ -e filename ] ; then
```

例子:

```
[root@localhost sbin]# if [ -d /home/ ]; then echo ok; fi
ok
[root@localhost sbin]# if [ -f /home/ ]; then echo ok; fi
```

因为 /home/ 为目录为非文件，所以并不会显示 ``ok'' .

```
[root@localhost sbin]# if [ -f /root/test.txt ]; then echo ok; fi
ok
[root@localhost sbin]# if [ -r /root/test.txt ]; then echo ok; fi
ok
[root@localhost sbin]# if [ -w /root/test.txt ]; then echo ok; fi
ok
[root@localhost sbin]# if [ -x /root/test.txt ]; then echo ok; fi
[root@localhost sbin]# if [ -e /root/test1.txt ]; then echo ok; fi
```

在shell 脚本中，除了用if来判断逻辑外，还有一种常用的方式，那就是case了。具体格式为:

```
case  变量  in
value1)
          command
          ;;
value2)
          command
          ;;
value3)
          command
          ;;
*)
          command
          ;;
esac
```

上面的结构中，不限制value的个数， * 则代表除了上面的value外的其他值。下面阿铭写一个判断输入数值是奇数或者偶数的脚本:

```
[root@localhost sbin]# cat case.sh
#! /bin/bash

read -p "Input a number: " n
a=$[$n%2]
case $a in

    1)
        echo "The number is odd."
        ;;
    0)
        echo "The number is even."
        ;;
    *)
        echo "It's not a number!"
        ;;
esac
```

$a 的值或为1或为0，执行结果为:

```
[root@localhost sbin]# sh case.sh
Input a number: 100
The number is even.
[root@localhost sbin]# sh case.sh
Input a number: 101
The number is odd.
```

case脚本常用于编写系统服务的启动脚本，例如/etc/init.d/iptables中就用到了，您不妨去查看一下。

16.4 shell脚本中的循环

Shell脚本中也算是一门简易的编程语言了，当然循环是不能缺少的。常用到的循环有for循环和while循环。下面就分别介绍一下两种循环的结构。

1. for循环

```
[root@localhost sbin]# cat for.sh
#!  /bin/bash

for i in `seq 1 5`; do
    echo $i
done
```

脚本中的 seq 1 5 表示从1到5的一个序列。您可以直接运行这个命令试下。脚本执行结果为:

```
[root@localhost sbin]# sh for.sh
1
2
3
4
5
```

通过这个脚本就可以看到for循环的基本结构:

```
for 变量名 in 循环的条件； do
     command
done
```

这里的 ``循环的条件'' 可以写成一组字符串或者数字(用1个或者多个空格隔开), 也可以是一条命令的执行结果:

```
[root@localhost sbin]# for i in 1 2 3 a b; do echo $i; done
1
2
3
a
b
```

也可以写引用系统命令的执行结果，就像那个 seq 1 5 但是需要用反引号括起来:

```
[root@localhost sbin]# for file in `ls`; do echo $file; done
case.sh
first.sh
for.sh
if1.sh
if2.sh
if3.sh
option.sh
read.sh
sum.sh
variable.sh
```

2. while循环

```
[root@localhost sbin]# cat while.sh
#! /bin/bash

a=5
while [ $a -ge 1 ]; do
    echo $a
    a=$[$a-1]
done
```

while 循环格式也很简单:

```
while  条件; do

          command
done
```

上例脚本的执行结果为:

```
[root@localhost sbin]# sh while.sh
5
4
3
2
1
```

另外您可以把循环条件拿一个冒号替代，这样可以做到死循环，阿铭常常这样写监控脚本:

```
while :; do
    command
    sleep 3
done
```

16.5 shell脚本中的函数

如果您学过开发，肯定知道函数的作用。如果您是刚刚接触到这个概念的话，也没有关系，其实很好理解的。函数就是把一段代码整理到了一个小单元中，并给这个小单元起一个名字，当用到这段代码时直接调用这个小单元的名字即可。有时候脚本中的某段代总是重复使用，如果写成函数，每次用到时直接用函数名代替即可，这样就节省了时间还节省了空间。

下面阿铭写一个简单的带有函数功能的shell脚本:

```
[root@localhost sbin]# cat func.sh
#! /bin/bash

function sum()
{
    sum=$[$1+$2]
    echo $sum
}

sum $1 $2
```

执行结果如下:

```
[root@localhost sbin]# sh func.sh 1 2
3
```

func.sh中的 sum() 为自定义的函数，在shell脚本函数的格式为:

```
function 函数名() {

command

}
```

有一点阿铭要提醒您一下，在shell脚本中，函数一定要写在最前面，不能出现在中间或者最后，因为函数是要被调用的，如果还没有出现就被调用，肯定是会出错的。

16.6 shll脚本练习题

Shell脚本大体上就介绍这么多了，阿铭所举的例子都是最基础的，所以即使您把所有例子完全掌握也不代表您的shell脚本编写能力有多么好。所以剩下的日子里请您尽量要多练习，多写脚本，写的脚本越多，您的shell脚本能力就越强。希望您能够找专门介绍shell脚本的书籍深入的去研究一下它。随后阿铭将留几个shell脚本的练习题，您最好不要偷懒。

1. 编写shell脚本，计算1-100的和；
2. 编写shell脚本，要求输入一个数字，然后计算出从1到输入数字的和，要求，如果输入的数字小于1，则重新输入，直到输入正确的数字为止；
3. 编写shell脚本，把/root/目录下的所有目录（只需要一级）拷贝到/tmp/目录下；
4. 编写shell脚本，批量建立用户user_00, user_01, ... user_100并且所有用户同属于users组；
5. 编写shell脚本，截取文件test.log中包含关键词 `abc' 的行中的第一列（假设分隔符为 ''':'' ），然后把截取的数字排序（假设第一列为数字），然后打印出重复次数超过10次的列；
6. 编写shell脚本，判断输入的IP是否正确（IP的规则是，n1.n2.n3.n4，其中1<n1<255，0<n2<255，0<n3<255，0<n4<255）。

习题答案:

```
1. #! /bin/bash

sum=0

for i in `seq 1 100`; do
        sum=$[$i+$sum]
done

echo $sum

2. #! /bin/bash

n=0
while [ $n -lt "1" ]; do
        read -p "Please input a number, it must greater than "1":" n
done

sum=0

for i in `seq 1 $n`; do
        sum=$[$i+$sum]
done

echo $sum


3. #! /bin/bash

cd /root
for f in `ls `; do
        if [ -d $f ] ; then
                cp -r $f /tmp/
        fi
done



4. #! /bin/bash

groupadd users

for i in `seq 0 9`; do
        useradd -g users user_0$i
done



for j in `seq 10 100`; do
        useradd -g users user_$j
done



5. #! /bin/bash

awk -F':' '$0~/abc/ {print $1}' test.log >/tmp/n.txt
sort -n n.txt |uniq -c |sort -n >/tmp/n2.txt
```

```
awk '$1>10 {print $2}' /tmp/n2.txt



6. #! /bin/bash

checkip() {

        if echo $1 |egrep -q '^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$' ; then
                a=`echo $1 | awk -F. '{print $1}'`
                b=`echo $1 | awk -F. '{print $2}'`
                c=`echo $1 | awk -F. '{print $3}'`
                d=`echo $1 | awk -F. '{print $4}'`

                for n in $a $b $c $d; do
                        if [ $n -ge 255 ] || [ $n -le 0 ]; then
                                echo "the number of the IP should less than 255 and greate than 0"
                                return 2
                        fi
                done
        else

                echo "The IP you input is something wrong, the format is like 192.168.100.1"
                return 1
        fi

}



rs=1
while [ $rs -gt 0 ]; do
    read -p  "Please input the ip:" ip
    checkip $ip
    rs=`echo $?`
done

echo "The IP is right!"
```

阿铭建议您最好再扩展学习一下: http://www.lishiming.net/thread-5439-1-1.html

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

SIXTEEN

第十六章 LINUX系统日常管理

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

阿铭在前面介绍的内容都为linux系统基础类的，如果您现在把前面的内容全部很好的掌握了，那最好了。不过阿铭要说的是，即使您完全掌握了，您现在还是不能作为一名合格的linux系统管理员的，毕竟系统管理员要会做的事情太多了。本章以及后面章节阿铭会陆续教给您作为linux系统管理员所必备的知识。只要您熟练掌握那绝对可以胜任一个最初级的管理员职位，不过只是初级的，因为您还需要在日常的管理工作中获得成长。

17.1 监控系统的状态

1. w查看当前系统的负载

```
[root@localhost sbin]# w
 15:23:46 up  3:34,  2 users,  load average: 0.03, 0.05, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
root     tty1     -                12:26    2:55m  0.11s  0.11s -bash
root     pts/0    10.72.137.53     12:28    1:17m  1:32   1:32  -bash
```

相信所有的linux管理员最常用的命令就是这个 w 了，该命令显示的信息还是蛮丰富的。第一行从左面开始显示的信息依次为：时间，系统运行时间，登录用户数，平均负载。第二行开始以及下面所有的行，告诉我们的信息是，当前登录的都有哪些用户，以及他们是从哪里登录的等等。其实，在这些信息当中，阿铭认为我们最应该关注的应该是第一行中的 `load average:' 后面的三个数值。

第一个数值表示1分钟内系统的平均负载值；第二个数值表示5分钟内系统的平均负载值；第三个数值表示15分钟系统的平均负载值。这个值的意义是，单位时间段内CPU活动进程数。当然这个值越大就说明您的服务器压力越大。一般情况下这个值只要不超过服务器的cpu数量就没有关系，如果服务器cpu数量为8，那么这个值若小于8，就说明当前服务器没有压力，否则就要关注一下了。到这里您肯定会问，如何查看服务器有几个cpu？

```
[root@localhost sbin]# cat /proc/cpuinfo
processor       : 0
vendor_id       : AuthenticAMD
cpu family      : 16
model           : 6
model name      : AMD Phenom(tm) II N660 Dual-Core Processor
stepping        : 3
cpu MHz         : 3000.000
cache size      : 1024 KB
fdiv_bug        : no
hlt_bug         : no
f00f_bug        : no
```

```
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 5
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 sysc
bogomips        : 6000.00
clflush size    : 64
cache_alignment : 64
address sizes   : 36 bits physical, 48 bits virtual
power management: ts ttp tm stc 100mhzsteps hwpstate
```

`/proc/cpuinfo' 这个文件记录了cpu的详细信息。目前市面上的服务器通常都是2颗4核cpu，在linux看来，它就是8个cpu。查看这个文件时则会显示8段类似的信息，而最后一段信息中processor：后面跟的是 `7' 所以查看当前系统有几个cpu，我们可以使用这个命令： grep -c 'processor' /proc/cpuinfo 而如何看几颗物理cpu呢，需要查看关键字 ``physical id'', 由于阿铭的虚拟机只有一个cpu所以并未显示关于``physical id'' 的信息。

2. vmstat 监控系统的状态

```
[root@localhost ~]# vmstat
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0    512   5392  37460 213276    0    0   100   549  153   59  1  4 93  1  0
```

上面讲的 w 查看的是系统整体上的负载，通过看那个数值可以知道当前系统有没有压力，但是具体是哪里（CPU, 内存，磁盘等）有压力就无法判断了。通过 vmstat 就可以知道具体是哪里有压力。vmstat命令打印的结果共分为6部分：procs, memory, swap, io, system, cpu. 请重点关注一下r b si so bi bo几列。

1）procs 显示进程相关信息

r ：表示运行和等待cpu时间片的进程数，如果长期大于服务器cpu的个数，则说明cpu不够用了；

b ：表示等待资源的进程数，比如等待I/O, 内存等，这列的值如果长时间大于1，则需要关注一下了；

2）memory 内存相关信息

swpd ：表示切换到交换分区中的内存数量 ；

free ：当前空闲的内存数量；

buff ：缓冲大小，（即将写入磁盘的）；

cache ：缓存大小，（从磁盘中读取的）；

3）swap 内存交换情况

si ：由交换区写入到内存的数据量；

so ：由内存写入到交换区的数据量；

4）io 磁盘使用情况

bi ：从块设备读取数据的量（读磁盘）；

bo： 从块设备写入数据的量（写磁盘）；

5）system 显示采集间隔内发生的中断次数

in ：表示在某一时间间隔中观测到的每秒设备中断数；

cs ：表示每秒产生的上下文切换次数；

6）CPU 显示cpu的使用状态

us ：显示了用户下所花费 cpu 时间的百分比；

sy ：显示系统花费cpu时间百分比；

id ：表示cpu处于空闲状态的时间百分比；

wa ：表示I/O等待所占用cpu时间百分比；

st ：表示被偷走的cpu所占百分比（一般都为0，不用关注）；

以上所介绍的各个参数中，阿铭经常会关注r列，b列，和wa列，三列代表的含义在上边说得已经很清楚。IO部分的bi以及bo也是要经常参考的对象。如果磁盘io压力很大时，这两列的数值会比较高。另外当si, so两列的数值比较高，并且在不断变化时，说明内存不够了，内存中的数据频繁交换到交换分区中，这往往对系统性能影响极大。

我们使用 vmstat 查看系统状态的时候，通常都是使用这样的形式来看的:

```
[root@localhost ~]# vmstat 1 5
```

或者:

```
[root@localhost ~]# vmstat 1
```

前面表示，每隔一秒钟打印一次状态，共打印5次，而后面的表示每隔1秒打印一次状态，一直打印，除非我们按 Ctrl + c 结束

3. top 显示进程所占系统资源

```
[root@localhost ~]# top
top - 16:31:49 up  4:42,  3 users,  load average: 0.02, 0.05, 0.00
Tasks:  74 total,   1 running,  73 sleeping,   0 stopped,   0 zombie
Cpu(s):  1.4%us,  7.8%sy,  0.0%ni, 89.2%id,  1.0%wa,  0.3%hi,  0.3%si,  0.0%st
Mem:    326616k total,   321172k used,     5444k free,    23664k buffers
Swap:  2097144k total,      588k used,  2096556k free,   227416k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
11194 root      20   0 43936  36m 1128 S  3.8 11.3   2:38.95 perl
 5373 root      20   0  2572 1072  860 R  0.6  0.3   0:00.05 top
24160 root      20   0 12412 2124 1376 S  0.3  0.7   0:01.12 sshd
    1 root      20   0  2900  800  652 S  0.0  0.2   0:01.52 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
```

这个命令用于动态监控进程所占系统资源，每隔3秒变一次。这个命令的特点是把占用系统资源（CPU，内存，磁盘IO等）最高的进程放到最前面。top命令打印出了很多信息，包括系统负载（loadaverage）、进程数（Tasks）、cpu使用情况、内存使用情况以及交换分区使用情况。其实上面这些内容可以通过其他命令来查看，所以用top重点查看的还是下面的进程使用系统资源详细状况。这部分东西反映的东西还是比较多的，不过需要您关注的也就是几项：%CPU, %MEM, COMMAND 这些项目所代表的意义，不用阿铭介绍相信您也能看懂吧，RES 这一项为进程所占内存大小，而 %MEM 为使用内存百分比。在 top 状态下，按 ``shift + m'', 可以按照内存使用大小排序。按数字 `1' 可以列出各颗cpu的使用状态。

另外，阿铭经常用的一个命令 top -bn1 它表示非动态打印系统资源使用情况，可以用在shell脚本中:

```
[root@localhost ~]# top -bn1
top - 16:44:12 up  4:54,  3 users,  load average: 0.54, 0.18, 0.05
Tasks:  78 total,   1 running,  77 sleeping,   0 stopped,   0 zombie
Cpu(s):  1.4%us,  3.3%sy,  0.0%ni, 93.3%id,  1.4%wa,  0.1%hi,  0.5%si,  0.0%st
Mem:    326616k total,   318672k used,     7944k free,    62704k buffers
Swap:  2097144k total,      588k used,  2096556k free,   177848k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 7236 root      20   0  2936 1220  624 D  7.8  0.4   0:03.22 ls
```

```
 7237 root      20   0  2568  956  760 R  1.9  0.3   0:00.03 top
    1 root      20   0  2900  800  652 S  0.0  0.2   0:01.52 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   0:11.08 ksoftirqd/0
    5 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:00.94 watchdog/0
    7 root      20   0     0    0    0 S  0.0  0.0   0:04.38 events/0
```

和 top 命令唯一的区别就是，它一次性全部把所有信息输出出来而非动态显示。

4. sar监控系统状态

sar 命令很强大，它可以监控系统所有资源状态，比如平均负载、网卡流量、磁盘状态、内存使用等等。它不同于其他系统状态监控工具的地方在于，它可以打印历史信息，可以显示当天从零点开始到当前时刻的系统状态信息。如果您系统没有安装这个命令，请使用 yum install -y sysstat 命令安装。初次使用sar命令会报错，那是因为sar工具还没有生成相应的数据库文件（时时监控就不会了，因为不用去查询那个库文件）。它的数据库文件在 ``/var/log/sa/'' 目录下，默认保存一个月。因为这个命令太过复杂，所以阿铭只介绍几个。

1）查看网卡流量 sar -n DEV

```
[root@localhost ~]# sar -n DEV
Linux 2.6.32-358.el6.i686 (localhost.localdomain)       2013年05月25日  _i686_ (1 CPU)

00时00分01秒     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
00时10分01秒        lo      0.00      0.00      0.00      0.00      0.00   0.00      0.00
00时10分01秒      eth0     31.94      0.09      3.89      0.02      0.00   0.00      0.00
00时20分01秒        lo      0.00      0.00      0.00      0.00      0.00   0.00      0.00
00时20分01秒      eth0     32.40      0.04      3.96      0.01      0.00   0.00      0.00
00时30分01秒        lo      0.00      0.00      0.00      0.00      0.00   0.00      0.00
00时30分01秒      eth0     31.18      0.06      3.76      0.02      0.00   0.00      0.00
```

阿铭并没有把全部信息贴出来，因为太多了。 IFACE这列表示设备名称，rxpck/s 表示每秒进入收取的包的数量，txpck/s 表示每秒发送出去的包的数量，rxbyt/s 表示每秒收取的数据量（单位Byte），txbyt/s表示每秒发送的数据量。后面几列不需要关注。如果有一天您所管理的服务器丢包非常严重，那么您就应该看一看这个网卡流量是否异常了，如果rxpck/s 那一列的数值大于4000，或者rxbyt/s那列大于5,000,000则很有可能是被攻击了，正常的服务器网卡流量不会高于这么多，除非是您自己在拷贝数据。上面的命令是查看网卡流量历史的，如何时时查看网卡流量呢？

```
[root@localhost ~]# sar -n DEV 1 5
Linux 2.6.32-358.el6.i686 (localhost.localdomain)       2013年05月25日  _i686_ (1 CPU)

16时46分55秒     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s    txcmp/s  rxmcst/s
16时46分56秒        lo      0.00      0.00      0.00      0.00      0.00    0.00      0.00
16时46分56秒      eth0     36.36      0.00      4.16      0.00      0.00    0.00      0.00

16时46分56秒     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s    txcmp/s  rxmcst/s
16时46分57秒        lo      0.00      0.00      0.00      0.00      0.00    0.00      0.00
16时46分57秒      eth0     69.39      1.02     10.66      0.39      0.00    0.00      0.00

16时46分57秒     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s    txcmp/s  rxmcst/s
16时46分58秒        lo      0.00      0.00      0.00      0.00      0.00    0.00      0.00
16时46分58秒      eth0     42.00      1.00      7.56      0.38      0.00    0.00      0.00

16时46分58秒     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s    txcmp/s  rxmcst/s
16时46分59秒        lo      0.00      0.00      0.00      0.00      0.00    0.00      0.00
16时46分59秒      eth0     51.52      1.01      7.73      0.39      0.00    0.00      0.00
```

```
16时46分59秒     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s    txcmp/s  rxmcst/s
16时47分00秒        lo      0.00      0.00      0.00      0.00      0.00    0.00      0.00
16时47分00秒      eth0     60.00      1.00      5.51      0.38      0.00    0.00      0.00

平均时间:     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s    txcmp/s  rxmcst/s
平均时间:        lo      0.00      0.00      0.00      0.00      0.00    0.00      0.00
平均时间:      eth0     51.81      0.81      7.11      0.31      0.00    0.00      0.00
```

另外也可以查看某一天的网卡流量历史，使用-f选项，后面跟文件名，如果您的系统格式Redhat或者CentOS那么sar的库文件一定是在/var/log/sa/目录下的。：

```
[root@localhost ~]# sar -n DEV -f /var/log/sa/sa24
Linux 2.6.32-358.el6.i686 (localhost.localdomain)       2013年05月24日  _i686_ (1 CPU)

10时49分36秒       LINUX RESTART

10时50分01秒     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
11时00分01秒        lo      0.00      0.00      0.00      0.00      0.00   0.00      0.00
11时00分01秒      eth0     58.96      0.02      7.87      0.01      0.00   0.00      0.00
11时10分01秒        lo      0.00      0.00      0.00      0.00      0.00   0.00      0.00
11时10分01秒      eth0     61.36      0.34      8.29      0.05      0.00   0.00      0.00
11时20分01秒        lo      0.00      0.00      0.00      0.00      0.00   0.00      0.00
11时20分01秒      eth0     57.17      0.22      7.65      0.03      0.00   0.00      0.00
11时30分01秒        lo      0.00      0.00      0.00      0.00      0.00   0.00      0.00
11时30分01秒      eth0     54.99      0.05      7.03      0.01      0.00   0.00      0.00
11时40分01秒        lo      0.00      0.00      0.00      0.00      0.00   0.00      0.00
```

2）查看历史负载 sar -q

```
[root@localhost ~]# sar -q
Linux 2.6.32-358.el6.i686 (localhost.localdomain)       2013年05月25日  _i686_ (1 CPU)

00时00分01秒   runq-sz  plist-sz   ldavg-1   ldavg-5  ldavg-15
00时10分01秒         0       142      0.00      0.00      0.00
00时20分01秒         0       143      0.00      0.00      0.00
00时30分01秒         0       143      0.00      0.01      0.00
00时40分01秒         0       143      0.05      0.01      0.00
00时50分01秒         0       143      0.00      0.00      0.00
01时00分01秒         0       143      0.00      0.00      0.00
01时10分01秒         0       143      0.08      0.03      0.00
```

这个命令有助于我们查看服务器在过去的某个时间的负载状况。关于sar的介绍阿铭不愿写太多，毕竟介绍太多会给您带来更多的压力，其实阿铭介绍这个命令的目的只是让您学会查看网卡流量（这是非常有用的）。如果您很感兴趣那就man一下吧，它的用法太多了。

5. free查看内存使用状况

```
[root@localhost ~]# free
             total       used       free     shared    buffers     cached
Mem:        326616     137332     189284          0      34480      73336
-/+ buffers/cache:      29516     297100
Swap:      2097144       1144    2096000
```

只需要敲一个 free 然后回车就可以当前系统的总内存大小以及使用内存的情况。从上例中可看到当前系统内存总大小为326616（单位是k）已经使用137332, 剩余189284. 其实真正剩余并不是这个189284, 而是第二行的297100, 真正使用的也是第二行的29516, 这是因为系统初始化时，就已经分配出很大一部分内存给缓存，这部分缓存用来随时提供给程序使用，如果程序不用，那这部分内存就空闲。所以，查看内存使用多少，剩余多少请看第二行的数据。另外我们还可以加-m 或者-g选项分别以M或G为单位打印内存使用状况:

```
[root@localhost ~]# free  -m
             total       used       free     shared    buffers     cached
Mem:           318        135        183          0         34         72
-/+ buffers/cache:         28        290
Swap:         2047          1       2046
```

6. ps 查看系统进程

作为系统管理员，一定要知道您所管理的系统都有那些进程在运行，在windows下只要打开任务管理器即可查看。在linux下呢？其实在上面介绍的top命令就可以，但是不容易看，当然还有专门显示系统进程的命令：

```
[root@localhost ~]# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.2   2900   852 ?        Ss   11:49   0:01 /sbin/init
root         2  0.0  0.0      0     0 ?        S    11:49   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    11:49   0:00 [migration/0]
root         4  0.0  0.0      0     0 ?        S    11:49   0:11 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S    11:49   0:00 [migration/0]
root         6  0.0  0.0      0     0 ?        S    11:49   0:00 [watchdog/0]
root         7  0.0  0.0      0     0 ?        S    11:49   0:04 [events/0]
root         8  0.0  0.0      0     0 ?        S    11:49   0:00 [cgroup]
root         9  0.0  0.0      0     0 ?        S    11:49   0:00 [khelper]
```

阿铭也经常看到有的人喜欢用 ps -elf 大同小异，显示的信息基本上是一样的。ps命令还有更多的用法，阿铭不再做介绍，因为您只要会用这个命令就足够了，请man一下。下面介绍几个参数的意义。

PID ：进程的id，这个id很有用，在linux中内核管理进程就得靠pid来识别和管理某一个程，比如我想终止某一个进程，则用‘kill 进程的pid 有时并不能杀掉，则需要加一个-9选项了 kill -9 进程pid

STAT ：表示进程的状态，进程状态分为以下几种（不要求记住，但要了解）

D 不能中断的进程（通常为IO）

R 正在运行中的进程

S 已经中断的进程，通常情况下，系统中大部分进程都是这个状态

T 已经停止或者暂停的进程，如果我们正在运行一个命令，比如说 sleep 10 如果我们按一下ctrl -z 让他暂停，那么我们用ps查看就会显示T这个状态

W 这个好像是说，从内核2.6xx 以后，表示为没有足够的内存页分配

X 已经死掉的进程（这个好像从来不会出现）

Z 僵尸进程，杀不掉，打不死的垃圾进程，占系统一小点资源，不过没有关系。如果太多，就有问题了。一般不会出现。

< 高优先级进程

N 低优先级进程

L 在内存中被锁了内存分页

s 主进程

l 多线程进程

+ 代表在前台运行的进程

这个ps命令是阿铭在工作中用的非常多的命令之一，所以请记住它吧。关于ps命令的使用，阿铭经常会连同管道符一起使用，用来查看某个进程或者它的数量。

```
[root@localhost ~]# ps aux |grep -c mingetty
6
[root@localhost ~]# ps aux |grep  mingetty
root       952  0.0  0.1   2008   440 tty2     Ss+  11:49   0:00  /sbin/mingetty /dev/tty2
root       954  0.0  0.1   2008   440 tty3     Ss+  11:49   0:00  /sbin/mingetty /dev/tty3
root       956  0.0  0.1   2008   440 tty4     Ss+  11:49   0:00  /sbin/mingetty /dev/tty4
root       958  0.0  0.1   2008   436 tty5     Ss+  11:49   0:00  /sbin/mingetty /dev/tty5
root       960  0.0  0.1   2008   444 tty6     Ss+  11:49   0:00  /sbin/mingetty /dev/tty6
root      8440  0.0  0.2   5984   732 pts/3    S+   17:12   0:00  grep mingetty
```

上面的6不对，需要减掉1，因为使用grep命令时，grep命令本身也算作了一个。

7. netstat 查看网络状况

```
[root@localhost ~]# netstat -lnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address               Foreign Address    State       PID/Program name
tcp        0      0 0.0.0.0:22                  0.0.0.0:*          LISTEN      929/sshd
tcp        0      0 :::80                       :::*               LISTEN      25005/httpd
tcp        0      0 :::22                       :::*               LISTEN      929/sshd
udp        0      0 0.0.0.0:68                  0.0.0.0:*                      1597/dhclient
Active UNIX domain sockets (only servers)
Proto RefCnt Flags       Type       State         I-Node PID/Program name      Path
unix  2      [ ACC ]     STREAM     LISTENING     6783   1/init                @/com/ubuntu/upstart
```

netstat命令用来打印网络连接状况、系统所开放端口、路由表等信息。阿铭最常用的关于netstat的命令就是这个 `netstat -lnp` （打印当前系统启动哪些端口）以及 `netstat -an` （打印网络连接状况）这两个命令非常有用，请一定要记住。

```
[root@localhost ~]# netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address    State
tcp        0      0 0.0.0.0:22                  0.0.0.0:*          LISTEN
tcp        0      0 10.72.137.159:22            10.72.137.53:50507 ESTABLISHED
tcp        0     52 10.72.137.159:22            10.72.137.53:50827 ESTABLISHED
tcp        0      0 :::80                       :::*               LISTEN
tcp        0      0 :::22                       :::*               LISTEN
udp        0      0 0.0.0.0:68                  0.0.0.0:*
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
unix  2      [ ACC ]     STREAM     LISTENING     6783   @/com/ubuntu/upstart
unix  2      [ ]         DGRAM                    6953   @/org/kernel/udev/udevd
unix  3      [ ]         DGRAM                    6973
unix  3      [ ]         DGRAM                    6972
```

如果您所管理的服务器是一台提供web服务（80端口）的服务器，那么您就可以使用 `netstat -an | grep 80` 查看当前连接web服务的有哪些IP了。

抓包工具tcpdump

有时候，也许您会有这样的需求，想看一下某个网卡上都有哪些数据包，尤其是当您初步判定您的服务器上有流量攻击。这时，使用抓包工具来抓一下数据包，就可以知道有哪些IP在攻击您了。

```
[root@localhost ~]# tcpdump -nn -i eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
19:13:56.689147 IP 10.72.137.159.22 > 10.72.137.53.50827: Flags [P.], seq 2793829986:2793830182, ack 1384443306, win 106
19:13:56.691389 IP 10.72.137.159.22 > 10.72.137.53.50827: Flags [P.], seq 196:376, ack 1, win 1067, length 180
19:13:56.691541 IP 10.72.137.53.50827 > 10.72.137.159.22: Flags [.], ack 376, win 16266, length 0
```

```
19:13:56.694499 IP 10.72.137.159.22 > 10.72.137.53.50827: Flags [P.], seq 376:636, ack 1, win 1067, length 260
19:13:56.695659 IP 10.72.137.159.22 > 10.72.137.53.50827: Flags [P.], seq 636:800, ack 1, win 1067, length 164
19:13:56.695793 IP 10.72.137.53.50827 > 10.72.137.159.22: Flags [.], ack 800, win 16160, length 0
19:13:56.698429 IP 10.72.137.159.22 > 10.72.137.53.50827: Flags [P.], seq 800:1060, ack 1, win 1067, length 260
19:13:56.700332 IP 10.72.137.159.22 > 10.72.137.53.50827: Flags [P.], seq 1060:1224, ack 1, win 1067, length 164
19:13:56.700419 IP 10.72.137.53.50827 > 10.72.137.159.22: Flags [.], ack 1224, win 16425, length 0
```

如果没有tcpdump 这个命令，需要用 `yum install -y tcpdump` 命令去安装一下。上例中第三列和第四列显示的信息为哪一个IP+port在连接哪一个IP+port，后面的信息是该数据包的相关信息，如果不懂也没有关系，毕竟我们不是专门搞网络的，而这里需要关注的只是第三列以及第四列。-i 选项后面跟设备名称，如果您想抓eth1网卡的包，后面则要跟eth1.至于-nn选项的作用是让第三列和第四列显示成IP+端口号的形式，如果不加-nn则显示的是主机名+服务名称。

17.2 Linux网络相关

1. ifconfig 查看网卡IP

ifconfig类似与windows的ipconfig，不加任何选项和参数只打印当前网卡的IP相关信息（子网掩码、网关等）在之前的章节中阿铭曾经介绍过它。在windows下设置IP非常简单，然而在命令窗口下如何设置？这就需要去修改配置文件/etc/sysconfig/network-scripts/ifcfg-eth0了，如果是eth1那么配置文件是/etc/sysconfig/network-scripts/ifcfg-eth1.

如果Linux上有多个网卡，而只想重启某一个网卡的话，可以使用这个命令:

```
[root@localhost ~]# ifdown eth0; ifup eth0
```

ifdown 即停掉网卡，ifup即启动网卡。有一点要提醒您的是，如果我们远程登录服务器，当使用ifdown eth0这个命令的时候，很有可能后面的命令ifup eth0不会被运行，这样导致我们断网而无法连接服务器，所以请尽量使用 `service network restart` 这个命令来重启网卡。

2. 给一个网卡设定多个IP

在linux系统中，网卡是可以设定多重IP的，阿铭曾经管理的一台服务器的eth1就设定了5个IP，实在是够变态的。

```
[root@localhost ~]# cd /etc/sysconfig/network-scripts/
[root@localhost network-scripts]# cp ifcfg-eth0 ifcfg-eth0\:1
```

然后编辑ifcfg-eth0:1 这个配置文件，内容如下，一定要注意 DEVICE 这里要写成 ``eth0:1''

```
[root@localhost network-scripts]# cat ifcfg-eth0\:1
DEVICE=eth0:1
HWADDR=00:0C:29:D9:F0:52
TYPE=Ethernet
UUID=a5442526-0329-421d-86cf-8d7f16d01374
ONBOOT=yes
BOOTPROTO=none
IPADDR=192.168.80.5
NETMASK=255.255.255.0
GATEWAY=192.168.80.2
NM_CONTROLLED=yes
```

编辑好后，重启网卡:

```
[root@localhost network-scripts]# ifdown eth0 && ifup eth0
```

之后再查看网卡ip:

```
[root@localhost network-scripts]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:D9:F0:52
          inet addr:10.72.137.159  Bcast:10.72.137.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fed9:f052/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2587605 errors:2 dropped:0 overruns:0 frame:0
          TX packets:773070 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1934306928 (1.8 GiB)  TX bytes:54602387 (52.0 MiB)
          Interrupt:18 Base address:0x1080

eth0:1    Link encap:Ethernet  HWaddr 00:0C:29:D9:F0:52
          inet addr:192.168.80.5  Bcast:192.168.80.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:18 Base address:0x1080

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:39 errors:0 dropped:0 overruns:0 frame:0
          TX packets:39 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:16066 (15.6 KiB)  TX bytes:16066 (15.6 KiB)
```

可以看到多了一个ip.

3. 查看网卡连接状态

```
[root@localhost ~]# mii-tool eth0
SIOCGMIIPHY on 'eth0' failed: Operation not supported
```

如果是在服务器上不会显示成这样的，由于是虚拟机所以显示 ``not supported'', 如果是真机应该显示如下内容:

```
[root@db-dd sphinx]# mii-tool eth0
eth0: negotiated 100baseTx-FD, link ok
```

只要看到 ``link ok'' 就说明网卡为连接状态，如果显示 ``no link'' 说明网卡坏掉了或者没有连接网线。

4. 更改主机名

当装完系统后，默认主机名为localhost，使用hostname就可以知道您的linux的主机名是什么:

```
[root@localhost ~]# hostname
localhost.localdomain
```

同样使用hostname可以更改您的主机名:

```
[root@localhost ~]# hostname Aming
[root@localhost ~]# hostname
Aming
```

下次登录时就会把命令提示符 [root@localhost ~] 中的 localhost 更改成 Aming 不过这样修改只是保存在内存中，下次重启还会变成未改之前的主机名，所以需要您还要去更改相关的配置文件 ``/etc/sysconfig/network''

```
[root@localhost ~]# vim /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=Aming.localdomain
```

5. 设置DNS

DNS是用来解析域名用的，平时我们访问网站都是直接输入一个网址，而dns把这个网址解析到一个IP。关于dns的概念，如果您很陌生的话，那就去网上查一下吧。在linux下面设置dns非常简单，只要把dns地址写到一个配置文件中即可。这个配置文件就是/etc/resolv.conf

```
[root@localhost ~]# vim /etc/resolv.conf
; generated by /sbin/dhclient-script
nameserver 202.106.46.151
```

resolv.conf有它固有的格式，一定要写成 ``nameserver IP'' 的格式，上面那行以 `;' 为开头的行是一行注释，没有实际意义，建议写两个或多个namserver ，默认会用第一个namserver去解析域名，当第一个解析不到时会使用第二个。在linux下面有一个特殊的文件/etc/hosts也能解析域名，不过是需要我们手动在里面添加IP+域名这些内容，它的作用是临时解析某个域名，非常有用。

```
[root@localhost ~]# vim /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.1.111 www.baidu.com
```

保存后，再ping一下 www.baidu.com 就会到 192.168.1.111 了:

```
[root@localhost ~]# ping www.baidu.com
PING www.baidu.com (192.168.1.111) 56(84) bytes of data.
```

/etc/hosts 的格式很简单，每一行作为一条记录，分成两部分，第一部分是IP，第二部分是域名。关于hosts文件，有几点需要您注意：

1）一个IP后面可以跟多个域名，可以是几十个甚至上百个；

2）每行只能有一个IP，也就是说一个域名不能对应多个IP；

3）如果有多行中出现相同的域名（前面IP不一样），会按最前面出现的记录来解析。

17.3 Linux的防火墙

1. selinux

Selinux是Redhat/CentOS系统特有的安全机制。不过因为这个东西限制太多，配置也特别繁琐所以几乎没有人去真正应用它。所以装完系统，我们一般都要把selinux关闭，以免引起不必要的麻烦。关闭selinux的方法为，使 ``SELINUX=disabled'', 默认为 enforcing

```
[root@localhost ~]# vim /etc/selinux/config

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protected,
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

保存该配置文件后，重启机器方可生效，临时关闭selinux的命令为:

```
[root@localhost ~]# setenforce 0
```

我们可以使用 getenforce 命令获得当前selinux的状态:

```
[root@localhost ~]# getenforce
Disabled
```

阿铭的selinux早就关闭了，默认会输出 ``enforcing''，当使用 setenforce 0 这个命令后，再 getenforce 会输出 ``permissive''

2. iptables

Iptables是linux上特有的防火墙机制，其功能非常强大，然而阿铭在日常的管理工作中仅仅用到了一两个应用，这并不代表iptables不重要。作为一个网络管理员，iptables是必要要熟练掌握的。但是作为系统管理员，我们也应该会最基本的iptables操作，认识iptables的基本规则。

CentOS上默认是设有iptables规则的，这个规则虽然很安全，但是对于我们来说没有用，反而会造成某些影响，所以阿铭建议您先清除规则，然后把清除后的规则保存一下:

```
[root@localhost ~]# iptables -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
destination
  400  176K ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
state RELATED,ESTABLISHED
    0     0 ACCEPT     icmp --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
    1    52 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
state NEW tcp dpt:22
    3   234 REJECT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
destination
    0     0 REJECT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT 167 packets, 16963 bytes)
 pkts bytes target     prot opt in     out     source
destination
[root@localhost ~]# iptables -F; /etc/init.d/iptables save
iptables：将防火墙规则保存到 /etc/sysconfig/iptables：     [确定]
```

-nvL 就是查看规则， -F 是把当前规则清除，但这个只是临时的，重启系统或者重启 iptalbes 服务后还会加载已经保存的规则，所以需要使用 /etc/init.d/iptables save 保存一下规则，通过上边的命令输出我们也可以看到，防火墙规则保存在了/etc/sysconfig/iptables 您可以查看一下这个文件。

1）iptalbes的三个表

filter 这个表主要用于过滤包的，是系统预设的表，这个表也是阿铭用的最多的。内建三个链INPUT、OUTPUT以及FORWARD。INPUT作用于进入本机的包；OUTPUT作用于本机送出的包；FORWARD作用于那些跟本机无关的包。

nat 主要用处是网络地址转换，也有三个链。PREROUTING 链的作用是在包刚刚到达防火墙时改变它的目的地址，如果需要的话。OUTPUT链改变本地产生的包的目的地址。POSTROUTING链在包就要离开防火墙之前改变其源地址。该表阿铭用的不多，但有时候会用到。

mangle 这个表主要是用于给数据包打标记，然后根据标记去操作哪些包。这个表几乎不怎么用。除非您想成为一个高级网络工程师，否则您就没有必要花费很多心思在它上面。

2）iptables 基本语法

A. 查看规则以及清除规则

```
[root@localhost ~]# iptables -t nat -nvL
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

-t 后面跟表名，-nvL 即查看该表的规则，其中-n表示不针对IP反解析主机名；-L表示列出的意思；而-v表示列出的信息更加详细。如果不加-t ，则打印filter表的相关信息:

```
[root@localhost ~]# iptables -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

这个和-t filter 打印的信息是一样的。

关于清除规则的命令中，阿铭用的最多就是:

```
[root@localhost ~]# iptables -F
[root@localhost ~]# iptables -Z
```

不加-t默认是针对表filter来操作的，-F 表示把所有规则全部删除；-Z表示把包以及流量计数器置零（这个阿铭认为很有用）。

B. 增加/删除一条规则

```
# iptables -A INPUT -s 10.72.11.12 -p tcp --sport 1234 -d 10.72.137.159 --dport 80 -j DROP
```

这就是增加了一条规则，省略-t所以针对的是filter表。-A 表示增加一条规则，另外还有-I 表示插入一条规则，-D删除一条规则；后面的INPUT即链名称，还可以是OUTPUT或者FORWORD；-s 后跟源地址；-p 协议（tcp, udp, icmp）； --sport/--dport 后跟源端口/目标端口；-d 后跟目的IP（主要针对内网或者外网）；-j 后跟动作（DROP即把包丢掉，REJECT即包拒绝；ACCEPT即允许包）。这样讲可能很乱，那阿铭多举几个例子来帮您理解:

```
[root@localhost ~]# iptables -I INPUT -s 1.1.1.1 -j DROP
```

上例表示：插入一条规则，把来自1.1.1.1的所有数据包丢掉。

```
[root@localhost ~]# iptables -D INPUT -s 1.1.1.1 -j DROP
```

删除刚刚插入的规则。注意要删除一条规则时，必须和插入的规则一致，也就是说，两条iptables命令，除了-I 和-D不一样外，其他地方都一样。

```
[root@localhost ~]# iptables -I INPUT -s 2.2.2.2 -p tcp --dport 80 -j DROP
```

上例表示把来自2.2.2.2 并且是tcp协议到本机的80端口的数据包丢掉。这里要说的是，--dport/--sport 必须要和-p选项一起使用，否则会出错。

```
[root@localhost ~]# iptables -I OUTPUT -p tcp --dport 22 -d 10.0.1.14 -j DROP
```

这条规则表示，把发送到10.0.2.34的22端口的数据包丢掉。

至于FORWORD链的应用阿铭几乎没有用到过，所以不再举例。再总结一下各个选项的作用：

-A/-D ：增加删除一条规则；

-I ：插入一条规则，其实跟-A的效果一样；

-p ：指定协议，可以是tcp，udp或者icmp；

--dport ：跟-p一起使用，指定目标端口；

--sport ：跟-p一起使用，指定源端口；

-s ：指定源IP（可以是一个ip段）；

-d ：指定目的IP（可以是一个ip段）；

-j ：后跟动作，其中ACCEPT表示允许包，DROP表示丢掉包，REJECT表示拒绝包；

-i ：指定网卡（不常用，但有时候能用到）；

```
[root@localhost ~]# iptables -A INPUT -s 192.168.1.0/24 -i eth0 -j ACCEPT
[root@localhost ~]# iptables -nvL |grep '192.168.1.0/24'
    0     0 ACCEPT     all  --  eth0   *       192.168.1.0/24       0.0.0.0/0
```

上例中表示，把来自192.168.1.0/24这个网段的并且作用在eth0上的包放行。有时候您的服务器上iptables过多了，想删除某一条规则时，又不容易掌握当时创建时的规则。其实有一种比较简单的方法:

```
[root@localhost ~]# iptables -nvL --line-numbers
Chain INPUT (policy ACCEPT 133 packets, 9740 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1        0     0 ACCEPT     all  --  eth0   *       192.168.1.0/24       0.0.0.0/0
```

删除某一条规则使用如下命令:

```
[root@localhost ~]# iptables -D INPUT 1
```

-D 后跟链名，然后是规则num，这个num就是查看iptables规则时第一列的值。再次查看刚才的规则，已经没有了:

```
[root@localhost ~]# iptables -nvL --line-numbers
```

iptables还有一个选项经常用到，-P（大写）选项，表示预设策略。用法如下:

```
[root@localhost ~]# iptables -P INPUT DROP
```

-P后面跟链名，策略内容或者为DROP或者为ACCEPT，默认是ACCEPT。注意：如果您在连接远程服务器，千万不要随便敲这个命令，因为一旦您敲完回车您就会断掉。

这个策略一旦设定后，只能使用 iptables -P ACCEPT 才能恢复成原始状态，而不能使用-F参数。下面阿铭针对一个小需求讲述一下这个iptables规则如何设定。

需求：只针对filter表，预设策略INPUT链DROP，其他两个链ACCEPT，然后针对192.168.137.0/24开通22端口，对所有网段开放80端口，对所有网段开放21端口。这个需求不算复杂，但是因为有多条规则，所以最好写成脚本的形式。脚本内容如下:

```
[root@localhost ~]# cat /usr/local/sbin/iptables.sh
#! /bin/bash

ipt="/sbin/iptables"
```

```
$ipt -F
$ipt -P INPUT DROP
$ipt -P OUTPUT ACCEPT
$ipt -P FORWARD ACCEPT
$ipt -A INPUT -s 192.168.137.0/24 -p tcp --dport 22 -j ACCEPT
$ipt -A INPUT -p tcp --dport 80 -j ACCEPT
$ipt -A INPUT -p tcp --dport 21 -j ACCEPT
```

完成脚本的编写后，直接运行 /bin/sh /usr/local/sbin/iptables.sh 即可。如果想开机启动时初始化防火墙规则，则需要在 /etc/rc.d/rc.local 中添加一行 ``/bin/sh /usr/local/sbin/iptables.sh''

```
[root@localhost ~]# sh /usr/local/sbin/iptables.sh
[root@localhost ~]# iptables -nvL
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
   20  1580 ACCEPT     tcp  --  *      *       192.168.137.0/24     0.0.0.0/0           tcp dpt:22
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:80
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:21
```

运行脚本后，查看规则就是这样的，可以看到阿铭的第一条规则中已经有20个包（第一列）被放行过了。

关于icmp的包有一个比较常见的应用:

```
[root@localhost ~]# iptables -I INPUT -p icmp --icmp-type 8 -j DROP
```

--icmp-type 这个选项是要跟-p icmp 一起使用的，后面指定类型编号。这个8指的是能在本机ping通其他机器，而其他机器不能ping通本机。这个有必要记一下。

C. nat表的应用

其实，linux的iptables功能是十分强大的，阿铭曾经的一个老师这样形容linux的网络功能：只有想不到没有做不到！也就是说只要您能够想到的关于网络的应用，linux都能帮您实现。在日常生活中相信您接触过路由器吧，它的功能就是分享上网。本来一根网线过来（其实只有一个公网IP），通过路由器后，路由器分配了一个网段（私网IP），这样连接路由器的多台pc都能连接intnet而远端的设备认为您的IP就是那个连接路由器的公网IP。这个路由器的功能其实就是由linux的iptables实现的，而iptables又是通过nat表作用而实现的这个功能。

至于具体的原理以及过程，阿铭不阐述，请查看相关资料。在这里举一个例子来说明iptables如何实现的这个功能。假设您的机器上有两块网卡eth0和eth1，其中eth0的IP为10.0.2.68 ，eth1的IP为192.168.1.1 。eth0连接了intnet 但eth1没有连接，现在有另一台机器（192.168.1.2）和eth1是互通的，那么如何设置也能够让连接eth1的这台机器能够连接intnet（即能和10.0.2.68互通）？

```
[root@localhost ~]# echo "1" > /proc/sys/net/ipv4/ip_forward
[root@localhost ~]# iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth0 -j MASQUERADE
```

就是这样简单的两条命令就能实现上面的需求。第一个命令涉及到了内核参数相关的配置文件，它的目的是为了打开路由转发功能，否则无法实现我们的应用。第二个命令则是iptables对nat表做了一个IP转发的操作，-o 选项后跟设备名，表示出口的网卡，MASQUERADE表示伪装的意思。 关于nat表，阿铭不想讲太多内容，您只要学会这个路由转发即可。其他的东西交给网络工程师去学习吧，毕竟您将来可是要做linux系统工程师的。

D. 保存以及备份iptalbes规则

刚才在上面的内容中阿铭也提到了，咱们设定的防火墙规则只是保存在内存中，并没有保存到某一个文件中，也就说当系统重启后以前设定的规则就没有了，所以设定好规则后要先保存一下。

```
[root@localhost ~]# service iptables save
iptables：将防火墙规则保存到 /etc/sysconfig/iptables：     [确定]
```

它会提示防火墙规则保存在了/etc/sysconfig/iptables文件内，这个文件就是iptables的配置文件了。所以日后，如果您遇到备份防火墙规则的任务，其实就是要拷贝一份这个文件的副本。

有时，我们会需要把防火墙所有规则都清除，使用 iptables -F 命令虽然可以，但是最好的办法是把防火墙服务停止:

```
[root@localhost ~]# service iptables stop
iptables：清除防火墙规则：                                [确定]
iptables：将链设置为政策 ACCEPT：nat filter               [确定]
iptables：正在卸载模块：                                  [确定]
```

这样防火墙就失效了，但是一旦重新设定规则后（哪怕只有一条），防火墙服务会自动开启。下面阿铭介绍给您一个用来备份防火墙规则的命令:

```
[root@localhost ~]# sh /usr/local/sbin/iptables.sh
[root@localhost ~]# iptables-save > myipt.rule
[root@localhost ~]# cat myipt.rule
# Generated by iptables-save v1.4.7 on Sat Jun  1 18:14:03 2013
*filter
:INPUT DROP [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [50:4528]
-A INPUT -s 192.168.137.0/24 -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 21 -j ACCEPT
COMMIT
# Completed on Sat Jun  1 18:14:03 2013
```

先执行一下刚才我们写的iptables脚本，使用 iptables-save 命令重定向到一个文件里。要想恢复这些规则使用下面的命令即可:

```
[root@localhost ~]# iptables-restore < myipt.rule
```

17.4 linux系统的任务计划

这部分内容太重要了，其实大部分系统管理工作都是通过定期自动执行某一个脚本来完成的，那么如何定期执行某一个脚本呢？这就要借助linux的cron功能了。

关于cron任务计划功能的操作都是通过crontab这个命令来完成的。其中常用的选项有：

-u ：指定某个用户，不加-u选项则为当前用户；

-e ：制定计划任务；

-l ：列出计划任务；

-r ：删除计划任务。

阿铭要创建第一个任务计划了:

```
[root@localhost ~]# crontab -e
no crontab for root - using an empty one
```

使用 crontab -e 来进行编写任务计划，这实际上是使用vim工具打开了crontab的配置文件，我们写下如下内容:

```
01 10 05 06 3 echo "ok" > /root/cron.log
```

每个字段的数字分表表示什么呢？从左到右，依次为：分，时，日，月，周，命令行。而上面的例子的含义是：在6月5日（这一天必须是星期3）的10点01分执行命令 `echo "ok" > /root/cron.log`

crontab -e 实际上是打开了 ``/var/spool/cron/username'' （如果是root则打开的是/var/spool/cron/root）这个文件。使用的是vim编辑器，所以要保存的话则在命令模式下输入:wq即可。但是，您千万不要直接去编辑那个文件，因为可能会出错，所以一定要使用 `crontab -e` 来编辑。查看已经设定的任务计划使用 `crontab -l` 命令:

```
[root@localhost ~]# crontab -l
01 10 05 06 3 echo "ok" > /root/cron.log
```

删除计划任务要用 `crontab -r`

```
[root@localhost ~]# crontab -r
[root@localhost ~]# crontab -l
no crontab for root
```

cron的内容不算太难，但是需要您牢固掌握，阿铭给出一些练习题，帮助您熟悉这个cron的应用。

1. 每天凌晨1点20分清除/var/log/slow.log这个文件
2. 每周日3点执行 ``/bin/sh /usr/local/sbin/backup.sh''
3. 每月14号4点10分执行 ``/bin/sh /usr/local/sbin/backup_month.sh''
4. 每隔8小时执行 ``ntpdate time.windows.com''
5. 每天的1点，12点，18点执行 ``/bin/sh /usr/local/sbin/test.sh''
6. 每天的9点到18点执行 ``/bin/sh /usr/local/sbin/test2.sh''

习题答案:

```
1.  20 1 * * *  echo "" >/var/log/slow.log
2.  0 3 * * 0  /bin/sh /usr/local/sbin/backup.sh
3.  10 4 14 * *  /bin/sh /usr/local/sbin/backup_month.sh
4.  0 */8 * * *  ntpdate time.windows.com
5.  0 1,12,18 * *  /bin/sh /usr/local/sbin/test.sh
6.  0 9-18 * * *  /bin/sh /usr/local/sbin/test2.sh
```

练习完上面的题目，相信您会有一些小疑问，这里要简单说一下，每隔8小时，就是用全部小时（0-23）去除以8，仔细想一下结果，其实算出来应该是0,8,16三个数。当遇到多个数（分钟、小时、月、周）例如第5题，则需要用逗号隔开。而时间段是可以用 `n-m` 的方式表示的，比如第六题中的(9-18)。等设置好了所有的计划任务后需要查看一下crond服务是否启动:

```
[root@localhost ~]# service crond status
crond (pid  945) 正在运行...
```

如果是停止状态，则需要启动它:

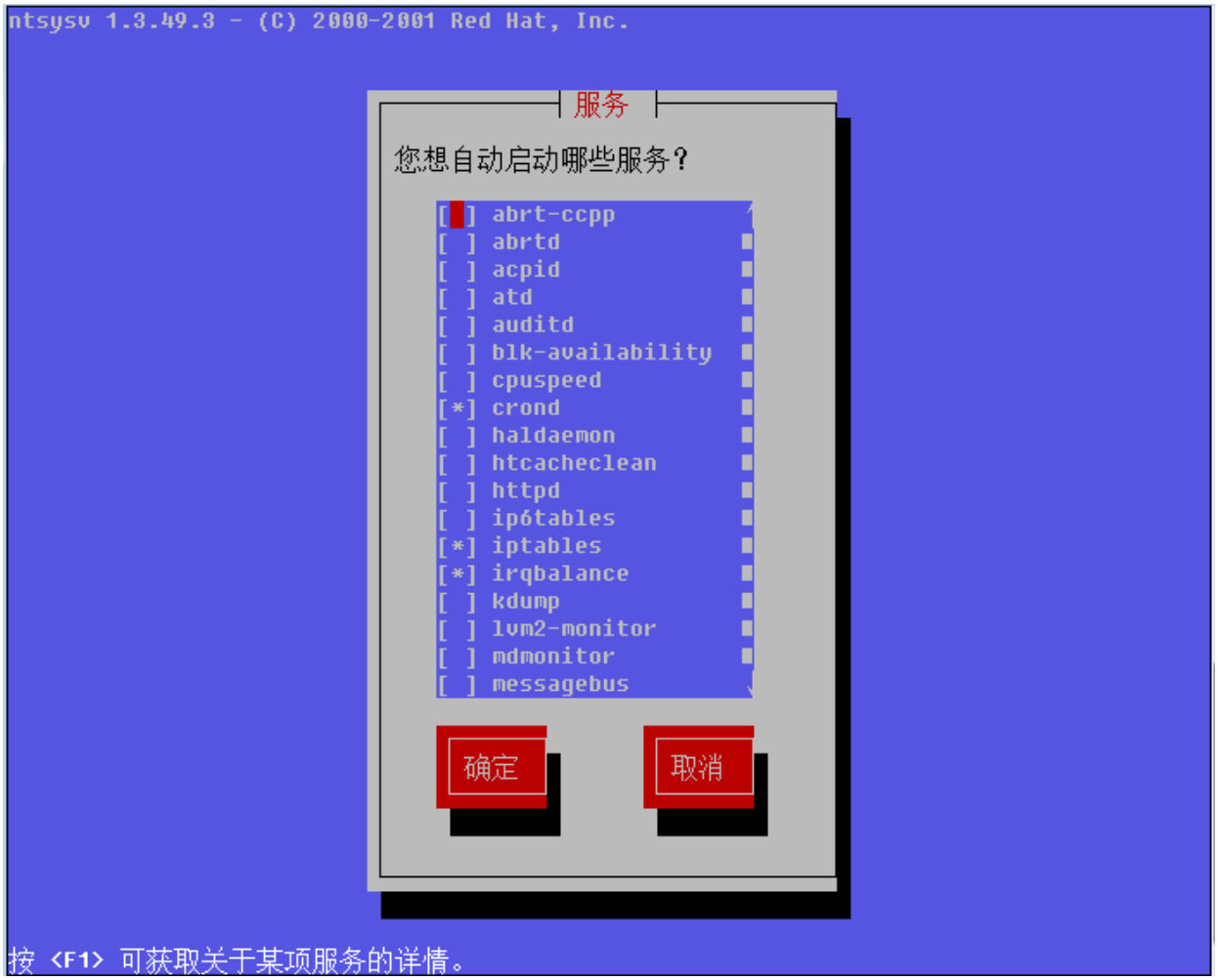
```
[root@localhost ~]# service crond status
crond 已停
[root@localhost ~]# service crond start
正在启动 crond：                                           [确定]
```

17.5 linux的系统服务管理

如果您对windows非常熟悉的话，相信您肯定配置过开机启动的服务，有些服务我们日常用不到则要把它停掉，一来可以节省资源，二来可以减少安全隐患。在linux上同样也有相关的工具来管理系统的服务。

1. ntsysv服务配置工具

用来配置哪些服务开启或者关闭，有点类似图形界面，不过是使用键盘来控制的。如果没有这个命令请使用 yum install -y ntsysv 安装它。安装好后，直接运行命令 ntsysv 回车后弹出一个配置界面：



按键盘的上下方向键可以调节红色光标，按空格可以选择开启或者不开启，如果前面的中括号内显示有 * 则表示开启否则不开启。通过这个工具也可以看到目前系统中所有的服务。建议除 ``crond, iptables, network, sshd, syslog, irqbalance, sendmail, microcode_ctl'' 外其他服务全部停掉。选择好后，按 ``tab'' 键选择 ``确定'', 然后回车，需要重启机器才能生效。

2. chkconfig服务管理工具

Linux系统所有的预设服务可以查看/etc/init.d/目录得到:

```
[root@localhost ~]# ls /etc/init.d/
abrt-ccpp         cpuspeed      ip6tables     mdmonitor   postfix      sandbox
abrtd             crond         iptables      messagebus  psacct
saslauthd
abrt-oops         functions     irqbalance    netconsole  quota_nld    single
acpid             haldaemon     kdump         netfs       rdisc        smartd
atd               halt          killall       network     restorecond  sshd
auditd            htcacheclean  lvm2-lvmetad  ntpd        rngd         sysstat
blk-availability  httpd         lvm2-monitor  ntpdate     rsyslog      udev-post
```

其实这就是系统所有的预设服务了。为什么这样讲，因为系统预设服务都是可以通过这样的命令实现

service 服务名 start|stop|restart 这里的服务名就是/etc/init.d/目录下的这些文件了。除了可以使用service crond start 启动crond外，还可以使用 /etc/init.d/crond start 来启动。

言归正传，我们可以使用 chkconfig --list 列出所有的服务以及每个级别是否开启:

```
[root@localhost ~]# chkconfig --list
abrt-ccpp       0:关闭  1:关闭  2:关闭  3:关闭  4:关闭  5:启用  6:关闭
abrtd           0:关闭  1:关闭  2:关闭  3:关闭  4:关闭  5:启用  6:关闭
acpid           0:关闭  1:关闭  2:启用  3:关闭  4:启用  5:启用  6:关闭
atd             0:关闭  1:关闭  2:关闭  3:关闭  4:启用  5:启用  6:关闭
auditd          0:关闭  1:关闭  2:启用  3:关闭  4:启用  5:启用  6:关闭
blk-availability        0:关闭  1:启用  2:启用  3:关闭  4:启用  5:启用  6:关闭
cpuspeed        0:关闭  1:启用  2:启用  3:关闭  4:启用  5:启用  6:关闭
crond           0:关闭  1:关闭  2:启用  3:启用  4:启用  5:启用  6:关闭
haldaemon       0:关闭  1:关闭  2:关闭  3:关闭  4:启用  5:启用  6:关闭
htcacheclean    0:关闭  1:关闭  2:关闭  3:关闭  4:关闭  5:关闭  6:关闭
httpd           0:关闭  1:关闭  2:关闭  3:关闭  4:关闭  5:关闭  6:关闭
ip6tables       0:关闭  1:关闭  2:启用  3:关闭  4:启用  5:启用  6:关闭
iptables        0:关闭  1:关闭  2:启用  3:启用  4:启用  5:启用  6:关闭
irqbalance      0:关闭  1:关闭  2:关闭  3:启用  4:启用  5:启用  6:关闭
kdump           0:关闭  1:关闭  2:关闭  3:关闭  4:启用  5:启用  6:关闭
lvm2-monitor    0:关闭  1:启用  2:启用  3:关闭  4:启用  5:启用  6:关闭
mdmonitor       0:关闭  1:关闭  2:启用  3:关闭  4:启用  5:启用  6:关闭
messagebus      0:关闭  1:关闭  2:启用  3:关闭  4:启用  5:启用  6:关闭
netconsole      0:关闭  1:关闭  2:关闭  3:关闭  4:关闭  5:关闭  6:关闭
netfs           0:关闭  1:关闭  2:关闭  3:关闭  4:启用  5:启用  6:关闭
network         0:关闭  1:关闭  2:启用  3:启用  4:启用  5:启用  6:关闭
ntpd            0:关闭  1:关闭  2:关闭  3:关闭  4:关闭  5:关闭  6:关闭
ntpdate         0:关闭  1:关闭  2:关闭  3:关闭  4:关闭  5:关闭  6:关闭
postfix         0:关闭  1:关闭  2:启用  3:关闭  4:启用  5:启用  6:关闭
psacct          0:关闭  1:关闭  2:关闭  3:关闭  4:关闭  5:关闭  6:关闭
quota_nld       0:关闭  1:关闭  2:关闭  3:关闭  4:关闭  5:关闭  6:关闭
rdisc           0:关闭  1:关闭  2:关闭  3:关闭  4:关闭  5:关闭  6:关闭
restorecond     0:关闭  1:关闭  2:关闭  3:关闭  4:关闭  5:关闭  6:关闭
rngd            0:关闭  1:关闭  2:关闭  3:关闭  4:关闭  5:关闭  6:关闭
rsyslog         0:关闭  1:关闭  2:启用  3:关闭  4:启用  5:启用  6:关闭
saslauthd       0:关闭  1:关闭  2:关闭  3:关闭  4:关闭  5:关闭  6:关闭
smartd          0:关闭  1:关闭  2:关闭  3:关闭  4:关闭  5:关闭  6:关闭
sshd            0:关闭  1:关闭  2:启用  3:启用  4:启用  5:启用  6:关闭
sysstat         0:关闭  1:启用  2:启用  3:启用  4:启用  5:启用  6:关闭
udev-post       0:关闭  1:启用  2:启用  3:关闭  4:启用  5:启用  6:关闭
```

这里的级别（0,1,2,3,4,5,6）就是 /etc/inittab 里面的那几个启动级别了，0、1、6运行级别被系统保留：其中0作为shutdown动作，1作为重启至单用户模式，6为重启；在一般的Linux系统实现中，都使用了2、3、4、5几个级别，在CentOS系统中，2表示无NFS支持的多用户模式，3表示完全多用户模式（也是最常用的级别），4保留给用户自定义，5表示图形登录方式。我们可以使用grep命令把我们想要看的服务过滤出来:

```
[root@localhost ~]# chkconfig --list |grep cron
crond           0:关闭  1:关闭  2:启用  3:启用  4:启用  5:启用  6:关闭
```

现在我们只是看到了各服务在每个级别下是否开启，那么如何去更改哪个级别下是否开启呢？

```
[root@localhost ~]# chkconfig --level 3 crond off
[root@localhost ~]# chkconfig --list |grep cron
crond           0:关闭  1:关闭  2:启用  3:关闭  4:启用  5:启用  6:关闭
```

用 --level 指定级别，后面是服务名，然后是off或者on，`--level 后还可以跟多个级别:

```
[root@localhost ~]# chkconfig --level 345 crond off
[root@localhost ~]# chkconfig --list |grep cron
crond           0:关闭  1:关闭  2:启用  3:关闭  4:关闭  5:关闭  6:关闭
```

另外还可以省略级别，默认是针对2,3,4,5级别操作:

```
[root@localhost ~]# chkconfig crond on
[root@localhost ~]# chkconfig --list |grep cron
crond           0:关闭  1:关闭  2:启用  3:启用  4:启用  5:启用  6:关闭
```

chkconfig 还有一个功能就是可以把某个服务加入到系统服务，即可以使用 service 服务名 start 这样的形式，并且可以在 chkconfig --list 中查找到。当然也能删除掉。

```
[root@localhost ~]# chkconfig --del crond
[root@localhost ~]# chkconfig --list |grep cron
[root@localhost ~]# chkconfig --add crond
[root@localhost ~]# chkconfig --list |grep cron
crond           0:关闭  1:关闭  2:启用  3:启用  4:启用  5:启用  6:关闭
```

这个功能常用在把自定义的启动脚本加入到系统服务当中。关于系统服务就讲这些内容，其实还有很多内容阿铭没有介绍，道理很简单，一来讲多了您不能消化二来讲多了您也用不上。

17.6 linux下的数据备份工具rsync

数据备份，毫无疑问很重要。阿铭就曾经有过一次非常痛苦的经历，备份策略没有做好，结果磁盘坏掉数据丢失，简直是撕心裂肺的痛呀。还好数据重要性不是特别高，即使是不高也是丢失了数据，这是作为系统管理员最不应该出现的事故。所以，在您以后的系统维护工作中，一定要把数据备份当回事，认真对待。在linux系统下数据备份的工具很多，但阿铭就只用一种那就是rsync. 从字面上的意思您可以理解为remote sync （远程同步）这样可以让您理解的更深刻一些。Rsync不仅可以远程同步数据（类似于scp[1]），当然还可以本地同步数据（类似于cp），但不同于cp或scp的一点是，rsync不像cp/scp一样会覆盖以前的数据（如果数据已经存在），它会先判断已经存在的数据和新数据有什么不同，只有不同时才会把不同的部分覆盖掉。如果您的linux没有rsync命令请使用 yum install -y rsync 安装。

下面阿铭先举一个例子，然后再详细讲解rsync的用法:

```
[root@localhost ~]# rsync -av 123.txt /tmp/
 sending incremental file list
 123.txt

 sent 71 bytes  received 31 bytes  204.00 bytes/sec
 total size is 0  speedup is 0.00
```

上面例子表示把当前目录下的123.txt同步到/tmp/目录下，也可以更改目标文件的名字， rsync -av 123.txt /tmp/234.txt 如果是远程拷贝的话就是这样的形式了: IP:path （如：10.0.2.34:/root/）

```
[root@localhost ~]# rsync -av 123.txt 192.168.0.101:/data/
The authenticity of host '192.168.0.101 (192.168.0.101)' can't be established.
RSA key fingerprint is b4:54:5f:73:ec:c2:60:5f:c3:79:c0:f9:51:e9:ac:e5.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.101' (RSA) to the list of known hosts.
root@192.168.0.101's password:
```

[1] scp 用来远程拷贝数据，通过ssh协议通信。它的语法很简单，类似于cp, 唯一不同的是，源地址或者目标地址需要使用远程主机的ip或者hostname. 例如要把本地的数据拷贝到远程一台主机(192.168.0.111)的/data/目录下，可以这样实现: scp /dir/filename root@192.168.0.111:/data/ 其中filename 可以是目录也可以是文件。或者也可以把远程的文件拷贝到本地: scp root@192.168.0.111:/data/filename /data/

首次连接会提示是否要继续连接，我们输入yes继续，当建立连接后，需要输入密码。如果手动去执行这些操作还好，但若是写在脚本中怎么办？这就涉及到添加信任关系了，该部分内容稍后会详细介绍。

1. rsync的命令格式

rsync [OPTION]... SRC DEST

rsync [OPTION]... SRC [USER@]HOST:DEST

rsync [OPTION]... [USER@]HOST:SRC DEST

rsync [OPTION]... [USER@]HOST::SRC DEST

rsync [OPTION]... SRC [USER@]HOST::DEST

阿铭在一开始举的两个例子，第一个例子即为第一种格式，第二个例子即为第二种格式，但不同的是，阿铭并没有加user@host 如果不加默认指的是root. 第三种格式是从远程目录同步数据到本地。第四种以及第五种格式使用了两个冒号，这种方式和前面的方式的不同在于验证方式不同，稍后详细介绍。

2. rsync常用选项

-a 归档模式，表示以递归方式传输文件，并保持所有属性，等同于-rlptgoD, -a选项后面可以跟一个--no-OPTION 这个表示关闭-rlptgoD中的某一个例如 -a--no-l 等同于-rptgoD

-r 对子目录以递归模式处理，主要是针对目录来说的，如果单独传一个文件不需要加-r，但是传输的是目录必须加-r选项

-v 打印一些信息出来，比如速率，文件数量等

-l 保留软链结

-L 向对待常规文件一样处理软链结，如果是SRC中有软连接文件，则加上该选项后将会把软连接指向的目标文件拷贝到DST

-p 保持文件权限

-o 保持文件属主信息

-g 保持文件属组信息

-D 保持设备文件信息

-t 保持文件时间信息

--delete 删除那些DST中SRC没有的文件

--exclude=PATTERN 指定排除不需要传输的文件，等号后面跟文件名，可以是万用字符模式（如*.txt）

--progress 在同步的过程中可以看到同步的过程状态，比如统计要同步的文件数量、同步的文件传输速度等等

-u 加上这个选项后将会把DST中比SRC还新的文件排除掉，不会覆盖

选项确实有点多，不过不用担心，阿铭工作这么多年，常用的选项页仅仅那么几个: (-a -v --delete --exclude), 请熟记他们吧。

下面阿铭将会针对这些选项做一些列小实验：

1） 建立目录以及文件:

```
[root@localhost ~]# mkdir rsync
[root@localhost ~]# cd rsync
[root@localhost rsync]# mkdir test1
[root@localhost rsync]# cd test1
[root@localhost test1]# touch 1 2 3
[root@localhost test1]# ln -s /root/123.txt ./123.txt
[root@localhost test1]# ls -l
```

```
总用量 0
-rw-r--r-- 1 root root  0  6月 10 12:58 1
lrwxrwxrwx 1 root root 13  6月 10 12:59 123.txt -> /root/123.txt
-rw-r--r-- 1 root root  0  6月 10 12:58 2
-rw-r--r-- 1 root root  0  6月 10 12:58 3
[root@localhost test1]# cd ..
```

阿铭建立这些文件的目的就是为做试验做一些准备工作。

2）使用 -a 选项

```
[root@localhost rsync]# rsync -a test1 test2
[root@localhost rsync]# ls test2
test1
[root@localhost rsync]# ls test2/test1/
1  123.txt  2  3
```

这里有一个问题，就是本来想把test1目录直接拷贝成test2目录，可结果rsync却新建了test2目录然后把test1放到test2当中。为了避免这样的情况发生，可以这样做:

```
[root@localhost rsync]# rm -rf test2
[root@localhost rsync]# rsync -a test1/ test2/
[root@localhost rsync]# ls -l test2/
总用量 0
-rw-r--r-- 1 root root  0  6月 10 12:58 1
lrwxrwxrwx 1 root root 13  6月 10 12:59 123.txt -> /root/123.txt
-rw-r--r-- 1 root root  0  6月 10 12:58 2
-rw-r--r-- 1 root root  0  6月 10 12:58 3
```

加一个斜杠就好了，所以阿铭建议您在使用rsync备份目录时要养成加斜杠的习惯。在上面讲了-a选项等同于-rlptgoD，而且 -a 还可以和 --no-OPTIN 一并使用。下面看看-l选项的作用:

```
[root@localhost rsync]# rsync -av --no-l test1/ test2/
sending incremental file list
created directory test2
./
1
skipping non-regular file "123.txt"
2
3

sent 200 bytes  received 72 bytes  544.00 bytes/sec
total size is 13  speedup is 0.05
```

使用-v选项看来就是方便呀，上例告诉我们跳过了非普通文件123.txt，其实123.txt是一个软连接文件，如果不使用-l选项则不理会软连接文件的。虽然加上-l选项会把软连接文件给拷贝过去，但是软连接的目标文件却没有拷贝过去，有时候咱们指向拷贝软连接文件所指向的目标文件，那这时候该怎么办呢？

3）使用-L选项

```
[root@localhost rsync]# rsync -avL test1/ test2/
sending incremental file list
created directory test2
./
1
123.txt
2
3
```

```
sent 231 bytes  received 91 bytes  644.00 bytes/sec
total size is 0  speedup is 0.00
[root@localhost rsync]# ls -l test2/
总用量 0
-rw-r--r-- 1 root root 0  6月 10 12:58 1
-rw-r--r-- 1 root root 0  6月 10 12:39 123.txt
-rw-r--r-- 1 root root 0  6月 10 12:58 2
-rw-r--r-- 1 root root 0  6月 10 12:58 3
```

加上 -L 选项就可以把SRC中软连接的目标文件给拷贝到DST.

4） 使用-u选项

首先查看一下test1/1 和test2/1的创建时间（肯定是一样的），然后使用touch修改一下test2/1的创建时间（此时test2/1要比test1/1的创建时间晚了一些），如果不加-u选项的话，会把test2/1的创建时间变成和test1/1的创建时间一样。这样讲也许您会迷糊，不妨看一看:

```
[root@localhost rsync]# ll test1/1 test2/1
-rw-r--r-- 1 root root 0  6月 10 12:58 test1/1
-rw-r--r-- 1 root root 0  6月 10 12:58 test2/1
```

两者之间的创建时间是一样的，下面修改test2/1 的创建时间，然后不加-u同步:

```
[root@localhost rsync]# touch test2/1
[root@localhost rsync]# ll test2/1
-rw-r--r-- 1 root root 0  6月 10 13:20 test2/1
[root@localhost rsync]# rsync -a test1/1 test2/
[root@localhost rsync]# ll test2/1
-rw-r--r-- 1 root root 0  6月 10 12:58 test2/1
```

test2/1 的创建时间又变成和test1/1的创建时间一样了。下面加上 -u 再看看结果是怎么样的:

```
[root@localhost rsync]# touch test2/1
[root@localhost rsync]# ll test2/1
-rw-r--r-- 1 root root 0  6月 10 13:31 test2/1
[root@localhost rsync]# rsync -avu test1/ test2/
sending incremental file list
./
123.txt -> /root/123.txt

sent 100 bytes  received 18 bytes  236.00 bytes/sec
total size is 13  speedup is 0.11
[root@localhost rsync]# ll test2/1
-rw-r--r-- 1 root root 0  6月 10 13:31 test2/1
[root@localhost rsync]# ll test1/1
-rw-r--r-- 1 root root 0  6月 10 12:58 test1/1
```

加上-u 选项后，不会再把 test1/1 同步为 test2/1 了，现在您明白 -u 选项的妙用了吧。

5）使用 --delete 选项

首先删除test1/123.txt:

```
[root@localhost rsync]# rm -f test1/123.txt
[root@localhost rsync]# ls test1/
1  2  3
```

然后把test1/ 目录 同步到 test2/ 目录下:

```
[root@localhost rsync]# rsync -av test1/ test2/
sending incremental file list
./
1

sent 94 bytes  received 34 bytes  256.00 bytes/sec
total size is 0  speedup is 0.00
[root@localhost rsync]# ls test2/
1  123.txt  2  3
```

test2/目录并没有删除掉123.txt, 下面加上 --delete 选项:

```
[root@localhost rsync]# rsync -av --delete test1/ test2/
sending incremental file list
deleting 123.txt

sent 52 bytes  received 12 bytes  128.00 bytes/sec
total size is 0  speedup is 0.00
[root@localhost rsync]# ls test2/
1  2  3
```

test2/ 目录里的123.txt也被删除了，这就是 --delete 选项的用处。还有一种情况就是如果在DST增加文件了，而SRC当中没有这些文件，同步时加上 --delete 选项后同样会删除新增的文件:

```
[root@localhost rsync]# touch test2/4
[root@localhost rsync]# ls test1/
1  2  3
[root@localhost rsync]# ls test2/
1  2  3  4
[root@localhost rsync]# rsync -a --delete test1/ test2/
[root@localhost rsync]# ls test1/
1  2  3
[root@localhost rsync]# ls test2/
1  2  3
```

6）使用 --exclude 选项

```
[root@localhost rsync]# touch test1/4
[root@localhost rsync]# rsync -a --exclude="4" test1/ test2/
[root@localhost rsync]# ls test1/
1  2  3  4
[root@localhost rsync]# ls test2/
1  2  3
```

另外还可以使用匹配字符 *

```
[root@localhost rsync]# touch test1/1.txt test1/2.txt
[root@localhost rsync]# ls test1/
1  1.txt  2  2.txt  3  4
[root@localhost rsync]# rsync -a --progress --exclude="*.txt" test1/ test2/
sending incremental file list
./
4
            0 100%    0.00kB/s    0:00:00 (xfer#1, to-check=0/5)

sent 104 bytes  received 34 bytes  276.00 bytes/sec
total size is 0  speedup is 0.00
[root@localhost rsync]# ls test2/
1  2  3  4
```

上例中，阿铭也连带着使用了 --progress 选项，这个主要是用来观察rsync同步过程的状态的。最后简单总结一下，平时您使用rsync同步数据的时候，使用-a选项基本上就可以达到我们想要的效果了，只是有时候会有个别的需求，会用到 -a --no-OPTION, -u, -L, --delete, --exclude 以及 progress 这些选项，还有些选项阿铭都没有介绍，如果在以后的工作中遇到特殊需求了，就去查一下rsync的man文档吧。

3. rsync 应用实例

1）通过ssh的方式

最上面介绍的5种方式当中，第二、第三（1个冒号）就属于通过ssh的方式，这种方式其实就是让用户去登录到远程机器，然后执行rsync的任务。

```
[root@localhost rsync]# rsync -avL test1/ www@192.168.0.101:/tmp/test2/
www@192.168.0.101's password:
sending incremental file list
created directory /tmp/test2
./
1
1.txt
2
2.txt
3
4

sent 327 bytes  received 129 bytes  182.40 bytes/sec
total size is 0  speedup is 0.00
```

这种方式就是前面介绍的第二种方式了，是通过ssh拷贝的数据，需要输入192.168.0.101 那台机器 www 账户的密码。当然也可以使用第三种方式拷贝:

```
[root@localhost rsync]# rsync -avL www@192.168.0.101:/tmp/test2/ ./test3/
www@192.168.0.101's password:
receiving incremental file list
created directory ./test3
./
1
1.txt
2
2.txt
3
4

sent 128 bytes  received 351 bytes  38.32 bytes/sec
total size is 0  speedup is 0.00
```

以上两种方式如果写到脚本里，备份起来就有麻烦了，因为要输入密码，脚本本来就是自动的，不可能做到的。但是不代表没有解决办法。那就是通过密钥验证，密钥不设立密码就ok了。还记得在前面阿铭曾经介绍过通过密钥登录远程主机吗，下面要讲的内容就是那些东西了。

在操作之前我们先讲明主机信息： 192.168.0.10 （主机名Aming-1）和 192.168.0.101 （主机名Aming）需要从Aming-1上拷贝数据到Aming上。

首先确认一下Aming-1上是否有这个文件 /root/.ssh/id_rsa.pub:

```
[root@Aming-1 ~]# ssh-keygen
Generating public/private rsa key pair.
```

阿铭之前生成过密钥对，所以这个文件已经存在了，如果您的Linux不存在这个文件，请按照如下方法生成:

```
[root@Aming-1 ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
3b:74:af:e8:08:ac:99:30:3f:ef:84:7a:a0:a6:3d:89 root@Aming-1
```

在这个过程中会有一些交互的过程，它首先提示要输入这个密钥的密码，出于安全考虑应该定义个密码，但是我们的目的就是为了自动化同步数据，所以这里不输入任何密码，直接按回车，即密码为空。最后则生成了私钥(/root/.ssh/id_rsa)和公钥文件(/root/.ssh/id_rsa.pub)

把公钥文件的内容拷贝到目标机器上:

```
[root@Aming-1 ~]# cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA5SPyJ/kliGTAMUan/GCN325VS8jMxvOn4LU/NqBpCI3MrmvSucv6EAz....aming-1
```

复制主机Aming-1的/root/.ssh/id_rsa.pub文件内容，并粘贴到主机Aming的/home/www/.ssh/authorized_keys中:

```
[root@Aming ~]# vim /home/www/.ssh/authorized_keys
```

在这一步也许您会遇到/home/www/.ssh目录不存在的问题，可以手动创建，并修改目录权限为700也可以执行ssh-keygen命令生成这个目录。保存/home/www/.ssh/authorized_keys文件后，再到主机Aming-1上执行:

```
[root@Aming-1 ~]# ssh www@192.168.0.101
Last login: Wed Jun 12 12:24:34 2013 from 192.168.0.10
[www@Aming ~]$
```

现在不用输入密码也可以登录主机Aming了。下面先从Aming主机退出来，再从主机Aming-1上执行一下rsync命令试试吧。

```
[root@Aming-1 ~]# rsync -av rsync/test1/ www@192.168.0.101:/tmp/test4/
sending incremental file list
created directory /tmp/test4
./
1
1.txt
2
2.txt
3
4

sent 327 bytes  received 129 bytes  912.00 bytes/sec
total size is 0  speedup is 0.00
```

2） 通过后台服务的方式

这种方式可以理解成这样，在远程主机上建立一个rsync的服务器，在服务器上配置好rsync的各种应用，然后本机作为rsync的一个客户端去连接远程的rsync服务器。下面阿铭就介绍一下，如何去配置一台rsync服务器。

1. 建立并配置rsync的配置文件 /etc/rsyncd.conf

```
[root@Aming-1 ~]# vim /etc/rsyncd.conf
#port=873
```

```
log file=/var/log/rsync.log
pid file=/var/run/rsyncd.pid
#address=192.168.0.10

[test]
path=/root/rsync
use chroot=true
max connections=4
read only=no
list=true
uid=root
gid=root
auth users=test
secrets file=/etc/rsyncd.passwd
hosts allow=192.168.0.101
```

其中配置文件分为两部分：全部配置部分和模块配置部分，全局部分就是几个参数而已，就像阿铭的rsyncd.conf中port, log file, pid file, address这些都属于全局配置，而[test]以下部分就是模块配置部分了。一个配置文件中可以有多个模块，模块名自定义，格式就像阿铭的rsyncd.conf中的这样。其实模块中的一些参数例如use chroot, max connections, udi, gid, auth users, secrets file以及hosts allow都可以配置成全局的参数。当然阿铭给出的参数并不是所有的，你可以通过man rsyncd.conf 获得更多信息。下面就简单解释一下这些参数的意义：

port 指定在哪个端口启动rsyncd服务，默认是873

log file 指定日志文件

pid file 指定pid文件，这个文件的作用涉及到服务的启动以及停止等进程管理操作

address 指定启动rsyncd服务的IP，假如你的机器有多个IP，就可以指定其中一个启动rsyncd服务，默认是在全部IP上启动

[test] 指定模块名，自定义

path 指定数据存放的路径

use chroot true|false 默认是true，意思是在传输文件以前首先chroot到path参数所指定的目录下。这样做的原因是实现额外的安全防护，但是缺点是需要以roots权限，并且不能备份指向外部的符号连接所指向的目录文件。默认情况下chroot值为true，如果你的数据当中有软连接文件的话建议设置成false。

max connections 指定最大的连接数，默认是0即没有限制

read only ture|false 如果为true则不能上传到该模块指定的路径下

list 指定当用户查询该服务器上的可用模块时，该模块是否被列出，设定为true则列出，false则隐藏

uid/gid 指定传输文件时，以哪个用户/组的身份传输

auth users 指定传输时要使用的用户名

secrets file 指定密码文件，该参数连同上面的参数如果不指定则不使用密码验证，注意该密码文件的权限一定要是600

hosts allow 指定被允许连接该模块的主机，可以是IP或者网段，如果是多个，之间用空格隔开

2. 编辑secrets file，保存后要赋予600权限，如果权限不对，不能完成同步

```
[root@Aming-1 ~]# cat /etc/rsyncd.passwd
test:test123
[root@Aming-1 ~]# chmod 600 /etc/rsyncd.passwd
```

3. 启动rsyncd服务

```
[root@Aming-1 ~]# rsync --daemon --config=/etc/rsyncd.conf
```

启动后，可以查看一下日志，并查看端口是否启动:

```
[root@Aming-1 ~]# cat /var/log/rsync.log
[root@Aming-1 ~]# netstat -lnp |grep 873
tcp     0    0 0.0.0.0:873        0.0.0.0:*  LISTEN      12066/rsync
tcp     0    0 :::873             :::*       LISTEN      12066/rsync
```

如果想开机启动，请把 `rsync --daemon --confg=/etc/rsyncd.conf` 写入到/etc/rc.d/rc.local文件。

4. 到另一台机器上测试

```
[root@Aming ~]# rsync -avL test@192.168.0.10::test/test1/ /tmp/test5/
Password:
receiving incremental file list
created directory /tmp/test5
./
1
1.txt
2
2.txt
3
4

sent 143 bytes  received 354 bytes  994.00 bytes/sec
total size is 0  speedup is 0.00
```

阿铭刚刚提到有一个选项叫做 ``use chroot'' 默认为true，如果是true，同步的文件中如果有软连接，则会有问题，首先在主机Aming-1的/root/rsync/test1/ 目录下创建一个软连接文件:

```
[root@Aming-1 ~]# ln -s /root/test.txt rsync/test1/test.txt
[root@Aming-1 ~]# ls -l rsync/test1/test.txt
lrwxrwxrwx 1 root root 14  6月 12 13:24 rsync/test1/test.txt -> /root/test.txt
```

然后再到主机Aming上，同步:

```
[root@Aming ~]# rsync -avL test@192.168.0.10::test/test1/ /tmp/test6/
Password:
receiving incremental file list
symlink has no referent: "/test1/test.txt" (in test)
created directory /tmp/test6
./
1
1.txt
2
2.txt
3
4

sent 143 bytes  received 419 bytes  1124.00 bytes/sec
total size is 0  speedup is 0.00
rsync error: some files/attrs were not transferred (see previous errors) (code
23) at main.c(1532) [generator=3.0.6]
```

可以看到，如果设置 ``use chroot'' 为true则同步软连接文件会有问题，下面阿铭把主机Aming-1的rsync配置文件修改一下，把true改为false:

```
[root@Aming-1 ~]# sed -i 's/use chroot=true/use chroot=false/'  /etc/rsyncd.conf
[root@Aming-1 ~]# grep 'use chroot' /etc/rsyncd.conf
use chroot=false
```

然后再到主机Aming上再次执行同步:

```
[root@Aming ~]# rsync -avL test@192.168.0.10::test/test1/ /tmp/test7/
Password:
receiving incremental file list
created directory /tmp/test7
./
1
1.txt
2
2.txt
3
4
test.txt

sent 162 bytes  received 410 bytes  1144.00 bytes/sec
total size is 0  speedup is 0.00
```

这样就没有任何问题啦，您也许会奇怪，为什么阿铭修改完rsyncd.conf配置文件后，没有重启rsyncd服务呢？其实这是rsync的一个特定机制，配置文件时即时生效的，不用重启服务。

上面的例子中，阿铭都有输入密码，这样同样也不能写入脚本中自动执行，其实这种方式也是可以不用手动输入密码的，它有两种实现方式。

第一种，指定密码文件

在客户端上，也就是主机Aming上，编辑一个密码文件:

```
[root@Aming ~]# vim /etc/pass
```

加入test用户的密码:

```
[root@Aming ~]# cat /etc/pass
test123
```

修改密码文件的权限:

```
[root@Aming ~]# chmod 600 /etc/pass
```

在同步的时候，指定一下密码文件，就可以省去输入密码的步骤了:

```
[root@Aming ~]# rsync -avL test@192.168.0.10::test/test1/ /tmp/test8/ --password-file=/etc/pass
receiving incremental file list
created directory /tmp/test8
./
1
1.txt
2
2.txt
3
4
test.txt

sent 190 bytes  received 451 bytes  1282.00 bytes/sec
total size is 0  speedup is 0.00
```

第二种：在rsync服务器端不指定用户

在服务端也就是主机Aming-1上修改配置文件rsyncd.conf，去掉关于认证账户的配置项(auth user 和 secrets file这两行):

```
sed -i 's/auth users/#auth users/;s/secrets file/#secrets file/' /etc/rsyncd.conf
```

上面的这个命令是把 ``auth users'' 和 ``secrets file'' 两行的最前面加一个 ``#''，这样就把这两行注释掉，使其失去意义。在前面阿铭未曾讲过sed的这种用法，其实也不难弄明白，只是用分号把两个替换的子命令块给替换了而已。然后我们再到客户端主机Aming上测试:

```
[root@Aming ~]# rsync -avl 192.168.0.10::test/test1/ /tmp/test9/
receiving incremental file list
created directory /tmp/test9
./
1
1.txt
2
2.txt
3
4
test.txt

sent 162 bytes  received 410 bytes  1144.00 bytes/sec
total size is 0  speedup is 0.00
```

注意，这里不用再加test这个用户了，默认是以root的身份拷贝的，现在已经不需要输入密码了。

17.7 linux系统日志

日志重要吗？必须的，没有日志我们怎么知道系统状况？没有日志如何排查一个trouble？日志记录了系统每天发生的各种各样的事情，您可以通过他来检查错误发生的原因，或者受到攻击时攻击者留下的痕迹。日志主要的功能有：审计和监测，还可以实时的监测系统状态，监测和追踪侵入者等等。

阿铭常查看的日志文件为/var/log/message，它是核心系统日志文件，包含了系统启动时的引导消息，以及系统运行时的其他状态消息。IO错误、网络错误和其他系统错误都会记录到这个文件中。另外其他信息，比如某个人的身份切换为root以及用户自定义安装的软件（apache）的日志也会在这里列出。通常，/var/log/messages是在做故障诊断时首先要查看的文件。那您肯定会说了，这么多日志都记录到这个文件中，那如果服务器上有很多服务岂不是这个文件很快就会写的很大，没错，但是系统有一个日志轮询的机制，每星期切换一个日志，变成message.xxxxxxxx, message.xxxxxxxx, ... messages.xxxxxxxx 连同messages一共有5个这样的日志文件。这里的xxxxxxxx就是按照日期的格式生成的文件，在CentOS5里，这个后缀并不是日期而是数字1,2,3,4. 这是通过logrotate工具的控制来实现的，它的配置文件是/etc/logrotate.conf如果没有特殊需求请不要修改这个配置文件。

```
[root@localhost ~]# cat /etc/logrotate.conf
# see "man logrotate" for details
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# use date as a suffix of the rotated file
```

```
dateext

# uncomment this if you want your log files compressed
#compress

# RPM packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp and btmp -- we'll rotate them here
/var/log/wtmp {
    monthly
    create 0664 root utmp
        minsize 1M
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0600 root utmp
    rotate 1
}

# system-specific logs may be also be configured here.
```

/var/log/messages是由syslogd这个守护进程产生的，如果停掉这个服务则系统不会产生/var/log/messages，所以这个服务不要停。Syslogd服务的配置文件为/etc/syslog.conf这个文件定义了日志的级别，具体详细的东西阿铭不再阐述，因为若没有特殊需求是不需要修改这个配置文件的，请使用 man syslog.conf 获得更多关于它的信息。

除了关注/var/log/messages外，你还应该多关注一下 dmesg 这个命令，它可以显示系统的启动信息，如果你的某个硬件有问题（比如说网卡）用这个命令也是可以看到的。

```
[root@localhost ~]# dmesg |less
Initializing cgroup subsys cpuset
Initializing cgroup subsys cpu
Linux version 2.6.32-220.el6.x86_64 (mockbuild@c6b18n3.bsys.dev.centos.org)
(gcc version 4.4.6 20110731 (Red Hat 4.4.6-3) (GCC) ) #1 SMP Tue Dec 6
19:48:22 GMT 2011
Command line: ro root=UUID=7912412b-3e66-401d-9ef5-3c2aba8dc737 rd_NO_LUKS
KEYBOARDTYPE=pc KEYTABLE=us rd_NO_MD quiet rhgb crashkernel=auto
LANG=zh_CN.UTF-8 rd_NO_LVM rd_NO_DM
KERNEL supported cpus:
  Intel GenuineIntel
  AMD AuthenticAMD
  Centaur CentaurHauls
BIOS-provided physical RAM map:
BIOS-e820: 0000000000000000 - 000000000009a400 (usable)
BIOS-e820: 000000000009a400 - 00000000000a0000 (reserved)
BIOS-e820: 00000000000d2000 - 00000000000d4000 (reserved)
BIOS-e820: 00000000000e4000 - 0000000000100000 (reserved)
BIOS-e820: 0000000000100000 - 00000000cff60000 (usable)
BIOS-e820: 00000000cff60000 - 00000000cff69000 (ACPI data)
```

关于安全方面的日志，阿铭简单介绍几个命令或者日志。

命令：last

```
[root@localhost ~]# last |head
root     pts/0        192.168.0.207  Wed Jun 12 20:28   still logged in
root     pts/1        192.168.0.207  Wed Jun 12 20:27   still logged in
root     pts/0        192.168.0.161    Wed Jun 12 14:36 - 20:27  (05:50)
root     pts/0        192.168.0.161    Wed Jun 12 14:36 - 14:36  (00:00)
root     pts/0        192.168.0.207  Wed Jun 12 11:42 - 14:36  (02:54)
root     pts/0        192.168.0.207  Mon Jun 10 12:23 - 14:23  (02:00)
root     pts/0        192.168.0.70    Sat Jun  8 16:43 - 17:53  (01:09)
root     pts/0        192.168.0.70    Fri Jun  7 16:43 - 17:27  (00:44)
root     pts/0        192.168.0.70    Fri Jun  7 09:57 - 16:09  (06:11)
root     pts/0        192.168.0.70    Thu Jun  6 13:40 - 17:50  (04:09)
```

last命令用来查看登录Linux历史信息，从左至右依次为账户名称、登录终端、登录客户端ip、登录日期及时长。last命令输出的信息实际上是读取了二进制日志文件/var/log/wtmp, 只是这个文件不能直接使用cat, vim, head, tail等工具查看。

另外一个和登陆信息有关的日志文件为/var/log/secure, 该日志文件记录验证和授权等方面的信息，比如ssh登陆系统成功或者失败，都会把相关信息记录在这个日志里。

这一小节就介绍这么多，在结束之前，阿铭给您一个小小的建议。以后在您日常的管理工总中要养成多看日志的习惯，尤其是一些应用软件的日志，比如apache, mysql, php等常用的软件，看它们的日志（错误日志）可以帮助你排查问题以及监控它们的运行状况是否良好。

17.8 xargs与exec

1. xargs应用

在前面的例子中阿铭曾经使用过这个命令，你是否有印象呢？现在就详细介绍一下它，平时阿铭使用xargs还是比较多的，很方便。

```
[root@localhost ~]# echo "121212121212" > 123.txt
[root@localhost ~]# ls 123.txt | xargs cat
121212121212
```

它的作用就是把管道符前面的输出作为xargs后面的命令的输入。它的好处在于可以把本来两步或者多步才能完成的任务简单一步就能完成。xargs常常和find命令一起使用，比如，查找当前目录创建时间大于10天的文件，然后再删除。

```
[root@localhost ~]# find . -mtime +10 |xargs rm
```

这种应用是最为常见的，xargs后面的rm 也可以加选项，当是目录时，就需要-r选项了。在阿铭看来xargs的这个功能不叫什么，它的另一个功能才叫神奇。现在我有一个这样的需求，查找当前目录下所有.txt的文件，然后把这些.txt的文件变成.txt_bak。正常情况下，我们不得不写脚本去实现，但是使用xargs就一步。

```
[root@localhost ~]# mkdir test
[root@localhost ~]# cd test
[root@localhost test]# touch 1.txt 2.txt 3.txt 4.txt 5.txt
[root@localhost test]# ls
1.txt  2.txt  3.txt  4.txt  5.txt
[root@localhost test]# ls *.txt |xargs -n1 -i{} mv {} {}_bak
[root@localhost test]# ls
1.txt_bak  2.txt_bak  3.txt_bak  4.txt_bak  5.txt_bak
```

xargs -n1 –i{} 类似for循环，-n1意思是一个一个对象的去处理，-i{}把前面的对象使用{}取代，mv {} {}_bak 相当于 mv 1.txt 1.txt_bak。你刚开始接触这个命令时也许有点难以理解，多练习一下你就会熟悉了，笔者建议你记住这个应用，很实用。

2. exec应用

使用find命令时，经常使用一个选项就是这个-exec了，可以达到和xargs同样的效果。比如，查找当前目录创建时间大于10天的文件并删除:

```
[root@localhost ~]# find . -mtime +10 -exec rm -rf {} \;
```

这个命令中也是把{}作为前面find出来的文件的替代符，后面的 \ 为；的脱意符，不然shell会把分号作为该行命令的结尾。这个-exec有时候也挺实用的，它同样可以实现刚刚上面批量更改文件名的需求:

```
[root@localhost test]# ls
1.txt_bak  2.txt_bak  3.txt_bak  4.txt_bak  5.txt_bak
[root@localhost test]# find ./*_bak -exec mv {} {}_bak \;
[root@localhost test]# ls
1.txt_bak_bak  2.txt_bak_bak  3.txt_bak_bak  4.txt_bak_bak  5.txt_bak_bak
```

17.9 screen工具介绍

有时候，我们也许会有这样的需求，要执行一个命令或者脚本，但是需要几个小时甚至几天。这就要考虑一个问题，就是中途断网或出现其他意外情况，执行的任务中断了怎么办？您可以把命令或者脚本丢到后台运行，不过也不保险。阿铭下面就介绍两种方法来避免这样的问题发生。

1. 使用nohup

```
[root@localhost ~]# cat /usr/local/sbin/sleep.sh
#! /bin/bash
sleep 1000
[root@localhost ~]# nohup sh /usr/local/sbin/sleep.sh &
[1] 19997
[root@localhost ~]# nohup: 忽略输入并把输出追加到"nohup.out"
```

直接加一个 `&' 虽然丢到后台了，但是当退出该终端时很有可能这个脚本也会退出的，而在前面加上 nohup 就没有问题了，nohup的作用就是不挂断地运行命令。

2. screen工具的使用

简单来说，screen是一个可以在多个进程之间多路复用一个物理终端的窗口管理器。screen中有会话的概念，用户可以在一个screen会话中创建多个screen窗口，在每一个screen窗口中就像操作一个真实的SSH连接窗口那样。下面阿铭介绍screen的一个简单应用。

1）打开一个会话，直接输入screen命令然后回车，进入screen会话窗口。如果你没有screen命令，请用 `yum install -y screen` 安装。

```
[root@localhost ~]# screen
[root@localhost ~]#
```

2）screen -ls 查看已经打开的screen会话

```
[root@localhost ~]# screen -ls
There is a screen on:
        20001.pts-0.localhost   (Attached)
1 Socket in /var/run/screen/S-root.
```

3）Ctrl +a 再按d退出该screen会话，只是退出，并没有结束。结束的话输入Ctrl +d 或者输入exit

4）退出后还想再次登录某个screen会话，使用sreen -r [screen 编号]，这个编号就是上例中那个20001. 当只有一个screen会话时，后面的编号是可以省略的。当你有某个需要长时间运行的命令或者脚本时就打开一个screen会话，然后运行该任务。按ctrl +a 再按d退出会话，不影响终端窗口上的任何操作。

阿铭建议您最好再扩展学习一下: http://www.lishiming.net/thread-5440-1-1.html

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

SEVENTEEN

第十七章 LAMP环境搭建

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

经过前部分章节的学习，您已经掌握了linux的基础知识了。但是想成为一名系统管理员恐怕还有点难度，因为好多单位招聘这个职位的时候都要求有一定的工作经验。然而真正的经验一天两天是学不来的，是靠长时间积累得来的。不过您也不要灰心，所谓的工作经验无非也就是一些运行在linux系统上的软件的配置以及应用。就好像是装在windows上的office一样，大部分人都会装，但是十分会用的却不多。是因为office太难吗，当然不是，只是因为只有一小部分人花费了很长很长的时间去使用和研究office而已。

LAMP 是Linux Apache MySQL PHP的简写，其实就是把Apache, MySQL以及PHP安装在Linux系统上，组成一个环境来运行php的脚本语言。至于什么是php脚本语言，阿铭不介绍，请自己查资料吧。Apache是最常用的WEB服务软件，而MySQL是比较小型的数据库软件，这两个软件以及PHP都可以安装到windows的机器上。下面阿铭就教您如何构建这个LAMP环境。

18.1 安装MySQL

我们平时安装MySQL都是源码包安装的，但是由于它的编译需要很长的时间，所以，阿铭建议您安装二进制免编译包。您可以到MySQL官方网站去下载 http://dev.mysql.com/downloads/ 具体版本根据您的平台和需求而定，目前比较常用的为mysql-5.0/mysql-5.1, 5.5版本虽然已经发布有段日子了，但是貌似用在线上跑服务的还是少数。所以，阿铭建议您下载一个5.1的版本。可以使用阿铭提供的地址下载。下面是安装步骤：

1. 下载mysql到/usr/local/src/

```
cd /usr/local/src/
wget http://www.lishiming.net/data/attachment/forum/mysql-5.1.40-linux-i686-icc-glibc23.tar.gz
```

2. 解压

```
[root@localhost src]# tar zxvf /usr/local/src/mysql-5.1.40-linux-i686-icc-glibc23.tar.gz
```

3. 把解压完的数据移动到/usr/local/mysql

```
[root@localhost src]# mv mysql-5.1.40-linux-i686-icc-glibc23 /usr/local/mysql
```

4. 建立mysql用户

```
[root@localhost src]# useradd -s /sbin/nologin mysql
```

5. 初始化数据库

```
[root@localhost src]# cd /usr/local/mysql
[root@localhost mysql]# mkdir -p /data/mysql ; chown -R mysql:mysql /data/mysql
[root@localhost mysql]# ./scripts/mysql_install_db --user=mysql --datadir=/data/mysql
```

--user 定义数据库的所属主， --datadir 定义数据库安装到哪里，建议放到大空间的分区上，这个目录需要自行创建。这一步骤很关键，如果您看到两个 ``OK'' 说明执行正确，否则请仔细查看错误信息，如果您实在解决不了，请把问题发到论坛教程答疑版块(http://www.lishiming.net/forum-40-1.html)阿铭会来帮您解决问题。

6. 拷贝配置文件

```
[root@localhost mysql]# cp support-files/my-large.cnf /etc/my.cnf
```

7. 拷贝启动脚本文件并修改其属性

```
[root@localhost mysql]# cp support-files/mysql.server  /etc/init.d/mysqld
[root@localhost mysql]# chmod 755 /etc/init.d/mysqld
```

8. 修改启动脚本

```
[root@localhost mysql]# vim /etc/init.d/mysqld
```

需要修改的地方有 ``datadir=/data/mysql'' （前面初始化数据库时定义的目录）

9. 把启动脚本加入系统服务项，并设定开机启动，启动mysql

```
[root@localhost mysql]# chkconfig --add mysqld
[root@localhost mysql]# chkconfig mysqld on
[root@localhost mysql]# service mysqld start
```

如果启动不了，请到 /data/mysql/ 下查看错误日志，这个日志通常是主机名.err. 检查mysql是否启动的命令为:

```
[root@localhost mysql]# ps aux |grep mysqld
```

18.2 安装Apache

同样apache也需要到官网下载合适的版本，目前使用较多的版本为2.0或者2.2阿铭建议下载2.2版本。apache官网下载地址： http://www.apache.org/dyn/closer.cgi 您也可以使用阿铭提供的地址下载。

```
[root@localhost mysql]# cd /usr/local/src/
[root@localhost src]# wget  http://www.lishiming.net/data/attachment/forum/httpd-2.2.24.tar.bz2
```

解压:

```
[root@localhost src]# tar jvxf httpd-2.2.24.tar.bz2
```

配置编译参数:

```
[root@localhost src]# cd httpd-2.2.24
[root@localhost httpd-2.2.24]# ./configure \
--prefix=/usr/local/apache2 \
--with-included-apr \
--enable-so \
--enable-deflate=shared \
--enable-expires=shared \
```

```
--enable-rewrite=shared \
--with-pcre
```

--prefix 指定安装到哪里， --enable-so 表示启用DSO [1] --enable-deflate=shared 表示共享的方式编译deflate，后面的参数同理。如果这一步您出现了这样的错误:

```
error: mod_deflate has been requested but can not be built due to prerequisite failures
```

解决办法是:

```
yum install -y zlib-devel
```

为了避免在make的时候出现错误，所以最好是提前先安装好一些库文件:

```
yum install -y pcre pcre-devel apr apr-devel
```

编译:

```
[root@localhost httpd-2.2.24]# make
```

安装:

```
[root@localhost httpd-2.2.24]# make install
```

以上两个步骤都可以使用 echo $? 来检查是否正确执行，否则需要根据错误提示去解决问题。

18.3 安装PHP

阿铭写这本教程时，php当前最新版本为5.5, 相信大多网站还在跑着5.2甚至更老的版本，其实5.2版本的php很经典也很稳定，因为阿铭的公司一直在使用5.2版本，但是考虑到版本太老，难免会有些漏洞，所以建议您使用5.3或者5.4版本，php官方下载地址: http://www.php.net/downloads.php 当前php5.3的稳定版本为php-5.3.27.

下载php:

```
[rot@localhost httpd-2.2.24]# cd /usr/local/src
[root@localhost src]# wget http://am1.php.net/distributions/php-5.3.27.tar.gz
```

解压:

```
[root@localhost src]# tar zxf php-5.3.27.tar.gz
```

配置编译参数:

```
[root@localhost src]# cd php-5.3.27
[root@localhost php-5.3.27]# ./configure \
--prefix=/usr/local/php \
--with-apxs2=/usr/local/apache2/bin/apxs \
--with-config-file-path=/usr/local/php/etc  \
--with-mysql=/usr/local/mysql \
--with-libxml-dir \
--with-gd \
--with-jpeg-dir \
```

[1] DSO是Dynamic Shared Objects（动态共享目标）的缩写，它提供了一种在运行时将特殊格式的代码在程序运行需要时，将需要的部分从外存调入内存执行的方法。Apache 支持动态共享模块，也支持静态模块，静态的话，会把需要的目标直接编译进apache的可执行文件中，相比较动态，虽然省去了加载共享模块的步骤，但是也加大了二进制执行文件的空间，变得臃肿。

```
--with-png-dir \
--with-freetype-dir \
--with-iconv-dir \
--with-zlib-dir \
--with-bz2 \
--with-openssl \
--with-mcrypt \
--enable-soap \
--enable-gd-native-ttf \
--enable-mbstring \
--enable-sockets \
--enable-exif \
--disable-ipv6
```

在这一步，阿铭遇到如下错误:

```
configure: error: xml2-config not found. Please check your libxml2 installation.
```

解决办法是:

```
yum install -y libxml2-devel
```

还有错误:

```
configure: error: Cannot find OpenSSL's <evp.h>
```

解决办法是:

```
yum install -y openssl openssl-devel
```

错误:

```
checking for BZip2 in default path... not found
configure: error: Please reinstall the BZip2 distribution
```

解决办法:

```
yum install -y bzip2 bzip2-devel
```

错误:

```
configure: error: png.h not found.
```

解决办法:

```
yum install -y libpng libpng-devel
```

错误:

```
configure: error: freetype.h not found.
```

解决办法:

```
yum install -y freetype freetype-devel
```

错误:

```
configure: error: mcrypt.h not found. Please reinstall libmcrypt.
```

解决办法:

```
rpm -ivh "http://www.lishiming.net/data/attachment/forum/month_1211/epel-release-6-7.noarch.rpm"
yum install -y  libmcrypt-devel
```

因为centos6.x 默认的yum源没有libmcrypt-devel 这个包，只能借助第三方yum源。

编译:

```
[root@localhost php-5.3.27]# make
```

在这一步，您也许还会遇到诸多错误，没有关系，请仔细查看报错信息，解决办法很简单，就是装缺少的库。您可以把错误信息复制到google上搜一下，如果实在是解决不了，请到阿铭论坛(http://www.lishiming.net/forum-40-1.html)发帖请教阿铭吧。

安装:

```
[root@localhost php-5.3.27]# make install
```

拷贝配置文件:

```
[root@localhost php-5.3.27]# cp php.ini-production /usr/local/php/etc/php.ini
```

18.4 apache结合php

Apache主配置文件为：/usr/local/apache2/conf/httpd.conf

```
vim /usr/local/apache2/conf/httpd.conf
```

找到:

```
AddType application/x-gzip .gz .tgz
```

在该行下面添加:

```
AddType application/x-httpd-php .php
```

找到:

```
<IfModule dir_module>
    DirectoryIndex index.html
</IfModule>
```

将该行改为:

```
<IfModule dir_module>
    DirectoryIndex index.html index.htm index.php
</IfModule>
```

找到:

```
#ServerName www.example.com:80
```

修改为:

```
ServerName localhost:80
```

18.5 测试LAMP是否成功

启动apache之前先检验配置文件是否正确:

```
/usr/local/apache2/bin/apachectl -t
```

如果有错误，请继续修改httpd.conf, 如果是正确的则显示为 ``Syntax OK'', 启动apache的命令为:

```
/usr/local/apache2/bin/apachectl start
```

查看是否启动:

```
[root@localhost ~]# netstat -lnp |grep httpd
tcp        0      0 :::80                       :::*   LISTEN      7667/httpd
```

如果有显示这行，则启动了。 也可以使用curl命令简单测试:

```
[root@localhost ~]# curl localhost
<html><body><h1>It works!</h1></body></html>
```

只有显示这样才正确。

测试是否正确解析php:

```
vim /usr/local/apache2/htdocs/1.php
```

写入:

```
<?php
    echo "php解析正常";
?>
```

保存后，继续测试:

```
curl localhost/1.php
```

看是否能看到如下信息:

```
[root@localhost ~]# curl localhost/1.php
php解析正常[root@localhost ~]#
```

只有显示如阿铭这样才正确。

LAMP环境是搭建好了，这其实仅仅是安装上了软件而已，而具体的配置还是有很多工作要做的呢？也就是说，您虽然搭建出来了环境，但是如果不会配置细节的东西，相当于没有任何工作经验，所以还是多配置配置apache或者php吧，具体参考资料可以到阿铭论坛的相应版本中找到，大多帖子为阿铭工作中所配置过的，阿铭真心希望您能够按照阿铭的帖子配置一下，这样对您有很大的好处。论坛地址： http://www.lishiming.net/forum.php

阿铭建议您最好再扩展学习一下: http://www.lishiming.net/thread-5441-1-1.html

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

EIGHTEEN

第十八章 LNMP环境搭建

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

和LAMP不同的是LNMP中的N指的是Nginx（类似于Apache的一种web服务软件）其他都一样。目前这种环境应用的也是非常之多。Nginx设计的初衷是提供一种快速高效多并发的web服务软件。在静态页面的处理上Nginx的确胜Apache一筹，然而在动态页面的处理上Nginx并不比Apache有多少优势。但是，目前还是有很多爱好者对Nginx比较热衷，随着Nginx的技术逐渐成熟，它在web服务软件领域的地位越来越高。

19.1 安装MySQL

步骤和上一章LAMP中的mysql安装步骤(http://study.lishiming.net/chapter17.html#mysql)是一模一样的。

我们平时安装MySQL都是源码包安装的，但是由于它的编译需要很长的时间，所以，阿铭建议您安装二进制免编译包。您可以到MySQL官方网站去下载 http://dev.mysql.com/downloads/ 具体版本根据您的平台和需求而定，目前比较常用的为mysql-5.0/mysql-5.1, 5.5版本虽然已经发布有段日子了，但是貌似用在线上跑服务的还是少数。所以，阿铭建议您下载一个5.1的版本。可以使用阿铭提供的地址下载。下面是安装步骤：

1. 下载mysql到/usr/local/src/

```
cd /usr/local/src/
wget http://www.lishiming.net/data/attachment/forum/mysql-5.1.40-linux-i686-icc-glibc23.tar.gz
```

2. 解压

```
[root@localhost src]# tar zxvf /usr/local/src/mysql-5.1.40-linux-i686-icc-glibc23.tar.gz
```

3. 把解压完的数据移动到/usr/local/mysql

```
[root@localhost src]# mv mysql-5.1.40-linux-i686-icc-glibc23 /usr/local/mysql
```

4. 建立mysql用户

```
[root@localhost src]# useradd -s /sbin/nologin mysql
```

5. 初始化数据库

```
[root@localhost src]# cd /usr/local/mysql
[root@localhost mysql]# mkdir -p /data/mysql ; chown -R mysql:mysql /data/mysql
[root@localhost mysql]# ./scripts/mysql_install_db --user=mysql --datadir=/data/mysql
```

--user 定义数据库的所属主， --datadir 定义数据库安装到哪里，建议放到大空间的分区上，这个目录需要自行创建。这一步骤很关键，如果您看到两个 ``OK'' 说明执行正确，否则请仔细查看错误信息，如果您实在解决不了，请把问题发到论坛 (http://www.lishiming.net/forum-40-1.html)阿铭会来帮您解决问题。

6. 拷贝配置文件

```
[root@localhost mysql]# cp support-files/my-large.cnf /etc/my.cnf
```

7. 拷贝启动脚本文件并修改其属性

```
[root@localhost mysql]# cp support-files/mysql.server  /etc/init.d/mysqld
[root@localhost mysql]# chmod 755 /etc/init.d/mysqld
```

8. 修改启动脚本

```
[root@localhost mysql]# vim /etc/init.d/mysqld
```

需要修改的地方有 ``datadir=/data/mysql'' （前面初始化数据库时定义的目录）

9. 把启动脚本加入系统服务项，并设定开机启动，启动mysql

```
[root@localhost mysql]# chkconfig --add mysqld
[root@localhost mysql]# chkconfig mysqld on
[root@localhost mysql]# service mysqld start
```

如果启动不了，请到 /data/mysql/ 下查看错误日志，这个日志通常是主机名.err. 检查mysql是否启动的命令为:

```
[root@localhost mysql]# ps aux |grep mysqld
```

19.2 安装php

这里要先声明一下，针对Nginx的php安装和针对apache的php安装是有区别的，因为Nginx中的php是以fastcgi的方式结合nginx的，可以理解为nginx代理了php的fastcgi，而apache是把php作为自己的模块来调用的。同样的，阿铭建议您使用5.3版本。php官方下载地址: http://www.php.net/downloads.php

1. 下载php

```
[rot@localhost httpd-2.2.24]# cd /usr/local/src
[root@localhost src]# wget http://am1.php.net/distributions/php-5.3.27.tar.gz
```

2. 解压php

```
[root@localhost src]# tar zxf php-5.3.27.tar.gz
```

3. 创建相关账户

```
[root@localhost src]# useradd -s /sbin/nologin php-fpm
```

4. 配置编译参数

```
[root@localhost src]# cd php-5.3.27
[root@localhost php-5.3.27]# ./configure \
--prefix=/usr/local/php \
--with-config-file-path=/usr/local/php/etc \
--enable-fpm \
--with-fpm-user=php-fpm \
```

```
--with-fpm-group=php-fpm \
--with-mysql=/usr/local/mysql \
--with-mysql-sock=/tmp/mysql.sock \
--with-libxml-dir \
--with-gd \
--with-jpeg-dir \
--with-png-dir \
--with-freetype-dir \
--with-iconv-dir \
--with-zlib-dir \
--with-mcrypt \
--enable-soap \
--enable-gd-native-ttf \
--enable-ftp \
--enable-mbstring \
--enable-exif \
--enable-zend-multibyte \
--disable-ipv6 \
--with-pear \
--with-curl \
--with-openssl
```

该过程中，如果出现如下错误，请按照阿铭给出的解决办法解决，如果出现的错误阿铭并没有写出来，请参考上一章LAMP的php安装步骤(http://study.lishiming.net/chapter17.html#php)

错误信息:

```
configure: error: Please reinstall the libcurl distribution -
    easy.h should be in <curl-dir>/include/curl/
```

解决办法:

```
yum install -y libcurl-devel
```

5. 编译php

```
[root@localhost  php-5.3.27]# make
```

在这一步，您通常会遇到一些错误，没有关系，遇到错误是好事，这样可以增加您处理问题的经验。阿铭同样也遇到了错误:

```
/usr/bin/ld: cannot find -lltdl
collect2: ld returned 1 exit status
make: *** [sapi/fpm/php-fpm] 错误 1
```

阿铭是这样解决的:

```
yum install -y libtool-ltdl-devel
```

6. 安装php

```
[root@localhost  php-5.3.27]# make install
```

以上每一个步骤，如果没有完全执行正确，那么下一步是无法进行的，是否还记得判断执行是否正确的方法？ 使用 echo $? 看结果是否为 ``0''，如果不是，就是没有执行正确。

7. 修改配置文件

```
cp php.ini-production /usr/local/php/etc/php.ini
vim /usr/local/php/etc/php-fpm.conf
```

把如下内容写入该文件:

```
[global]
pid = /usr/local/php/var/run/php-fpm.pid
error_log = /usr/local/php/var/log/php-fpm.log
[www]
listen = /tmp/php-fcgi.sock
user = php-fpm
group = php-fpm
pm = dynamic
pm.max_children = 50
pm.start_servers = 20
pm.min_spare_servers = 5
pm.max_spare_servers = 35
pm.max_requests = 500
rlimit_files = 1024
```

保存配置文件后，检验配置是否正确的方法为:

```
/usr/local/php/sbin/php-fpm -t
```

如果出现诸如 ``test is successful'' 字样，说明配置没有问题。

8. 启动php-fpm

```
cp /usr/local/src/php-5.3.27/sapi/fpm/init.d.php-fpm /etc/init.d/php-fpm
chmod 755 /etc/init.d/php-fpm
service php-fpm start
```

如果想让它开机启动，执行:

```
chkconfig php-fpm on
```

检测是否启动:

```
ps aux |grep php-fpm
```

看看是不是有很多个进程（大概20多个）。

19.3 安装nginx

Nginx官方网站(http://nginx.org), 从官方网站可以看到nginx更新速度很快，这也反映了一个事实，目前使用nginx跑网站的公司或者个人越来越多。当前最新版本为1.5, 但是阿铭不建议您安装这么新的，因为它还太新，难免会有一些bug或者漏洞，所以阿铭建议您安装1.4版本的nginx.

(近期nginx报出一个安全漏洞，影响版本很广 CVE-2013-4547，所以之前的老版本都需要升级一下, 1.4.4, 1.5.7以及往后版本没有问题)

1. 下载nginx

```
cd /usr/local/src/
wget http://nginx.org/download/nginx-1.4.4.tar.gz
```

2. 解压nginx

```
tar zxvf nginx-1.4.4.tar.gz
```

3. 配置编译参数

```
cd nginx-1.4.4
./configure \
--prefix=/usr/local/nginx \
--with-http_realip_module \
--with-http_sub_module \
--with-http_gzip_static_module \
--with-http_stub_status_module  \
--with-pcre
```

4. 编译nginx

```
make
```

5. 安装nginx

```
make install
```

因为nginx比较小，所以很快就会安装完，而且也不会出什么错误，如果出错了，到阿铭论坛(http://www.lishiming.net/forum-40-1.html)发帖求助阿铭吧。

6. 编写nginx启动脚本，并加入系统服务

```
vim /etc/init.d/nginx
```

写入如下内容:

```
#!/bin/bash
# chkconfig: - 30 21
# description: http service.
# Source Function Library
. /etc/init.d/functions
# Nginx Settings

NGINX_SBIN="/usr/local/nginx/sbin/nginx"
NGINX_CONF="/usr/local/nginx/conf/nginx.conf"
NGINX_PID="/usr/local/nginx/logs/nginx.pid"
RETVAL=0
prog="Nginx"

start() {
        echo -n $"Starting $prog: "
        mkdir -p /dev/shm/nginx_temp
        daemon $NGINX_SBIN -c $NGINX_CONF
        RETVAL=$?
        echo
        return $RETVAL
}

stop() {
        echo -n $"Stopping $prog: "
        killproc -p $NGINX_PID $NGINX_SBIN -TERM
        rm -rf /dev/shm/nginx_temp
        RETVAL=$?
        echo
        return $RETVAL
```

```
}

reload(){
        echo -n $"Reloading $prog: "
        killproc -p $NGINX_PID $NGINX_SBIN -HUP
        RETVAL=$?
        echo
        return $RETVAL
}

restart(){
        stop
        start
}

configtest(){
    $NGINX_SBIN -c $NGINX_CONF -t
    return 0
}

case "$1" in
  start)
        start
        ;;
  stop)
        stop
        ;;
  reload)
        reload
        ;;
  restart)
        restart
        ;;
  configtest)
        configtest
        ;;
  *)
        echo $"Usage: $0 {start|stop|reload|restart|configtest}"
        RETVAL=1
esac

exit $RETVAL
```

保存后，更改权限:

```
chmod 755 /etc/init.d/nginx
chkconfig --add nginx
```

如果想开机启动，请执行:

```
chkconfig nginx on
```

7. 更改nginx配置

首先把原来的配置文件清空:

```
> /usr/local/nginx/conf/nginx.conf
```

``>'' 这个符号之前阿铭介绍过，为重定向的意思，单独用它，可以把一个文本文档快速清空。

```
vim /usr/local/nginx/conf/nginx.conf
```

写入如下内容:

```
user nobody nobody;
worker_processes 2;
error_log /usr/local/nginx/logs/nginx_error.log crit;
pid /usr/local/nginx/logs/nginx.pid;
worker_rlimit_nofile 51200;

events
{
    use epoll;
    worker_connections 6000;
}

http
{
    include mime.types;
    default_type application/octet-stream;
    server_names_hash_bucket_size 3526;
    server_names_hash_max_size 4096;
    log_format combined_realip '$remote_addr $http_x_forwarded_for [$time_local]'
    '$host "$request_uri" $status'
    '"$http_referer" "$http_user_agent"';
    sendfile on;
    tcp_nopush on;
    keepalive_timeout 30;
    client_header_timeout 3m;
    client_body_timeout 3m;
    send_timeout 3m;
    connection_pool_size 256;
    client_header_buffer_size 1k;
    large_client_header_buffers 8 4k;
    request_pool_size 4k;
    output_buffers 4 32k;
    postpone_output 1460;
    client_max_body_size 10m;
    client_body_buffer_size 256k;
    client_body_temp_path /usr/local/nginx/client_body_temp;
    proxy_temp_path /usr/local/nginx/proxy_temp;
    fastcgi_temp_path /usr/local/nginx/fastcgi_temp;
    fastcgi_intercept_errors on;
    tcp_nodelay on;
    gzip on;
    gzip_min_length 1k;
    gzip_buffers 4 8k;
    gzip_comp_level 5;
    gzip_http_version 1.1;
    gzip_types text/plain application/x-javascript text/css text/htm application/xml;

server
{
    listen 80;
    server_name localhost;
    index index.html index.htm index.php;
    root /usr/local/nginx/html;
```

```
    location ~ \.php$ {
        include fastcgi_params;
        fastcgi_pass unix:/tmp/php-fcgi.sock;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME /usr/local/nginx/html$fastcgi_script_name;
    }

}

}
```

保存配置后，先检验一下配置文件是否有错误存在:

```
/usr/local/nginx/sbin/nginx  -t
```

如果显示内容如下，则配置正确，否则需要根据错误提示修改配置文件:

```
nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok
nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful
```

启动nginx:

```
service nginx start
```

如果不能启动，请查看 ``/usr/local/nginx/logs/error.log'' 文件，检查nginx是否启动:

```
ps aux |grep nginx
```

看是否有进程。

19.4 测试是否解析php文件

创建测试文件:

```
vim /usr/local/nginx/html/2.php
```

内容如下:

```
<?php
    echo "测试php是否解析";
?>
```

测试:

```
[root@localhost nginx]# curl localhost/2.php
测试php是否解析[root@localhost nginx]#
```

显示成阿铭这样，才说明php解析正确。

到这里，LNMP环境就算介绍完了。但是您掌握的技能还远远不够日常工作中处理问题，所以阿铭建议您还是多多的去实践一下。阿铭论坛里(http://www.lishiming.net/)有很多相关的帖子，如果有兴趣请多看一看，做一做，对您将来找工作有极大的好处。

阿铭建议您最好再扩展学习一下: http://www.lishiming.net/thread-5442-1-1.html

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

NINETEEN

第十九章 学会使用简单的MYSQL操作

在前面两个章节中已经介绍过MySQL的安装了，但是光会安装还不够，您还需要会一些基本的相关操作。当然了，关于MySQL的内容也是非常多的，只不过对于linux系统管理员来讲，一些基本的操作已经可以应付日常的管理工作了，至于更高深的那是DBA（专门管理数据库的技术人员）的事情了。

20.1 更改mysql数据库root的密码

首次进入数据库是不用密码的:

```
[root@localhost ~]# /usr/local/mysql/bin/mysql -uroot
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.1.40-log MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

退出的话，直接输入quit或者exit即可。细心的读者也许会发现，阿铭在上一条命令中，使用的是绝对路径，这样不方便，但是单独只是输入一个 ``mysql'' 命令是不行的，因为 ``/usr/local/mysql/bin'' 没有在 PATH 这个环境变量里。如何把它加入环境变量PATH中？之前阿铭介绍过:

```
[root@localhost ~]# PATH=$PATH:/usr/local/mysql/bin
```

这样就可以了，但重启Linux后还会失效，所以需要让它开机加载:

```
[root@localhost ~]# echo "PATH=$PATH:/usr/local/mysql/bin" >> /etc/profile
[root@localhost ~]# source /etc/profile
[root@localhost ~]# mysql -uroot
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.1.40-log MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

阿铭再来解释一下上一条命令 -u 的含义，它用来指定要登录的用户，后边可以有空格，也可以无空格，root用户是mysql自带的管理员账户，默认没有密码的，那么如何给root用户设定密码？按如下操作:

```
[root@localhost ~]# mysqladmin -uroot password 'yourpassword'
```

这样就设置了 `root' 账号的密码了，不妨再来用上面的命令登陆一下试试看:

```
[root@localhost ~]# mysql -uroot
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password:NO)
```

报错了，这是在提示我们，root账号是需要密码登陆的。

```
[root@localhost ~]# mysql -uroot -p'yourpassword'
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.1.40-log MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

需要加一个 -p 选项，它后面可以直接跟密码，后面不可以有空格，不过密码最好用单引号括起来，不括也可以，但是密码中如果有特殊字符就会有问题了，所以最好是括起来吧。当然， -p后面也是可以不加密码，而是和用户交互的方式，让我们输入密码:

```
[root@localhost ~]# mysql -uroot -p
Enter password:
```

20.2 连接数据库

刚刚讲过通过使用 `mysql -u root -p` 就可以连接数据库了，但这只是连接的本地的数据库 ``localhost'', 可是有很多时候都是去连接网络中的某一个主机上的mysql。

```
[root@localhost ~]# mysql -uroot -p -h192.168.137.10 -P3306
Enter password:
```

其中后边的 -P(大写) 用来指定远程主机mysql的绑定端口，默认都是3306, -h 用来指定远程主机的IP.

20.3 一些基本的MySQL操作命令

1. 查询当前的库

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| test               |
+--------------------+
3 rows in set (0.06 sec)
```

mysql的命令，结尾处需要加一个分号。

2. 查询某个库的表

首先需要切换到某个库里去:

```
mysql> use mysql;
Database changed
```

然后再把表列出来:

```
mysql> show tables;
+---------------------------+
| Tables_in_mysql           |
+---------------------------+
| columns_priv              |
| db                        |
| event                     |
| func                      |
| general_log               |
| help_category             |
| help_keyword              |
| help_relation             |
| help_topic                |
| host                      |
| ndb_binlog_index          |
| plugin                    |
| proc                      |
| procs_priv                |
| servers                   |
| slow_log                  |
| tables_priv               |
| time_zone                 |
| time_zone_leap_second     |
| time_zone_name            |
| time_zone_transition      |
| time_zone_transition_type |
| user                      |
+---------------------------+
23 rows in set (0.06 sec)
```

3. 查看某个表的全部字段

```
mysql> desc slow_log;
+----------------+------------------+------+-----+-------------------+-----------------------------+
| Field          | Type             | Null | Key | Default           | Extra                       |
+----------------+------------------+------+-----+-------------------+-----------------------------+
| start_time     | timestamp        | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
| user_host      | mediumtext       | NO   |     | NULL              |                             |
| query_time     | time             | NO   |     | NULL              |                             |
| lock_time      | time             | NO   |     | NULL              |                             |
| rows_sent      | int(11)          | NO   |     | NULL              |                             |
| rows_examined  | int(11)          | NO   |     | NULL              |                             |
| db             | varchar(512)     | NO   |     | NULL              |                             |
| last_insert_id | int(11)          | NO   |     | NULL              |                             |
| insert_id      | int(11)          | NO   |     | NULL              |                             |
| server_id      | int(10) unsigned | NO   |     | NULL              |                             |
| sql_text       | mediumtext       | NO   |     | NULL              |                             |
+----------------+------------------+------+-----+-------------------+-----------------------------+
11 rows in set (0.04 sec)
```

也可以使用两一条命令，显示比这个更详细，而且可以把建表语句全部列出来:

```
mysql> show create table slow_log\G;
*************************** 1. row ***************************
```

```
       Table: slow_log
Create Table: CREATE TABLE `slow_log` (
  `start_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  `user_host` mediumtext NOT NULL,
  `query_time` time NOT NULL,
  `lock_time` time NOT NULL,
  `rows_sent` int(11) NOT NULL,
  `rows_examined` int(11) NOT NULL,
  `db` varchar(512) NOT NULL,
  `last_insert_id` int(11) NOT NULL,
  `insert_id` int(11) NOT NULL,
  `server_id` int(10) unsigned NOT NULL,
  `sql_text` mediumtext NOT NULL
) ENGINE=CSV DEFAULT CHARSET=utf8 COMMENT='Slow log'
1 row in set (0.01 sec)
```

4. 查看当前是哪个用户

```
mysql> select user();
+----------------+
| user()         |
+----------------+
| root@localhost |
+----------------+
1 row in set (0.00 sec)
```

5. 查看当前所使用数据库

```
mysql> select database();
+------------+
| database() |
+------------+
| mysql      |
+------------+
1 row in set (0.01 sec)
```

6. 创建一个新库

```
mysql> create database db1;
Query OK, 1 row affected (0.05 sec)
```

7. 创建一个新表

```
mysql> use db1;
Database changed
mysql> create table t1 (`id` int(4), `name` char(40));
Query OK, 0 rows affected (0.02 sec)
```

要注意的是，字段名需要用反引号括起来。

8. 查看当前数据库版本

```
mysql> select version();
+------------+
| version()  |
+------------+
| 5.1.40-log |
+------------+
1 row in set (0.01 sec)
```

9. 查看当前mysql状态

```
mysql> show status;
+----------------------------------+----------+
| Variable_name                    | Value    |
+----------------------------------+----------+
| Aborted_clients                  | 0        |
| Aborted_connects                 | 5        |
| Binlog_cache_disk_use            | 0        |
| Binlog_cache_use                 | 0        |
| Bytes_received                   | 303      |
| Bytes_sent                       | 7001     |
```

由于内容太长，阿铭没有全部列出来，如果有兴趣可以网上找资料查一下每一行的含义。

10. 查看mysql的参数

```
mysql> show variables;
+---------------------------------------+---------------------+
| Variable_name                         | Value               |
+---------------------------------------+---------------------+
| auto_increment_increment              | 1                   |
| auto_increment_offset                 | 1                   |
| autocommit                            | ON                  |
| automatic_sp_privileges               | ON                  |
| back_log                              | 50                  |
| basedir                               | /usr/local/mysql/   |
```

限于篇幅，阿铭省略了很多参数没有显示，其中很多参数都是可以在/etc/my.cnf中定义的，并且有部分参数是可以在线编辑的。

11. 修改mysql的参数

```
mysql> show variables like 'max_connect%';
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| max_connect_errors | 10    |
| max_connections    | 151   |
+--------------------+-------+
2 rows in set (0.00 sec)

mysql> set global max_connect_errors = 1000;
Query OK, 0 rows affected (0.01 sec)

mysql> show variables like 'max_connect_errors';
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| max_connect_errors | 1000  |
+--------------------+-------+
1 row in set (0.01 sec)
```

在mysql命令行， ``%'' 类似于shell下的 *, 表示万能匹配。使用 ``set global'' 可以临时修改某些参数，但是重启mysqld服务后还会变为原来的，所以要想恒久生效，需要在配置文件 my.cnf 中定义。

12. 查看当前mysql服务器的队列

这个在日常的管理工作中使用最为频繁，因为使用它可以查看当前mysql在干什么，可以发现是否有锁表:

```
mysql> show processlist;
+----+------+-----------+------+---------+------+-------+------------------+
| Id | User | Host      | db   | Command | Time | State | Info             |
+----+------+-----------+------+---------+------+-------+------------------+
| 13 | root | localhost | db1  | Query   |    0 | NULL  | show processlist |
+----+------+-----------+------+---------+------+-------+------------------+
1 row in set (0.01 sec)
```

13. 创建一个普通用户并授权

```
mysql> grant all on *.* to user1 identified by '123456';
Query OK, 0 rows affected (0.01 sec)
```

all 表示所有的权限（读、写、查询、删除等等操作），*.* 前面的 * 表示所有的数据库，后面的 * 表示所有的表，identified by 后面跟密码，用单引号括起来。这里的user1指的是localhost上的user1，如果是给网络上的其他机器上的某个用户授权则这样:

```
mysql> grant all on db1.* to 'user2'@'10.0.2.100' identified by '111222';
Query OK, 0 rows affected (0.01 sec)
```

用户和主机的IP之间有一个@，另外主机IP那里可以用%替代，表示所有主机，例如:

```
mysql> grant all on db1.* to 'user3'@'%' identified by '231222';
Query OK, 0 rows affected (0.00 sec)
```

20.4 一些常用的sql

1. 查询语句

```
mysql> select count(*) from mysql.user;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.00 sec)
```

mysql.user表示mysql库的user表；count(*)表示表中共有多少行。

```
mysql> select * from mysql.db;
```

这个用来表示查询mysql库的db表中的所有数据，也可以查询单个字段或者多个字段:

```
mysql> select db from mysql.db;
mysql> select db,user  from mysql.db;
```

同样，在查询语句中可以使用万能匹配 ``%''

```
mysql> select * from mysql.db where host like '10.0.%';
```

2. 插入一行

```
mysql> insert into db1.t1 values (1, 'abc');
Query OK, 1 row affected (0.02 sec)

mysql> select * from db1.t1;
```

```
+------+------+
| id   | name |
+------+------+
|    1 | abc  |
+------+------+
1 row in set (0.00 sec)
```

3. 更改表的某一行

```
mysql> update db1.t1 set name='aaa' where id=1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from db1.t1;
+------+------+
| id   | name |
+------+------+
|    1 | aaa  |
+------+------+
1 row in set (0.00 sec)
```

4. 清空表数据

```
mysql> truncate table db1.t1;
Query OK, 0 rows affected (0.01 sec)

mysql> select count(*) from db1.t1;
+----------+
| count(*) |
+----------+
|        0 |
+----------+
1 row in set (0.00 sec)
```

5. 删除表

```
mysql> drop table db1.t1;
Query OK, 0 rows affected (0.00 sec)
```

6. 删除数据库

```
mysql> drop database db1;
Query OK, 0 rows affected (0.02 sec)
```

20.5 mysql数据库的备份与恢复

备份:

```
[root@localhost ~]# mysqldump  -uroot -p'yourpassword' mysql >/tmp/mysql.sql
```

使用 mysqldump 命令备份数据库，-u 和 -p 两个选项使用方法和前面说的 mysql 同样，而后面的 ``mysql'' 指的是库名，然后重定向到一个文本文档里。备份完后，您可以查看 /tmp/mysql.sql 这个文件里的内容。

恢复和备份正好相反:

```
[root@localhost ~]# mysql -uroot -p'yourpassword' mysql </tmp/mysql.sql
```

关于MySQL的基本操作阿铭就介绍这么多，当然学会了这些还远远不够，希望您能够在工作中学习到更多的知识，如果你对MySQL有很大兴趣，不妨深入研究一下，毕竟多学点总没有坏处。如果想学跟多的东西请去查看MySQL官方中文参考手册(5.1) http://dev.mysql.com/doc/refman/5.1/zh/index.html

阿铭建议您最好再扩展学习一下: http://www.lishiming.net/thread-5443-1-1.html

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

TWENTY

第二十章 NFS服务配置

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

NFS会经常用到，用于在网络上共享存储。这样讲，您对NFS可能不太了解，阿铭举一个例子来说明一下NFS是用来做什么的。假如有三台机器A, B, C，它们需要访问同一个目录，目录中都是图片，传统的做法是把这些图片分别放到A, B, C. 但是使用NFS只需要放到A上，然后A共享给B和C即可。访问的时候，B和C是通过网络的方式去访问A上的那个目录的。

21.1 服务端配置NFS

CentOS上使用NFS服务，需要安装两个包(nfs-utils和rpcbind)，不过当使用yum安装nfs-utils时会把rpcbind一起安装上:

```
[root@localhost ~]# yum install -y nfs-utils
```

在之前的CentOS版本中，是需要安装portmap包的，从CentOS6开始，就改为rpmbind了。NFS配置起来还是蛮简单的，只需要编辑配置文件/etc/exports即可。下面阿铭就先创建一个简单的NFS服务器。

首先是修改配置文件，默认该文件为空，现在编辑它:

```
[root@localhost ~]# vim /etc/exports
```

写入如下内容:

```
/home/ 192.168.137.0/24(rw,sync,all_squash,anonuid=501,anongid=501)
```

这个配置文件就这样简单一行。共分为三部分，第一部分就是本地要共享出去的目录，第二部分为允许访问的主机（可以是一个IP也可以是一个IP段）第三部分就是小括号里面的，为一些权限选项。关于第三部分，阿铭简单介绍一下：

rw ：读写；

ro ：只读；

sync ：同步模式，内存中数据时时写入磁盘；

async ：不同步，把内存中数据定期写入磁盘中；

no_root_squash ：加上这个选项后，root用户就会对共享的目录拥有至高的权限控制，就像是对本机的目录操作一样。不安全，不建议使用；

root_squash：和上面的选项对应，root用户对共享目录的权限不高，只有普通用户的权限，即限制了root；

all_squash：不管使用NFS的用户是谁，他的身份都会被限定成为一个指定的普通用户身份；

anonuid/anongid ：要和root_squash 以及all_squash一同使用，用于指定使用NFS的用户限定后的uid和gid，前提是本机的/etc/passwd中存在这个uid和gid。

介绍了上面的相关的权限选项后，再来分析一下阿铭刚刚配置的那个/etc/exports文件。其中要共享的目录为/home，信任的主机为192.168.137.0/24这个网段，权限为读写，同步，限定所有使用者，并且限定的uid和gid都为501。

编辑好配置文件后，就该启动NFS服务了:

```
[root@localhost ~]# /etc/init.d/rpcbind start;  /etc/init.d/nfs start
```

在启动nfs服务之前，需要先启动rpcbind服务，之前CentOS老版本中并不是rpcbind, 而是叫做portmap.

21.2 客户端上挂载nfs

客户端在挂载NFS之前，我们需要先看一看服务端都共享了哪些目录，这需要使用showmount命令，但是这个命令是nfs-utils这个包所带的，所以同样需要安装nfs-utils:

```
[root@localhost ~]# yum install -y nfs-utils
```

现在可以看看服务器端都共享了哪些目录了:

```
[root@localhost ~]# showmount -e 192.168.137.10
Export list for 192.168.137.10:
/home 192.168.137.0/24
```

可以看到刚才我们在服务端配置的nfs共享信息。 showmount -e 加IP就可以查看NFS的共享情况，上例中，就可以看到192.168.137.10的共享目录为/home，信任主机为192.168.137.0/24这个网段。

下面在客户端上挂载服务端的nfs:

```
[root@localhost ~]# mount -t nfs 192.168.137.10:/home/ /mnt/
[root@localhost ~]# df -h
文件系统              容量  已用  可用 已用%% 挂载点
/dev/sda3              14G  6.4G  6.7G  50% /
tmpfs                 160M     0  160M   0% /dev/shm
/dev/sda1              97M   27M   66M  29% /boot
/dev/sdb5             989M   19M  920M   3% /home
192.168.137.10:/home/
                      989M   19M  920M   3% /mnt
```

用 df -h 命令可以看到多出来一个/mnt分区，它就是NFS共享的目录了。

在这一章节里，使用的命令不多，另外还有一个常用的命令那就是exportfs，它的常用选项为[-aruv].

-a ：全部挂载或者卸载；

-r ：重新挂载；

-u ：卸载某一个目录；

-v ：显示共享的目录；

使用exportfs命令，当改变/etc/exports配置文件后，不用重启nfs服务直接用这个exportfs即可。接下来阿铭做一个实验，先改一下服务端的配置文件:

```
[root@localhost ~]# vim /etc/exports
```

增加一行:

```
/tmp/ 192.168.137.0/24(rw,sync,no_root_squash)
```

然后服务端上执行命令:

```
[root@localhost ~]# exportfs -arv
exporting 192.168.137.0/24:/tmp
exporting 192.168.137.0/24:/home
```

在之前的命令中用到了mount命令来挂载nfs，其实mount这个nfs服务还是有些说法的。首先是用-t nfs来指定挂载的类型为nfs。另外在使用nfs时，常用一个选项就是 -o nolock 了，即在挂载nfs服务时，不加锁。 在客户端上执行:

```
[root@localhost ~]# mkdir /test
[root@localhost ~]# mount -t nfs -o nolock 192.168.137.10:/tmp/ /test/
```

我们还可以把要挂载的nfs目录写到client上的/etc/fstab文件中，挂载时只需要执行 mount -a 即可。在/etc/fstab里加一行:

```
192.168.137.10:/tmp/            /test        nfs       nolock  0 0
```

因为刚刚挂载过，所以先卸载:

```
[root@localhost ~]# umount  /test/
```

然后执行:

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

这样也可以挂载上，而且以后开机会自动挂载上。

关于NFS部分就讲这么多，内容并不多，相信您很快就能掌握！

阿铭建议您最好再扩展学习一下: http://www.lishiming.net/thread-5444-1-1.html

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

TWENTYONE

第二十一章 配置FTP服务

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

也许您对FTP不陌生，但是您是否了解FTP到底是个什么玩意？ FTP 是File Transfer Protocol（文件传输协议）的英文简称，而中文简称为 ``文传协议'' 用于Internet上的控制文件的双向传输。同时，它也是一个应用程序（Application）。用户可以通过它把自己的PC机与世界各地所有运行FTP协议的服务器相连，访问服务器上的大量程序和信息。FTP的主要作用，就是让用户连接上一个远程计算机（这些计算机上运行着FTP服务器程序）查看远程计算机有哪些文件，然后把文件从远程计算机上拷到本地计算机，或把本地计算机的文件送到远程计算机去。FTP用的比NFS更多，所以请您一定要熟练配置它。

其实在CentOS或者RedHat Linux上有自带的ftp软件叫做vsftp, 但阿铭介绍的并不是它，如果您有兴趣可以和阿铭交流，阿铭本章使用pure-ftpd搭建ftp服务器，因为这个软件比vsftp配置起来更加灵活和安全。

22.1 安装pure-ftpd

1. 下载软件

pure-ftpd 官网是 http://www.pureftpd.org/project/pure-ftpd 当前最新版本为1.0.36, 但阿铭不建议使用最新版本，最新版有可能有一些小bug.

```
[root@localhost ~]# cd /usr/local/src/
[root@localhost src]# wget http://download.pureftpd.org/pub/pure-ftpd/releases/pure-ftpd-1.0.32.tar.bz2
```

2. 安装pure-ftpd

```
[root@localhost src]# tar jxf pure-ftpd-1.0.32.tar.bz2
[root@localhost src]# cd pure-ftpd-1.0.32
[root@localhost pure-ftpd-1.0.32]# ./configure \
--prefix=/usr/local/pureftpd \
--without-inetd \
--with-altlog \
--with-puredb \
--with-throttling \
--with-peruserlimits  \
--with-tls
[root@localhost pure-ftpd-1.0.32]# make && make install
```

22.2 配置pure-ftpd

1. 修改配置文件

pure-ftpd 编译安装很快就完成了，而且极少有出现错误的时候，下面就该配置它了:

```
[root@localhost pure-ftpd-1.0.32]# cd configuration-file
[root@localhost pure-ftpd-1.0.32]# mkdir -p /usr/local/pureftpd/etc/
[root@localhost configuration-file]# cp pure-ftpd.conf    /usr/local/pureftpd/etc/pure-ftpd.conf
[root@localhost configuration-file]# cp pure-config.pl    /usr/local/pureftpd/sbin/pure-config.pl
[root@localhost configuration-file]# chmod 755    /usr/local/pureftpd/sbin/pure-config.pl
```

在启动pure-ftpd之前需要先修改配置文件，配置文件为/usr/local/pureftpd/etc/pure-ftpd.conf, 您可以打开看一下，里面内容很多，如果英文好，可以好好研究一番，下面是阿铭的配置文件，如果您嫌麻烦，直接拷贝过去即可:

```
ChrootEveryone              yes
BrokenClientsCompatibility  no
MaxClientsNumber            50
Daemonize                   yes
MaxClientsPerIP             8
VerboseLog                  no
DisplayDotFiles             yes
AnonymousOnly               no
NoAnonymous                 no
SyslogFacility              ftp
DontResolve                 yes
MaxIdleTime                 15
PureDB                        /usr/local/pureftpd/etc/pureftpd.pdb
LimitRecursion              3136 8
AnonymousCanCreateDirs      no
MaxLoad                     4
AntiWarez                   yes
Umask                       133:022
MinUID                      100
AllowUserFXP                no
AllowAnonymousFXP           no
ProhibitDotFilesWrite       no
ProhibitDotFilesRead        no
AutoRename                  no
AnonymousCantUpload         no
PIDFile                     /usr/local/pureftpd/var/run/pure-ftpd.pid
MaxDiskUsage               99
CustomerProof              yes
```

2. 启动pure-ftpd

```
[root@localhost ~]# /usr/local/pureftpd/sbin/pure-config.pl /usr/local/pureftpd/etc/pure-ftpd.conf
```

如果是启动成功，会显示一行长长的以Running开头的信息，否则那就是错误信息，如果您解决不了，请到阿铭论坛(http://www.lishiming.net/forum-40-1.html)获取帮助吧。

3. 建立账号

```
[root@localhost ~]# mkdir /data/www/
[root@localhost ~]# useradd www
[root@localhost ~]# chown -R www:www /data/www/
[root@localhost ~]# /usr/local/pureftpd/bin/pure-pw useradd ftp_user1  -uwww -d /data/www/
Password:
Enter it again:
```

其中，-u将虚拟用户ftp_user1与系统用户www关联在一起，也就是说使用ftp_user1账号登陆ftp后，会以www的身份来读取文件或下载文件。-d 后边的目录为ftp_user1账户的家目录，这样可以使ftp_user1只能

访问其家目录/data/www/. 到这里还未完成，还有最关键的一步，就是创建用户信息数据库文件:

```
[root@localhost ~]#  /usr/local/pureftpd/bin/pure-pw mkdb
```

pure-pw还可以列出当前的ftp账号，当然也可以删除某个账号, 我们再创建一个账号:

```
[root@localhost ~]#  /usr/local/pureftpd/bin/pure-pw  useradd ftp_user2 -uwww -d /tmp
[root@localhost ~]#  /usr/local/pureftpd/bin/pure-pw mkdb
```

列出当前账号:

```
[root@localhost ~]# /usr/local/pureftpd/bin/pure-pw list
```

删除账号的命令为:

```
[root@localhost ~]#  /usr/local/pureftpd/bin/pure-pw  userdel ftp_user2
```

22.3 测试pure-ftpd

测试需要使用的工具叫做lftp, 先安装一下它:

```
[root@localhost ~]# yum install -y lftp
```

测试:

```
[root@localhost ~]# touch /data/www/123.txt
[root@localhost ~]# lftp ftp_user1@127.0.0.1
口令:
lftp ftp_user1@127.0.0.1:~> ls
drwxr-xr-x    2 514        www             4096 Jun 12 11:14 .
drwxr-xr-x    2 514        www             4096 Jun 12 11:14 ..
-rw-r--r--    1 514        www                0 Jun 12 11:14 123.txt
```

登陆后，使用 ls 命令可以列出当前目录都有什么文件。

阿铭建议您最好再扩展学习一下: http://www.lishiming.net/thread-5445-1-1.html

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

TWENTYTWO

第二十二章 配置SQUID服务

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

这一章，阿铭介绍一下squid, 它大多用作http服务的缓存服务器，缓存图片等静态文件可以加速客户端的请求返回速度。当然squid的功能可不仅仅局限于这么一小块。

23.1 Squid是什么

Squid是比较知名的代理软件，它不仅可以跑在linux上还可以跑在windows以及Unix上，它的技术已经非常成熟。目前使用Squid的用户也是十分广泛的。Squid与Linux下其它的代理软件如Apache、Socks、TIS FWTK和delegate相比，下载安装简单，配置简单灵活，支持缓存和多种协议。

Squid之所以用的很多，是因为它的缓存功能，Squid缓存不仅可以节省宝贵的带宽资源，也可以大大降低服务器的I/O. 从经济角度考虑，它是很多网站架构中不可或缺的角色。

Squid不仅可以做正向代理，又可以做反向代理。当作为正向代理时，Squid后面是客户端，客户端想上网不管什么网都得经过Squid. 当一个用户(客户端)想要请求一个主页时，它向Squid发出一个申请，要Squid替它请求，然后Squid 连接用户要请求的网站并请求该主页，接着把该主页传给用户同时保留一个备份，当别的用户请求同样的页面时，Squid把保存的备份立即传给用户，使用户觉得速度相当快。使用正向代理时，客户端需要做一些设置，才能实现，也就是平时我们在IE选项中设置的那个代理。而反向代理是，Squid后面为某个站点的服务器，客户端请求该站点时，会先把请求发送到Squid上，然后Squid去处理用户的请求动作。阿铭教您一个特别容易的区分：正向代理，Squid后面是客户端，客户端上网要通过Squid去上；反向代理，Squid后面是服务器，服务器返回给用户数据需要走Squid.

也许您会问，什么时候需要配置正向代理，又什么时候配置反向代理呢？阿铭的观点是，正向代理用在企业的办公环境中，员工上网需要通过Squid代理来上网，这样可以节省网络带宽资源。而反向代理用来搭建网站静态项(图片、html、流媒体、js、css等)的缓存服务器，它用于网站架构中。

23.2 搭建Squid正向代理

CentOS系统自带Squid包，但是需要安装一下:

```
[root@localhost ~]# yum install -y squid
```

当然您也可以源码包编译安装，Squid官方网站为 http://www.squid-cache.org/ 当前最新版本为3.3, 下载3.1版本即可，因为CentOS6上提供的版本也为3.1版本。如果您想编译安装Squid, 请参考阿铭提供的编译参数吧:

```
./configure --prefix=/usr/local/squid \
--disable-dependency-tracking \
--enable-dlmalloc \
--enable-gnuregex \
--disable-carp \
--enable-async-io=240 \
--with-pthreads \
--enable-storeio=ufs,aufs,diskd,null \
--disable-wccp \
--disable-wccpv2 \
--enable-kill-parent-hack \
--enable-cachemgr-hostname=localhost \
--enable-default-err-language=Simplify_Chinese \
--with-build-environment=POSIX_V6_ILP32_OFFBIG \
--with-maxfd=65535 \
--with-aio \
--disable-poll \
--enable-epoll \
--enable-linux-netfilter \
--enable-large-cache-files \
--disable-ident-lookups \
--enable-default-hostsfile=/etc/hosts \
--with-dl \
--with-large-files \
--enable-removal-policies=heap,lru \
--enable-delay-pools \
--enable-snmp \
--disable-internal-dns
```

这些参数不见得符合您的需求，阿铭只是提供一个参考，也许您在编译的过程中会遇到诸多错误，没有关系，请google或者到阿铭论坛(http://www.lishiming.net/forum-40-1.html)发帖求助。阿铭觉得使用CentOS中自带的squid已经满足需求，所以没有编译安装。

安装完后，可以查看squid版本:

```
[root@localhost ~]# squid -v
Squid Cache: Version 3.1.10
```

同时还可以看到squid的编译参数。下面阿铭需要配置一下squid, 来实现正向代理:

```
[root@localhost ~]# rm -f /etc/squid/squid.conf
[root@localhost ~]# vim /etc/squid/squid.conf
```

我们不使用默认配置文件，删除，重新写入如下配置:

```
http_port 3128
acl manager proto cache_object
acl localhost src 127.0.0.1/32 ::1
acl to_localhost dst 127.0.0.0/8 0.0.0.0/32 ::1
acl localnet src 10.0.0.0/8     # RFC1918 possible internal network
acl localnet src 172.16.0.0/12  # RFC1918 possible internal network
acl localnet src 192.168.0.0/16 # RFC1918 possible internal network
acl SSL_ports port 443
acl Safe_ports port 80 8080         # http
acl Safe_ports port 21          # ftp
acl Safe_ports port 443         # https
acl CONNECT method CONNECT
http_access allow manager localhost
http_access deny manager
```

```
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow localnet
http_access allow localhost
http_access allow all
cache_dir aufs /data/cache 1024 16 256
cache_mem 128 MB
hierarchy_stoplist cgi-bin ?
coredump_dir /var/spool/squid
refresh_pattern ^ftp:           1440    20%     10080
refresh_pattern ^gopher:        1440    0%      1440
refresh_pattern -i (/cgi-bin/|\?) 0     0%      0
refresh_pattern \.(jpg|png|gif|mp3|xml) 1440    50%     2880    ignore-reload
refresh_pattern .               0       20%     4320
```

配置文件中有几处阿铭要简单描述一下，第一行的 ``http_port 3128'' 这个指的是，squid服务启动后将要监听的端口，也可以是80. ``cache_dir'' 这个用来指定本地磁盘上的缓存目录，后边的1024为大小，单位是M，具体根据您的磁盘大小决定。 ``cache_mem'' 它用来规定缓存占用内存的大小，即把缓存的东西存到内存里，具体也需要根据您机器的内存定，如果您的机器只是跑Squid服务，那么留给系统512M内存外，其他可以都分给squid, 但阿铭做实验的虚拟机一共才300M内存，所以只分了128M.

配置文件保存好后，可以先检测一下是否有语法错误:

```
[root@localhost ~]# squid -kcheck
```

如果提示信息为:

```
squid: ERROR: No running copy
```

这是说squid还未启动，没有关系，显示成这样说明配置文件没有问题了。在启动前还得再做一件事，就是初始化缓存目录:

```
[root@localhost ~]# mkdir /data/cache
[root@localhost ~]# chown -R squid:squid /data/cache/
[root@localhost ~]# squid -z
2013/06/12 16:25:14| Creating Swap Directories
2013/06/12 16:25:14| /data/cache exists
```

好了，初始化完成后，就可以启动squid了:

```
[root@localhost ~]# /etc/init.d/squid start
正在启动 squid：.                                          [确定]
```

查看squid是否启动:

```
[root@localhost ~]# ps aux |grep squid
root      7201  0.0  0.7  14524  2444 ?        Ss   16:25   0:00 squid -f /etc/squid/squid.conf
squid     7204  0.0  2.7  17468  9024 ?        S    16:25   0:00 (squid) -f /etc/squid/squid.conf
squid     7205  0.0  0.2   3280   916 ?        S    16:25   0:00 (unlinkd)
```

现在您可以在您的真机上测测看squid的正向代理了，具体IE选项阿铭不再阐述，如果实在不懂请google或者求助阿铭(http://www.lishiming.net/forum-40-1.html), 而阿铭懒得去设置什么IE选项，直接使用curl命令测试即可:

```
[root@localhost ~]# curl -xlocalhost:3128  http://www.baidu.com/
```

如果您看到了一大串，说明squid正向代理设置ok啦。另外我们也可以观察squid对图片的缓存:

```
[root@localhost ~]# curl -xlocalhost:3128 http://www.lishiming.net/static/image/common/logo.png -I
HTTP/1.0 200 OK
Server: nginx/1.0.0
Date: Sat, 08 Jun 2013 04:30:17 GMT
Content-Type: image/png
Content-Length: 7785
Last-Modified: Wed, 13 Jan 2010 03:33:47 GMT
Accept-Ranges: bytes
X-Cache: HIT from dx_cache216.5d6d.com
X-Cache: MISS from localhost.localdomain
X-Cache-Lookup: MISS from localhost.localdomain:3128
Via: 1.0 dx_cache216.5d6d.com:80 (squid), 1.0 localhost.localdomain (squid/3.1.10)
Connection: keep-alive

[root@localhost ~]# curl -xlocalhost:3128 http://www.lishiming.net/static/image/common/logo.png -I
HTTP/1.0 200 OK
Server: nginx/1.0.0
Content-Type: image/png
Content-Length: 7785
Last-Modified: Wed, 13 Jan 2010 03:33:47 GMT
Accept-Ranges: bytes
Date: Sat, 08 Jun 2013 04:30:17 GMT
X-Cache: HIT from dx_cache216.5d6d.com
Age: 360898
Warning: 113 localhost.localdomain (squid/3.1.10) This cache hit is still fresh and more than 1 day old
X-Cache: HIT from localhost.localdomain
X-Cache-Lookup: HIT from localhost.localdomain:3128
Via: 1.0 dx_cache216.5d6d.com:80 (squid), 1.0 localhost.localdomain (squid/3.1.10)
Connection: keep-alive
```

阿铭连续访问了两次阿铭论坛的logo图片，可以发现前后两次的不同，其中 ``X-Cache-Lookup: HIT from localhost.localdomain:3128'' 显示，该请求已经HIT, 它直接从本地的3128端口获取了数据。

有时，我们会有这样的需求，就是想限制某些域名不能通过代理访问，或者说只想代理某几个域名，这如何做呢？在squid.conf中找到:

```
acl CONNECT method CONNECT
```

在其下面添加四行:

```
acl http proto HTTP
acl good_domain dstdomain .lishiming.net .aminglinux.com
http_access allow http good_domain
http_access deny http !good_domain
```

其中我的白名单域名为 ''.lishiming.net .aminglinux.com'' ，这里的 . 表示万能匹配，前面可以是任何字符，您只需要填写您的白名单域名即可。重启squid再来测测看:

```
[root@localhost ~]# /etc/init.d/squid restart
[root@localhost ~]# curl -xlocalhost:80 http://www.baidu.com/ -I
```

访问百度已经变为403了。如果要设置黑名单呢？道理是一样的:

```
acl http proto HTTP
acl bad_domain dstdomain .sina.com .souhu.com
http_access allow http !bad_domain
http_access deny http bad_domain
```

重启squid后，测试:

```
[root@localhost ~]# /etc/init.d/squid restart
[root@localhost ~]# curl -xlocalhost:80 http://www.sina.com/ -I
[root@localhost ~]# curl -xlocalhost:80 http://www.baidu.com/ -I
```

baidu.com可以访问，而sina.com不可以访问了。

23.3 搭建Squid反向代理

过程其实和前面的正向代理没有什么太大区别，唯一的区别是配置文件中一个地方需要改动一下。需要把:

```
http_port 3128
```

改为:

```
http_port 80  accel vhost vport
```

然后再增加您要代理的后端真实服务器信息:

```
cache_peer 123.125.119.147 parent 80 0 originserver name=a
cache_peer 61.135.169.125 parent 80 0 originserver name=b
cache_peer_domain a www.qq.com
cache_peer_domain b www.baidu.com
```

因为咱们之前没有配置网站信息，所以阿铭就拿qq.com和baidu.com来做个例子吧。其中cache_peer为配置后端的服务器ip以及端口，name后边为要配置的域名，这里和后面的cache_peer_domain相对应。实际的应用中，ip大多为内外ip，而域名也许会有多个，如果是squid要代理一台web上的所有域名，那么就写成这样:

```
cache_peer 192.168.10.111 80 0 originserver
```

后面连cache_peer_domain 也省了。

反向代理主要用于缓存静态项，因为诸多静态项目尤其是图片、流媒体等比较耗费带宽，在中国，联通网访问电信的资源本例就慢，如果再去访问大流量的图片、流媒体那更会慢了，所以如果在联通网配置一个squid反向代理，让联通客户端直接访问这个联通squid，而这些静态项已经被缓存在了squid上，这样就大大加快了访问速度。也许您听说过CDN, 其实它的设计原理就是这样的思路。好了，我们再测一测反向代理吧。

因为修改了配置文件，所以需要重启一下squid:

```
[root@localhost ~]# /etc/init.d/squid restart
[root@localhost ~]# curl -xlocalhost:80 http://www.baidu.com/
[root@localhost ~]# curl -xlocalhost:80 http://www.qq.com/
[root@localhost ~]# curl -xlocalhost:80 http://www.sina.com/
```

您会发现，baidu.com和qq.com都能正常访问，然而sina.com访问503了，这是因为阿铭并没有加sina.com的相关设置。

还有一个知识点，阿铭需要介绍给您:

```
[root@localhost ~]# squid -h
Usage: squid [-cdhvzCFNRVYX] [-s | -l facility] [-f config-file] [-[au] port] [-k signal]
    -a port   Specify HTTP port number (default: 3128).
    -d level  Write debugging to stderr also.
    -f file   Use given config-file instead of
              /etc/squid/squid.conf
```

```
    -h        Print help message.
    -k reconfigure|rotate|shutdown|interrupt|kill|debug|check|parse
              Parse configuration file, then send signal to
              running copy (except -k parse) and exit.
    -s | -l facility
              Enable logging to syslog.
    -u port   Specify ICP port number (default: 3130), disable with 0.
    -v        Print version.
    -z        Create swap directories
    -C        Do not catch fatal signals.
    -D        OBSOLETE. Scheduled for removal.
    -F        Don't serve any requests until store is rebuilt.
    -N        No daemon mode.
    -R        Do not set REUSEADDR on port.
    -S        Double-check swap during rebuild.
    -X        Force full debugging.
    -Y        Only return UDP_HIT or UDP_MISS_NOFETCH during fast reload.
```

上面阿铭把squid命令所用到的选项全部打印出来了，但阿铭觉得最常用的除了 `squid -k check` 外，还有一个那就是 `squid -k reconfigure` 它们俩都可以简写:

```
[root@localhost ~]# squid -kche
[root@localhost ~]# squid -krec
```

其中第二条命令表示重新加载配置文件，如果我们更改了配置文件后，不需要重启squid服务，直接使用该命令重新加载配置即可。

阿铭建议您最好再扩展学习一下: http://www.lishiming.net/thread-5446-1-1.html

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

TWENTYTHREE

第二十三章 配置TOMCAT

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

目前有很多网站使用jsp的程序编写，所以解析jsp的程序就必须要有相关的软件来完成。Tomcat就是用来解析jsp程序的一个软件，Tomcat是Apache 软件基金会（Apache Software Foundation）的Jakarta项目中的一个核心项目，由Apache、Sun和其他一些公司及个人共同开发而成。因为Tomcat技术先进、性能稳定，而且免费，因而深受Java爱好者的喜爱并得到了部分软件开发商的认可，成为目前比较流行的Web 应用服务器。

Tomcat是一个轻量级应用服务器，在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试JSP程序的首选。对于一个初学者来说，可以这样认为，当在一台机器上配置好Apache服务器，可利用它响应对HTML 页面的访问请求。实际上Tomcat 部分是Apache服务器的扩展，但它是独立运行的，所以当你运行tomcat时，它实际上作为一个与Apache 独立的进程单独运行的。

24.1 安装tomcat

Tomcat的安装分为两个步骤：安装JDK和安装Tomcat.

JDK(Java Development Kit)是Sun Microsystems针对Java开发员的产品。自从Java推出以来，JDK已经成为使用最广泛的Java SDK. JDK是整个Java的核心，包括了Java运行环境，Java工具和Java基础的类库。所以要想运行jsp的程序必须要有JDK的支持，理所当然安装Tomcat的前提是安装好JDK.

安装JDK

下载jdk-6u23-linux-i586.bin

```
cd /usr/local/src/
wget http://www.lishiming.net/data/attachment/forum/jdk-6u23-linux-i586.bin
```

您也可以从官方网站(http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html)下载其他版本。

```
chmod a+x jdk-6u23-linux-i586.bin
./jdk-6u23-linux-i586.bin
```

它会自动把文件解压出来，到最后会提示 ``Press Enter to continue.....'' ， 只需要按一下回车就可以了。

```
mv  jdk1.6.0_23  /usr/local/
```

设置环境变量

```
vim /etc/profile
```

在末尾输入以下内容:

```
JAVA_HOME=/usr/local/jdk1.6.0_23/
JAVA_BIN=/usr/local/jdk1.6.0_23/bin
JRE_HOME=/usr/local/jdk1.6.0_23/jre
PATH=$PATH:/usr/local/jdk1.6.0_23/bin:/usr/local/jdk1.6.0_23/jre/bin
CLASSPATH=/usr/local/jdk1.6.0_23/jre/lib:/usr/local/jdk1.6.0_23/lib:/usr/local/jdk1.6.0_23/jre/lib/charsets.jar
export  JAVA_HOME  JAVA_BIN JRE_HOME  PATH  CLASSPATH
```

保存文件后，使其生效:

```
source /etc/profile
```

检测是否设置正确:

```
java -version
```

如果显示如下内容，则配置正确:

```
java version "1.6.0_23"
Java(TM) SE Runtime Environment (build 1.6.0_23-b05)
Java HotSpot(TM) Client VM (build 19.0-b09, mixed mode, sharing)
```

安装Tomcat

上面介绍了那么多内容，仅仅是在为安装tomcat做准备工作而已，现在才是安装tomcat.

```
cd /usr/local/src/
wget http://www.lishiming.net/data/attachment/forum/apache-tomcat-7.0.14.tar.gz
```

如果觉得这个版本不适合，可以到官方网站(http://tomcat.apache.org/)下载。

```
tar zxvf apache-tomcat-7.0.14.tar.gz
mv apache-tomcat-7.0.14 /usr/local/tomcat
cp -p /usr/local/tomcat/bin/catalina.sh /etc/init.d/tomcat
vim /etc/init.d/tomcat
```

在第二行加入以下内容:

```
# chkconfig: 112 63 37
# description: tomcat server init script
# Source Function Library
. /etc/init.d/functions

JAVA_HOME=/usr/local/jdk1.6.0_23/
CATALINA_HOME=/usr/local/tomcat
```

保存文件后，执行以下操作:

```
chmod 755 /etc/init.d/tomcat
chkconfig --add tomcat
chkconfig tomcat on
```

启动tomcat:

```
service tomcat start
```

查看是否启动成功:

```
ps aux |grep tomcat
```

如果有进程的话，请在浏览器中输入http://IP:8080/ 你会看到tomcat的主界面。

24.2 配置tomcat

1. 配置tomcat服务的访问端口

tomcat默认启动的是8080，如果你想修改为80，则需要修改server.xml文件:

```
vim /usr/local/tomcat/conf/server.xml
```

找到:

```
<Connector port="8080" protocol="HTTP/1.1"
```

修改为:

```
<Connector port="80" protocol="HTTP/1.1"
```

2. 配置新的虚拟主机

```
cd /usr/local/tomcat/conf/
vim server.xml
```

找到</Host>下一行插入新的<Host>内容如下:

```
<Host name="www.123.cn" appBase="/data/tomcatweb"
    unpackWARs="false" autoDeploy="true"
    xmlValidation="false" xmlNamespaceAware="false">
    <Context path="" docBase="./" debug="0" reloadable="true" crossContext="true"/>
</Host>
```

保存后，重启tomcat:

```
service tomcat stop
service tomcat start
```

24.3 测试tomcat

先创建tomcat的测试文件:

```
vim /data/tomcatweb/111.jsp
```

加入如下内容:

```
<html><body><center>
    Now time is: <%=new java.util.Date()%>
</center></body></html>
```

保存后，使用curl测试:

```
[root@localhost ~]# curl -xlocalhost:80 www.123.cn/111.jsp
```

看看运行结果是否是:

```
<html><body><center>
    Now time is: Thu Jun 13 15:26:03 CST 2013
</center></body></html>
```

如果是这样的结果，说明tomcat搭建成功。另外，您也可以在您的真机上，绑定hosts，用IE来测试它。

阿铭建议您最好再扩展学习一下：http://www.lishiming.net/thread-5447-1-1.html

教程答疑：请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

TWENTYFOUR

第二十四章 配置SAMBA服务器

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

以前我们在windows上共享文件的话，只需右击要共享的文件夹然后选择共享相关的选项设置即可。然而如何实现windows和linux的文件共享呢？这就涉及到了samba服务了，这个软件配置起来也不难，使用也非常简单。

25.1 samba配置文件smb.conf

安装系统的时候大多会默认安装samba，如果没有安装，在CentOS上只需要运行这个命令安装即可:

```
yum install -y samba samba-client
```

Samba的配置文件为/etc/samba/smb.conf，通过修改这个配置文件来完成我们的各种需求。打开这个配置文件，你会发现很多内容都用 # 或者 ; 注视掉了。先看一下未被注释掉的部分:

```
[global]
        workgroup = MYGROUP
        server string = Samba Server Version %v
        security = user
        passdb backend = tdbsam
        load printers = yes
        cups options = raw
[homes]
        comment = Home Directories
        browseable = no
        writable = yes
[printers]
        comment = All Printers
        path = /var/spool/samba
        browseable = no
        guest ok = no
        writable = no
        printable = yes
```

主要有以上三个部分：[global], [homes], [printers]

[global] 定义全局的配置，workgroup用来定义工作组，相信如果您安装过windows的系统，你会对这个workgroup不陌生。一般情况下，需要我们把这里的MYGROUP改成WORKGROUP（windows默认的工作组名字）。

security = user #这里指定samba的安全等级。关于安全等级有四种：

share：用户不需要账户及密码即可登录samba服务器

user：由提供服务的samba服务器负责检查账户及密码（默认）

server：检查账户及密码的工作由另一台windows或samba服务器负责

domain：指定windows域控制服务器来验证用户的账户及密码。

passdb backend = tdbsam # passdb backend（用户后台），samba有三种用户后台：smbpasswd, tdbsam和ldapsam.

smbpasswd：该方式是使用smb工具smbpasswd给系统用户（真实用户或者虚拟用户）设置一个Samba密码，客户端就用此密码访问Samba资源。smbpasswd在/etc/samba中，有时需要手工创建该文件。

tdbsam：使用数据库文件创建用户数据库。数据库文件叫passdb.tdb，在/etc/samba中。passdb.tdb用户数据库可使用 smbpasswd -a 创建Samba用户，要创建的Samba用户必须先是系统用户。也可使用pdbedit创建Samba账户。pdbedit参数很多，列出几个主要的：

pdbedit -a username：新建Samba账户。

pdbedit -x username：删除Samba账户。

pdbedit -L：列出Samba用户列表，读取passdb.tdb数据库文件。

pdbedit -Lv：列出Samba用户列表详细信息。

pdbedit -c ``[D]'' -u username：暂停该Samba用户账号。

pdbedit -c ``[]'' -u username：恢复该Samba用户账号。

ldapsam：基于LDAP账户管理方式验证用户。首先要建立LDAP服务，设置 ``passdb backend = ldapsam:ldap://LDAP Server''

load printers 和 cups options 两个参数用来设置打印机相关。

除了这些参数外，还有几个参数需要你了解：

netbios name = MYSERVER # 设置出现在网上邻居中的主机名

hosts allow = 127. 192.168.12. 192.168.13. # 用来设置允许的主机，如果在前面加 '';'' 则表示允许所有主机

log file = /var/log/samba/%m.log #定义samba的日志，这里的%m是上面的netbios name

max log size = 50 # 指定日志的最大容量，单位是K

[homes] 该部分内容共享用户自己的家目录，也就是说，当用户登录到samba服务器上时实际上是进入到了该用户的家目录，用户登陆后，共享名不是homes而是用户自己的标识符，对于单纯的文件共享的环境来说，这部分可以注视掉。

[printers] 该部分内容设置打印机共享。

25.2 samba实践

注意：在试验之前，请先检测selinux是否关闭，否则可能会试验不成功。关于如何关闭selinux请查看第十六章linux系统日常管理的linux的防火墙部分(http://study.lishiming.net/chapter16.html#id3)

1. 共享一个目录，任何人都可以访问，即不用输入密码即可访问，要求只读

打开samba的配置文件/etc/samba/smb.conf 在[global]部分

把:

```
MYGROUP
```

改成:

```
WORKGROUP
```

把:

```
security = user
```

修改为:

```
security = share
```

然后在文件的最末尾处加入以下内容:

```
[share]
        comment = share all
        path = /tmp/samba
        browseable = yes
        public = yes
        writable = no
```

创建测试目录:

```
mkdir /tmp/samba
chmod 777 /tmp/samba
touch /tmp/samba/sharefiles
echo "111111" > /tmp/samba/sharefiles
```

启动samba服务:

```
/etc/init.d/smb start
```

测试：

首先测试你配置的smb.conf是否正确，用下面的命令:

```
testparm
```

您应该会看到一个警告:WARNING: The security=share option is deprecated, 不过影响不大，无需管它。如果没有错误，则在你的windows机器上的浏览器中输入:

```
file://IP/share
```

看是否能访问到sharefiles

2. 共享一个目录，使用用户名和密码登录后才可以访问，要求可以读写

打开samba的配置文件/etc/samba/smb.conf

[global] 部分内容如下:

```
[global]
        workgroup = WORKGROUP
        server string = Samba Server Version %v
        security = user
        passdb backend = tdbsam
        load printers = yes
        cups options = raw
```

还需要加入以下内容:

```
[myshare]
        comment = share for users
        path = /samba
        browseable = yes
        writable = yes
        public = no
```

保存配置文件，创建目录:

```
mkdir /samba
chmod 777 /samba
```

然后添加用户。因为在[globa]中 ``passdb backend = tdbsam''，所以要使用 pdbedit 来增加用户，注意添加的用户必须在系统中存在，所以需要先创建系统账号:

```
useradd user1
useradd user2
```

然后添加user1为samba账号:

```
pdbedit -a user1
```

再添加user2为samba账号:

```
pdbedit -a user2
```

我们可以列出samba所有账号:

```
pdbedit -L
```

重启samba服务:

```
service smb restart
```

测试：

打开IE浏览器输入:

```
file://IP/myshare/
```

然后输入用户名和密码

3. 使用linux访问samba服务器

Samba服务在linux下同样可以访问。前提是您的linux安装了samba-client软件包。安装完后就可以使用smbclient命令了。具体语法为:

```
smbclient //IP/共享名  -U 用户名
```

如:

```
[root@localhost]# smbclient //10.0.4.67/myshare/ -U user1
Password:
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.6.9-151.el6]
smb: \>
```

出现如上所示的界面。可以打一个 ''?'' 列出所有可以使用的命令。常用的有cd, ls, rm, pwd, tar, mkdir, chown, get, put等等，使用 help + 命令可以打印该命令如何使用，其中get是下载，put是上传。

另外的方式就是通过mount挂载了，如:

```
mount -t cifs //10.0.4.67/myshare /mnt -o username=user1,password=123456
```

格式就是这样，要指定 -t cifs //IP/共享名 本地挂载点 -o后面跟username 和 password 挂载完后就可以像使用本地的目录一样使用共享的目录了，注意共享名后面不能有斜杠。

阿铭建议您最好再扩展学习一下: http://www.lishiming.net/thread-5448-1-1.html

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

TWENTYFIVE

第二十五章 MYSQL REPLICATION(主从)配置

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

MySQL Replication 又叫做AB复制或者主从复制。它主要用于MySQL的时时备份或者读写分离。在配置之前先做一下准备工作，配置两台mysql服务器，或者在一台服务器上配置两个端口也可以。阿铭本章的实验中就是在一台服务器上跑了两个mysql。

26.1 配置mysql服务

详细步骤，请参考(http://study.lishiming.net/chapter17.html#mysql), 阿铭只把简单步骤写一下。

根据阿铭提供的地址，假如您已经搭建好了一个mysql，跑的是3306端口，下面阿铭再搭建一个3307端口的mysql:

```
[root@localhost ~]# cd /usr/local/
[root@localhost local]# cp -r mysql mysql_2
[root@localhost local]# cd mysql_2
[root@localhost mysql_2]# ./scripts/mysql_install_db --user=mysql --datadir=/data/mysql2
```

最后一步是初始化数据库目录，如果出现两个 ``OK'' 并且生成/data/mysql2目录才正确，否则请仔细查看错误信息，如果不能解决请到阿铭论坛(http://www.lishiming.net/forum-40-1.html)发帖咨询阿铭。拷贝配置文件到mysql_2下，并修改相关项目:

```
[root@localhost mysql_2]# cp /etc/my.cnf  ./my.cnf
[root@localhost mysql_2]# vim my.cnf
```

其中:

```
port          = 3306
```

改为:

```
port          = 3307
```

把:

```
socket        = /tmp/mysql.sock
```

改为:

```
socket        = /tmp/mysql2.sock
```

在这一行的下面再加一行:

```
datadir         = /data/mysql2
```

保存后就可以启动它了：

```
[root@localhost mysql_2]# cd bin/
[root@localhost bin]# ./mysqld_safe --defaults-file=../my.cnf --user=mysql &
```

如果以后想开机启动，就把它加入/etc/rc.d/rc.local文件中：

```
echo "./mysqld_safe --defaults-file=../my.cnf --user=mysql &" >>/etc/rc.d/rc.local
```

到此，目前阿铭已经在一个Linux上启动了两个mysql：

```
[root@localhost ~]# netstat -lnp |grep mysqld
tcp        0      0 0.0.0.0:3306                0.0.0.0:*    LISTEN      3169/mysqld
tcp        0      0 0.0.0.0:3307                0.0.0.0:*    LISTEN      3037/mysqld
unix  2      [ ACC ]     STREAM     LISTENING     29027  3037/mysqld    /tmp/mysql2.sock
unix  2      [ ACC ]     STREAM     LISTENING     29155  3169/mysqld    /tmp/mysql.sock
```

26.2 配置replication

阿铭打算把3307端口的mysql作为主(master)，而把3306的mysql作为从(slave). 为了让实验更加像生产环境，所以阿铭先在master上创建一个库db1，并且把mysql的库数据复制给它：

```
[root@localhost bin]# mysql -uroot -S /tmp/mysql2.sock

mysql> create database db1;
Query OK, 1 row affected (0.01 sec)

mysql> quit
Bye
```

也许您对阿铭使用的这个命令有点疑问，-S 后面指定mysql的socket文件路径，这也是登陆mysql的一种方法，因为在一台服务器上跑了两个mysql端口，所以，只能用 -S 这样的方法来区分。首先创建了db1库，然后把mysql库的数据复制给它：

```
mysqldump -uroot -S /tmp/mysql2.sock mysql > 123.sql
mysql -uroot -S /tmp/mysql2.sock db1 < 123.sql
```

1. 设置master

修改配置文件：

```
vim /usr/local/mysql_2/my.cnf
```

在[mysqld]部分查看是否有以下内容，如果没有则添加：

```
server-id=1
log-bin=mysql-bin
```

除了这两行是必须的外，还有两个参数，您可以选择性的使用：

```
binlog-do-db=databasename1,databasename2
binlog-ignore-db=databasename1,databasename2
```

binlog-do-db=需要复制的数据库名，多个数据库名，使用逗号分隔。binlog-ignore-db=不需要复制的数据库库名，多个数据库名，使用逗号分隔。这两个参数其实用一个就可以啦。

如果修改过配置文件需要重启mysqld服务，否则不需要重启:

```
[root@localhost ~]# pid=`ps uax |grep mysql2.sock |grep -v grep |awk '{print $2}'`
[root@localhost ~]# kill -0 $pid; sleep 3; kill $pid
[root@localhost ~]# cd /usr/local/mysql_2/bin/
[root@localhost bin]# ./mysqld_safe --defaults-file=../my.cnf --user=mysql &
```

由于阿铭没有编写启动脚本，所以3307端口的mysql重启有点麻烦。

设置mysql数据库的root访问密码:

```
mysqladmin -u root -S /tmp/mysql2.sock password '123456'
mysql -u root -S /tmp/mysql2.sock -p'123456'

mysql> grant replication slave on *.* to 'repl'@'127.0.0.1' identified by '123123';
//这里的repl是为slave端设置的访问master端mysql数据的用户，密码为123123，这里的127.0.0.1为slave的ip(因为阿铭配置的master和
mysql> flush tables with read lock;  //锁定数据库，此时不允许更改任何数据
mysql> show master status;  //查看状态，这些数据是要记录的，一会要在slave端用到
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000006 |   474952 |              |                  |
+------------------+----------+--------------+------------------+
```

2. 设置slave

先修改slave的配置文件my.cnf:

```
vim /etc/my.cnf
```

找到 ``server-id = 1'' 这一行，删除掉或者改为 ``server-id = 2'' 总之不能让这个id和master一样，否则会报错。另外在从上，您也可以选择性的增加如下两行，对应于主上增加的两行:

```
replicate-do-db=databasename1,databasename2
replicate-ignore-db=databasename1,databasename2
```

改完后，重启slave:

```
service mysqld restart
```

拷贝master上的db1库的数据到slave上，因为master和slave都在一台服务器上，所以操作起来简单了很多，如果是不同的机器，可能就需要远程拷贝了，希望您注意这一点:

```
[root@localhost ~]# mysqldump -uroot -S /tmp/mysql2.sock -p123456 db1 > db1.sql
[root@localhost ~]# mysql -uroot -S /tmp/mysql.sock -pyourpassword -e "create database db1"
[root@localhost ~]# mysql -uroot -S /tmp/mysql.sock -pyourpassword db1 < db1.sql
```

第二行中，阿铭使用了一个-e选项，这个选项阿铭在之前的章节中并没有介绍，它用来把mysql的命令写到shell中，这样可以方便把mysql操作写进脚本中，它的格式就是 -e "commond" 它很实用，阿铭用的也是蛮多的，所以请您熟记它。把数据拷贝过来后，就需要在slave上配置主从了:

```
[root@localhost ~]# mysql -uroot -S /tmp/mysql.sock -pyourpassword
mysql> slave stop;
mysql> change master to master_host='127.0.0.1', master_port=3307,
master_user='repl', master_password='123123',
master_log_file='mysql-bin.000006', master_log_pos=474952;
mysql> slave start;
```

相信聪明的您一定可以看懂上面的各个参数分别表示什么含义，其中master_log_file和master_log_pos是在上面使用 show master status 查到的数据。执行完这一步后，需要在master上执行一步:

```
mysql -uroot -S /tmp/mysql2.sock -p123456 -e "unlock tables"
```

然后查看slave的状态:

```
mysql> show slave status\G;
```

确认以下两项参数都为yes:

```
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
```

26.3 测试主从

在master上执行如下命令:

```
[root@localhost ~]# mysql -uroot -S /tmp/mysql2.sock -p123456 -e "use db1;
select count(*) from db"
+----------+
| count(*) |
+----------+
|        2 |
+----------+
[root@localhost ~]# mysql -uroot -S /tmp/mysql2.sock -p123456 -e "use db1;
truncate table db"
[root@localhost ~]# mysql -uroot -S /tmp/mysql2.sock -p123456 -e "use db1;
select count(*) from db"
+----------+
| count(*) |
+----------+
|        0 |
+----------+
```

这样清空了db1.db表的数据，下面查看slave上的该表数据:

```
[root@localhost ~]# mysql -uroot -S /tmp/mysql.sock -pyourpassword -e "use db1; select count(*) from db"
+----------+
| count(*) |
+----------+
|        0 |
+----------+
```

slave上的该表也被清空了。这样好像不太明显，不妨继续把db表删除试试:

```
[root@localhost ~]# mysql -uroot -S /tmp/mysql2.sock -p123456 -e "use db1; drop table db"
[root@localhost ~]# mysql -uroot -S /tmp/mysql.sock -pyourpassword -e "use db1; select count(*) from db"
ERROR 1146 (42S02) at line 1: Table 'db1.db' doesn't exist
```

这次很明显了。

主从配置起来很简单，但是这种机制也是非常脆弱的，一旦我们不小心在从上写了数据，那么主从也就被破坏了。另外如果重启master，务必要先把slave停掉，也就是说需要在slave上去执行 `slave stop` 命令，然后再去重启master的mysql服务，否则很有可能就会中断了。当然重启完后，还需要把slave给开启 `slave start`.

阿铭建议您最好再扩展学习一下: http://www.lishiming.net/thread-5449-1-1.html

教程答疑: 请移步这里.

欢迎您加入 阿铭学院 和阿铭一起学习Linux，让阿铭成为您Linux生涯中永远的朋友吧！

CHAPTER

TWENTYSIX

结语

跟阿铭学Linux邀请函 (http://www.aminglinux.com)，二期已开班，感兴趣的联系QQ:306798658，现在报名还有优惠！同样欢迎您 购买视频资料。

怎么样?学的还尽兴吧!首先感谢您对阿铭的信任，从第一章前言读到了结尾。阿铭的教程还并未结束哦，还在不断的优化、改进，更新中！阿铭衷心地希望您可以继续关注阿铭，关注阿铭的教程，更希望您多到 阿铭论坛 多学习多讨论！

到此，阿铭希望您已经学到了诸多实用的Linux技术，阿铭相信只要您坚持看完了每一章节的内容，并且做了充分的练习，那您一定是一名合格的Linux系统工程师了。但是，这还远远不够，您还需要尽快成长，进一步提高您的Linux技能水平。阿铭的建议是，您最好一边工作一边从工作中总结、学习和进步。这样做的优点是，学习稳扎稳打、现学现卖、知识掌握牢固，印象深刻！如果您有这样的工作机会，阿铭是为您高兴的，如果没有这样的机会，您也不用担心，阿铭会帮助您。你有Linux方面的问题都可以咨询我QQ:306798658，也可以在论坛发帖提问。

阿铭不相信捷径，但是阿铭在这里给您提供一条相对来说为捷径的学习道路，那就是要您站到阿铭的肩膀上，去眺望更远的地方。您在前面的章节中每章都会看到一个''扩展学习''的链接，没错，阿铭为了提高大家学习的积极性，特意设置了门槛，阿铭本意是好的，是想督促您、刺激您坚持学习，因为只有让您付出更多，您才会更加珍惜，才会收获更多，所以，请您跟随阿铭的脚步，来学习阿铭多年来积累的经验吧。

其实阿铭的公开的教程中介绍的相对较初级，要想成为中高级系统工程师，这点知识是远远不够的，所以您需要在阿铭论坛的vip版块中学习更多内容，在这里阿铭还设有几章高级的课程，阿铭真心希望您能够跟随阿铭一起走向更高级别！ vip版块 区的课程还包括：

nagions 监控

zabbix监控

DNS服务搭建

邮件服务搭建

HA高可用集群

LB负载均衡集群

自动化工具puppet

Linux下的虚拟化

等等……

最后，衷心祝愿，所有阅读过阿铭教程的朋友，能够在Linux这条路上走的更高、更远！其实成功离您就只有一步之遥！坚持是最重要的！！