Linux 简要

Geffzhang（张善友）

目录

作者介绍

张善友 2001 年开始他的职业生涯，他一直是一个微软技术的开发者，连续荣获 10 年的 ASP.NET MVP，热衷于开源，在社区积极推广开源技术 Mono。
张善友拥有 SUSE Linux 企业服务器，CentOS 以及 tLinux（腾讯自行研制的 Linux 发行版）的专业经验，他主要是在 CentOS 上部署 Mono 平台，在业余时间喜欢教别人如何使用和利用 Linux 操作系统的力量，特别针对 Windows 开发人员收集编写了这本 Linux 简要。希望对 Windows 上的.NET 开发人员顺利跨入 Linux 的 Mono 平台开发提供帮助。
业余时间运营微信公众号 dotNET 跨平台，微信号 opendotnet，欢迎关注。

第一章 Linux 和 Mono 的历史

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。其实我就不怎么了解 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 Google 一下吧，非常多，可谓汗牛充栋足够让您看几天的。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 Linux 的历史的，也算是我对 Linux 的创始人 Linus Torvalds 先生的尊重。

Linux 简要介绍

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。1991 年 linus 购买了自己的第一台 PC，并且决定开始开发自己的操作系统。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。他很快编写了自己的磁盘驱动程序和文件系统，并且慷慨地把源代码上传到互联网，Linus 把这个操作系统命名为 Linux，意指“Linus 的 Minix” (Linus’ Minix)。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。Linus 根本不会想到，这个内核迅速引起了全世界的兴趣，在短短的几年时间里，借助社区开发的推动力，Linux 迸发出强大的生命力。1994 年，Linux 内核 1.0 版本正式发布。

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 目前得到了大部分 IT 巨头的支持，并且进入了重要战略规划的核心领域。一个非盈利性的操作系统计划能够延续那么多年，并且最终成长为在各行各业发挥巨大影响力的产品。

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。有些发行版（发布） 保持由社区的志愿者，有些是有收费订阅和支持的公司。有些发行版被为了在笔记本电脑和台式机运行，而有些版本被设计为在服务器上运行。较知名的发行版有：

- ✧ Linux Mint
- ✧ Ubuntu
- ✧ Debian
- ✧ Fedora
- ✧ openSUSE
- ✧ Arch Linux
- ✧ CentOS
- ✧ Red Hat Enterprise Linux

Mono 平台常用的就是 CentOS ，Ubuntu 以及 SuSE，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。大部分互联网公司（Google、Facebook、腾讯、阿里、百度）的服务器全部大部分都是基于 CentOS 自定义系统，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。

本章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。

Mono 历史

Mono（就是西班牙语中的 monkey）是一个在非 Windows 操作系统中提供 C# 编译器和 CLR 的开源项目。目前，Mono 授权于 GPL 版本 2、LGPL 版本 2、MIT 以及双许可证，可以运行于 Mac，Linux、BSD 以及其他操作系统中。通过 C#编译器，还可以在 Mono 中运行其他语言，其中包括 F#，Java、Scala 和 Visual Basic 等等。

Mono 的创始人是 Miguel de Icaza, 微软在 2001 年把 CLI 和 C# 提交给了 ECMA[ECMA 是一个致力于推动行业范围内采用信息和通信技术的非特定供应商的国际标准组织]标准化 ECMA 335 和 ECMA 334)，比 Java 还早的标准化了.NET 平台。Miguel de Icaza 看到了 C#语言的优雅和高效率，Ximian 内部对如何创建能有效提升生产效率的工具进行了大量的讨论，他们的目标是通过这些创建出来的工具让用户可以在更短时间内创建出更多的应用程序从而缩短开发周期和降低开发成本。de Icaza 所在的 Ximian 公司在 2001 年 7 月开始启动一个名叫 Mono Project 的开放源码版本".NET"的开发项目，旨在使开发者能够编写同时在 Windows 和 Linux 上运行的.NET 程序。并在 2004 年发布了第一个版本，Mono 目前的最新版本是 4.0.1，同时 Mono 还在不断地持续更新。Mono 一直是由 de Icaza 直接领导，2012 年 Novell 公司被收购，Mono 项目的管理已经移交给 de Icaza 所创立的一家新公司 Xamarin，由其指引 Mono 的发展方向。现在 Xamarin 的职责是发展 Mono，同时负责开发 Xamarin.IOS 和 Xamarin.Android 以及让开发人员使用这些产品所需的软件。Xamarin 所领导的 Mono 现在已经覆盖到服务器，桌面和移动领域，我认为这些产品将是非常优秀的。

虽然期望 Mono 的功能可以尽可能多与.NET Framework 的功能相匹配，随着微软的开源政策的挑整，加强互操作性，成立了.NET 开源基金会。微软已经将整个.NET 开源了，Mono 项目拥有了许多与.NET Framework 功能相同的功能。

随着 Mono 一起的是名为 MonoDevelop 的开源 IDE,现已更名为 Xamarin Studio，该 IDE 作为 SharpDevelop IDE 的一个移植版本一起启动。MonoDevelop 最初只是一个运行在 Linux 上进行 Mono 开发的项目，随着 MonoDevelop 2.2 版本的发布，它也具备了在 Mac，Windows 以及其他非 Linux Unix 平台上用 Mono 进行开发的能力。

对 Linux 管理员的建议

Linux 系统和 windows 系统有太多不一样的地方，我相信大部分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，特别是.NET 程序员朋友，大家用的最多的 Windows，可能还没有碰过 Linux 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。下面针对 Linux 提几条建议：

1、 要安装什么版本的 Linux 操作系统？

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，本书介绍的知识点大多都是通用的。但是，我还是建议您安装 CentOS，因为这个版本的 Linux 在中国来说各大互联网公司用在服务器上的最多，而且它和 Redhat 是一样的，资料比较多。另外，最主要的原因是因为我在后续章节中所做的所有实验都是在 CentOS 上来实现的。

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。我早期学习 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 Mac 流畅，所以就不再使用图形界面了。随着使用 Linux 越来越多，在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，如果远程连接图形界面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，我建议读者朋友，从一开始学习 Linux 起就应该使用命令窗口。

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，一定要记得备份数据，否则，一旦出错您会后悔死。

- 尽量少使用 root

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，我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，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，而去还存到 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。我给您的建议是，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，也可以借助密码记录工具 keepass，后面会介绍。

第二章认识 Linux 系统

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，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)以及密码后进去，发现更不知所措了。没有关系，随着后面的逐渐深入的讲解，您会学到越来越多的知识。

Linux 是如何启动的

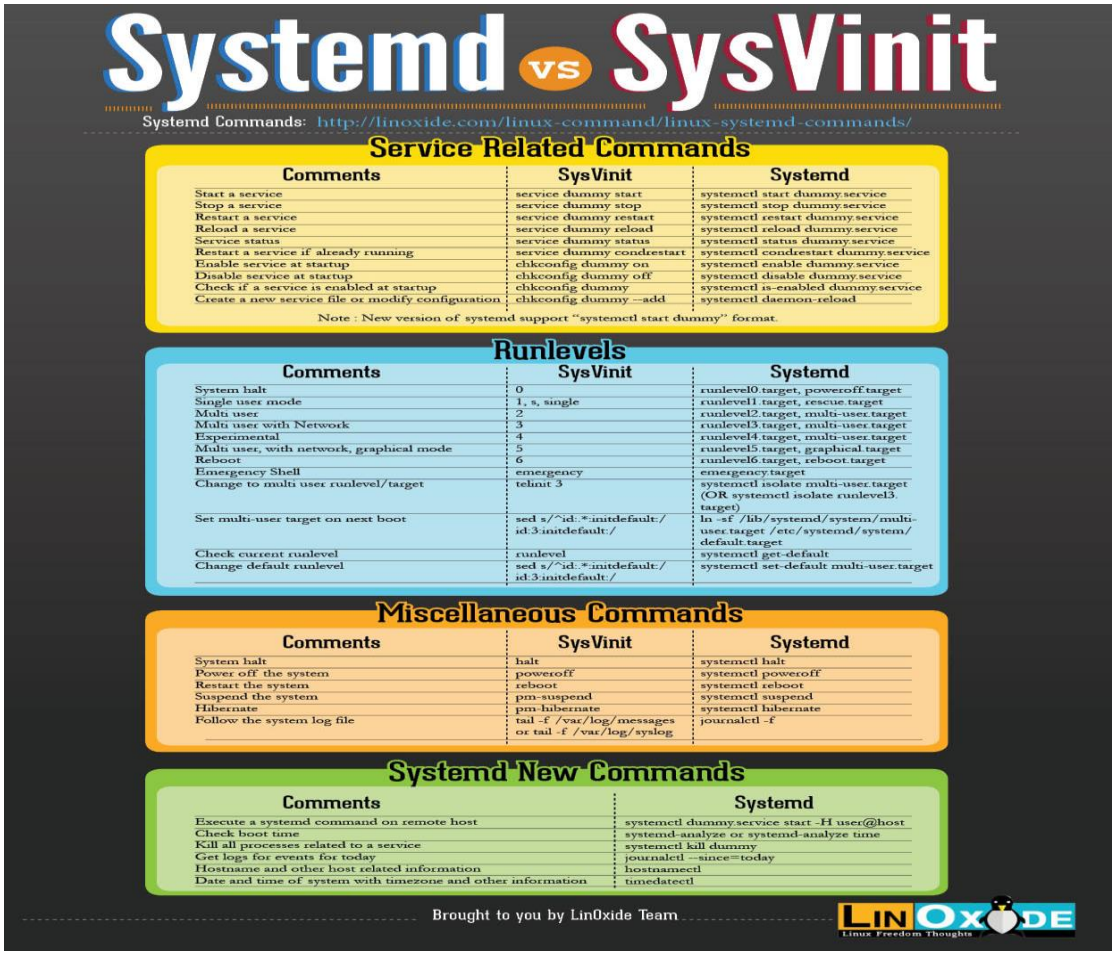
Linux 的启动其实和 windows 的启动过程很类似，不过 windows 我们是无法看到启动信息的，而 Linux 启动时我们会看到许多启动信息，例如某个服务是否启动。

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。

Linux 系统的 init 进程经历了两次重大的演进，传统的 sysvinit 已经逐渐淡出历史舞台，新的 UpStart 和 systemd 各有特点，越来越多的 Linux 发行版采纳了 systemd。Systemd 的目的是取代 Unix 时代以来一直在使用的 init 系统，以便于能够在进程启动的过程中更有效地引导加载服务。Systemd 之所以更快是因为它使用的脚本更少，并且尽量并行运行更多的任务。自 2010 年推出 Fedora 15 版本以来 Red Hat 就将 Systemd 作为默认功能。作为其操作系统计划的一部分，Red Hat 希望通过 Systemd 加强 CentOS 7 对 Docker 的支持方式。

CentOS 7 的第三个重大变化是使用 XFS 替代 ext4 作为默认的文件系统。虽然在 CentOS 6 中已经提供了 XFS 的选项，但是默认还是使用 ext4。XFS 支持高达 500TB 的容量，而 ext4 仅支持 50TB。不幸的是，除了备份和恢复之外目前还没有方法可以让用户从 ext4 或 btrfs 文件系统上迁移到 XFS。

具体参见文章 http://www.ibm.com/developerworks/cn/linux/1407_liuming_init3/index.html。



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.

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 程序首先是需要读取配置文件 /etc/inittab. inittab 是一个不可执行的文本文件，它有若干行指令所组成。关于各指令的具体说明这里不想阐述，因为对于初学者的您来说，可能搞不明白。不过没有关系，您也不需要去搞明白，以后 Linux 用的多了，自然会明白。

系统初始化

在 init 的配置文件中有这么一行：“si::sysinit:/etc/rc.d/rc.sysinit”它调用执行了“/etc/rc.d/rc.sysinit”文件，而 rc.sysinit 是一个 bash shell 的脚本，它主要是完成一些系统初始化的工作，rc.sysinit 是每一个运行级别都要首先运行的重要脚本。它主要完成的工作有：激活交换分区，检查磁盘，加载硬件模块以及其它一些需要优先执行任务。

当 rc.sysinit 程序执行完毕后，将返回 init 继续下一步。通常接下来会执行到“/etc/rc.d/rc”程序。以运行级别 3 为例，init 将执行配置文件 inittab 中的以下这行：
l5:5:wait:/etc/rc.d/rc 5 这一行表示以 5 为参数运行“/etc/rc.d/rc”,“/etc/rc.d/rc”是一个 Shell 脚本，它接受 5 作为参数，去执行“/etc/rc.d/rc5.d/” 目录下的所有的 rc 启动脚本，“/etc/rc.d/rc5.d/” 目录中的这些启动脚本实际上都是一些连接文件，而不是真正的 rc 启动脚本，真正的 rc 启动脚本实际上都是放在 “/etc/rc.d/init.d/” 目录下。而这些 rc 启动脚本有着类似的用法，它们一般能接受 start、stop、restart、status 等参数。“/etc/rc.d/rc5.d/”中的 rc 启动脚本通常是 K 或 S 开头的连接文件，对于以以 S 开头的启动脚本，将以 start 参数来运行。而如果发现存在相应的脚本也存在 K 打头的连接，而且已经处于运行态了(以“/var/lock/subsys/” 下的文件作为标志)，则将首先以 stop 为参数停止这些已经启动了的守护进程，然后再重新运行。这样做是为了保证是当 init 改变运行级别时，所有相关的守护进程都将重启。

建立终端

rc 执行完毕后，返回 init。这时基本系统环境已经设置好了，各种守护进程也已经启动了。init 接下来会打开 6 个终端，以便用户登录系统。在 inittab 中的以下 6 行就是定义了 6 个终端：

```
1:4355:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
```

从上面可以看出在 2、3、4、5 的运行级别中都将以 respawn 方式运行 mingetty 程序，mingetty 程序能打开终端、设置模式。同时它会显示一个文本登录界面，这个界面就是我们经常看到的登录界面，在这个登录界面中会提示用户输入用户名，而用户输入的用户将作为参数传给 login 程序来验证用户身份。

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。

图形界面与命令行界面切换

Linux 默认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。

学会使用快捷键

- Ctrl + C：这个是用来终止当前命令的快捷键，当然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。
- Tab： 这个键是最有用的键了，也是我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。
- Ctrl + D： 退出当前终端，同样您也可以输入 exit。
- 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。
- Ctrl + L： 清屏，使光标移动到第一行。

学会查询帮助文档 — man

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：

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

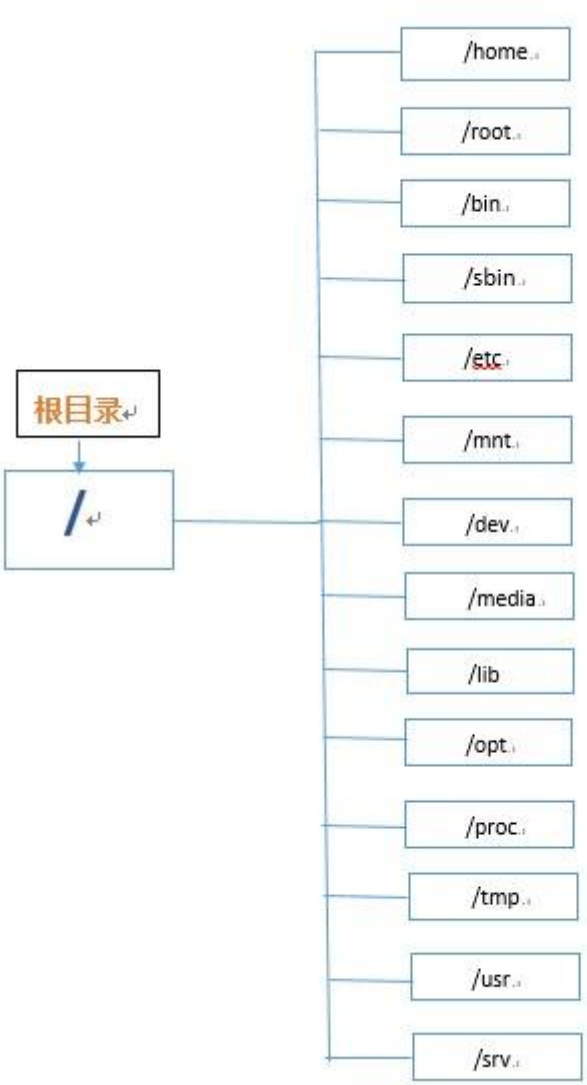
do not list implied . and ..

--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

Linux 系统目录结构

一般的 Linux 系统的文件结构如下图所示。



- / 系统根目录

  每个文件和目录都驻留在/目录下，很多的物理或者虚拟的存储设备也是连接在根目录 / 下。在 windows 系统中驱动器 C:\ 是和 Linux 根目录对应的，当另外一个存储设备连接到 Windows 系统时，它被指派新的驱动器号，例如 D:\。在 Linux 的系统中，存储设备是连接或者安装到/mnt 或/media/external 目录。

- /bin Binary

  bin 是 Binary 的缩写。这个目录存放着最经常使用的命令。其他非必要二进制文件都位于 /usr/bin。

- /etc 系统配置文件

  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 Home 目录

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。例如用户 mono 的主目录就是/home/mono，因为所有用户都有自己的主目录，他们都有让其数据保持私有，与其他系统或两者的组合上的用户共享选项。典型的主目录内容包括创建的用户、 文本文档、 度假照片、 音乐等文件。此外，在主目录中存储用户特定的配置。例如这些配置文件可以控制行为的用户的图形或文本的环境。

- /opt 可选的或第三方软件

  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。

- /tmp 临时空间

  这个目录是用来存放一些临时文件的。由系统或应用程序上的单个用户，可以使用此目录。通常在启动时，清理/tmp 目录下的空间，所以你不能在/tmp 存储需要长期保留的内容。

- /usr 用户相关的程序和只读数据驻留的位置

  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 的内容是由该系统而不是操作系统本身的实际用户使用。在 /usr 中存在整个目录层次结构。例如，/usr/bin 目录包含二进制文件和应用程序，而 /usr/share/doc 包含与这些应用程序相关的文档

- /var 可变数据

  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。

- /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。

- /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。

- /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。

- /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。

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: 这是系统默认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 的默认放置也是在这里。

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，我们先介绍一下这个 “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，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。
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。

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”我就讲这几个例子，当然它的可用选项还有很多哦，不过我只给介绍最常用的。因为我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，我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。

如何正确关机、重启

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）都是在在后台执行的，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、发送邮件， FTP 传送文件等等的，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文件中，所以有些服务的文件会有问题！）。

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，而要看后台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：sysnc -> 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.

第三章 Linux 系统的远程登陆

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 机房后者云计算中心，所以平时登录 Linux 系统都是通过远程登录的。Linux 系统中是通过 ssh 服务实现的远程登录功能。默认 ssh 服务开启了 22 端口，而且当我们安装完系统时，这个服务已经安装，并且是开机启动的。所以不需要我们额外配置什么就能直接远程登录 Linux 系统。ssh 服务的配置文件为 /etc/ssh/sshd_config，您可以修改这个配置文件来实现您想要的 ssh 服务。比如您可以更改启动端口为 115870。
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，而我喜欢用 Putty，只是因为它的小巧以及非常漂亮的颜色显示，而且是免费的。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

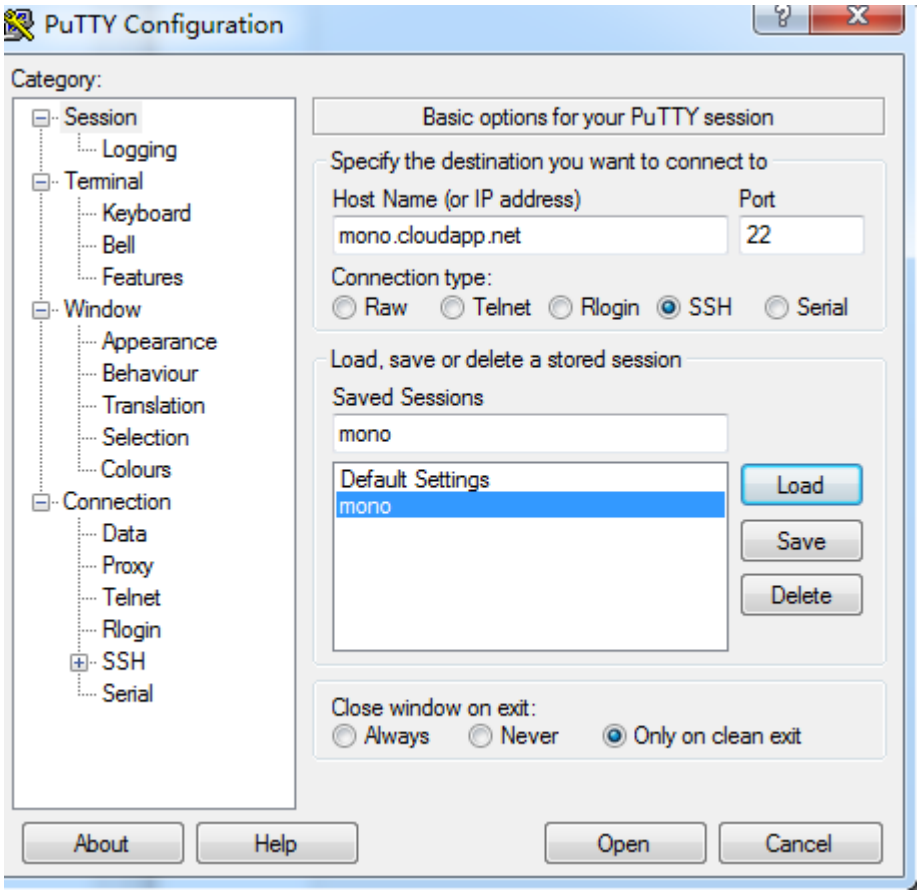
登录到 Linux 服务器上。
当开发机是 windows，服务器是 Linux 时，如何在 windows 操作系统和 linux 操作系统之间进行文件传输呢？ 使用 scp、wget 等命令拷贝文件，也有不怕麻烦的在服务器上安装 FTP 服务器，或者启用 samba 甚至 NFS。只是用 Linux 作为网页服务器，因为性能更好，更加安全稳定。这个时候要管理文件系统，想像 FTP 那么方便，又不想学习如何安装 FTP 服务器，winscp 就是一个极好的选择。只要你的 Linux 主机支持远程登录（ssh）,那么你下载一个 winscp，使用 ssh 账号登入，就可以像 FTP 那样简单管理文件系统了，不需要你在服务器端做任何操作和设置。
您不妨跟着我一起来用一用 Putty 配合 Winscp 远程管理 Linux 服务器。

下载 Putty 和 Winscp

建议您到 Putty 和 Winscp 的官方站点去下载英文版原版的 putty 和 Winscp。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，提醒各位，以后不管下载什么软件尽量去官方站点下载。
Putty 的官方下载地址：http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
Winscp 的官方下载地址：http://winscp.net/eng/download.php

用 putty 登陆您的 Linux

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，而去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，我的 Windows Azure 的 Linux 主机名为 “mono.cloudapp.net”.
- 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。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: azureuser
azureuser@mono.cloudapp.net's password:
Last login: Fri Aug  2 14:24:15 2013 from 183.17.166.96
[azureuser@mono ~]$
```

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。

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？

- 下载生成密钥工具

在本章前面提供的 putty 下载地址里，您一定看到了很多可以下载的东西，我们只让您下载了一个 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。

- 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。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，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。

- 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。

- 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 /home/azureuser/.ssh/authorized_keys 文件里。下面请跟着一起来做操作：

[root@localhost ~]# mkdir /home/azureuser/.ssh

[root@localhost ~]# chmod 700 /home/azureuser/.ssh 首先创建/home/azureuser/.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，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 /home/azureuser/.ssh/authorized_keys 文件。

[root@localhost ~]# vi /home/azureuser/.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。

- 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 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：

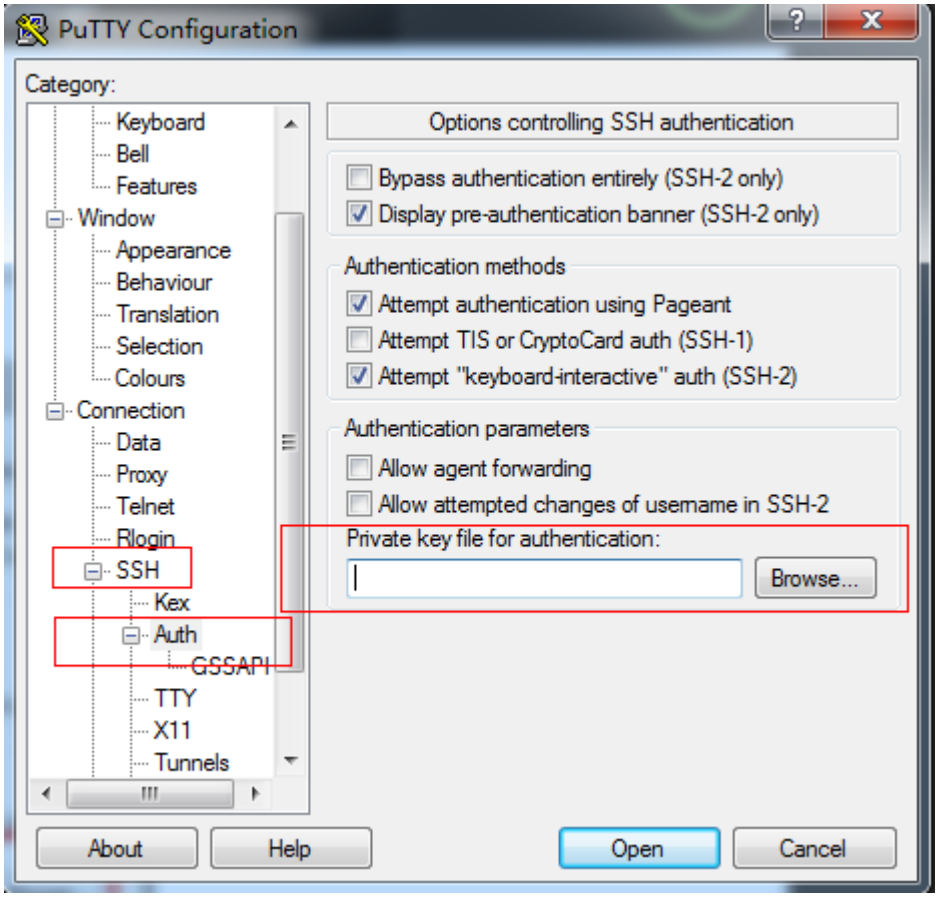
[root@localhost ~]# vi /etc/selinux/config 回车后，把光标移动到 “SELINUX=enforcing” 按一下 i 键，进入编辑模式，修改为

SELINUX=disabled

按 “Esc”, 输入 :wq 回车，然后重启系统

- 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”.



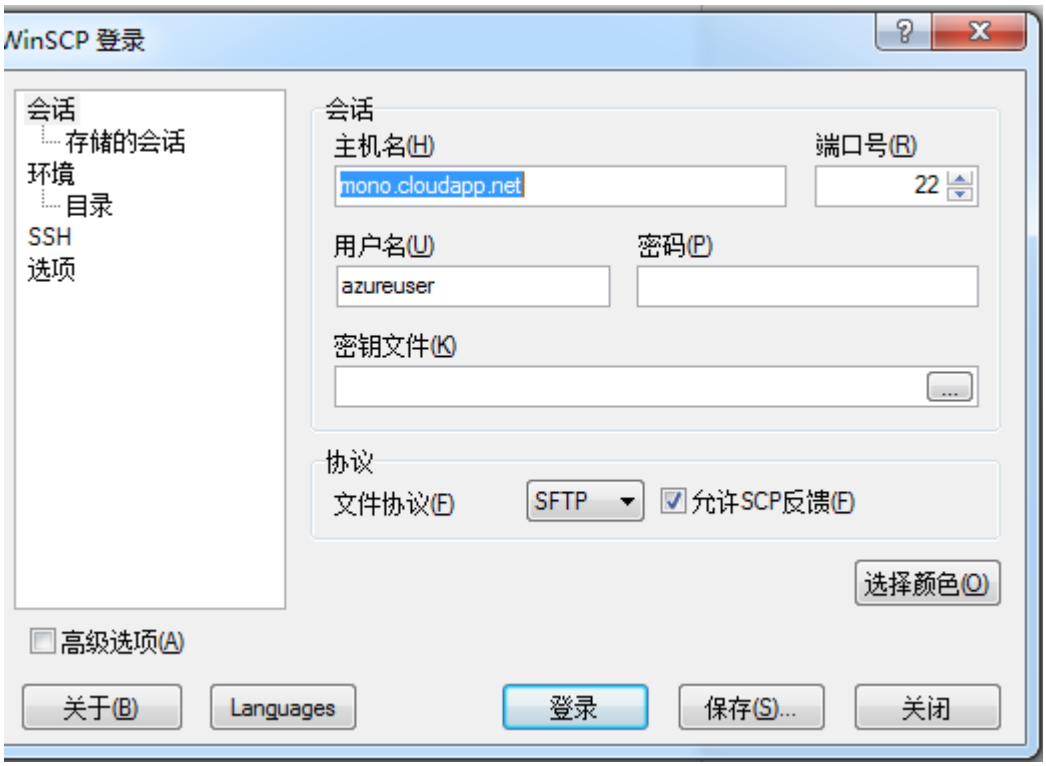
- 使用密钥验证登陆 Linux

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。

现在不再输入 azureuser 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 azureuser 后会直接登陆系统。

使用 WinSCP 在 windows 和 Linux 中进行文件传输

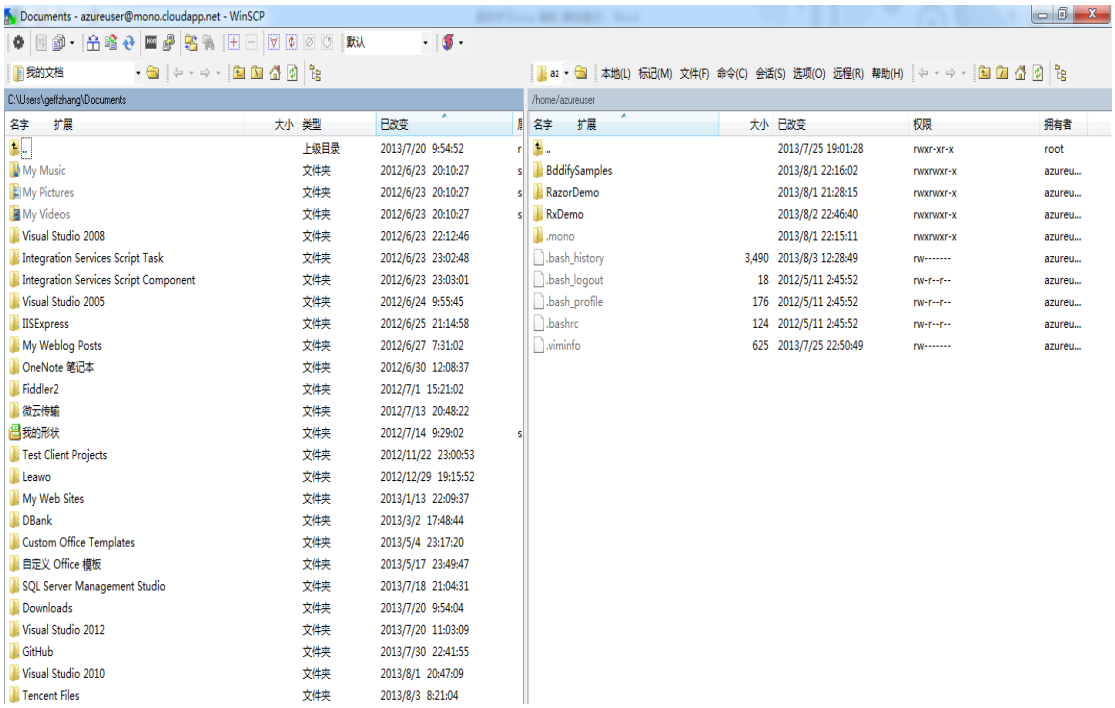
在本章前面提供的 WinSCP 下载地址里有个列表，其中有一项 Portable executables 是绿色版，不需要安装，推荐使用。 下载完成之后打开可执行文件，填写登录信息，选择协议之后，就可以进行图形化管理了。



我们只需要填写 3 个地方：

- 主机名 ： hostname 是虚拟机的 IP 地址，我的是 58.215.140.108。
- 用户名 ： username 就是登陆虚拟机时的用户名。
- 密码 ： password 就是登陆虚拟机时的密码。

点击登陆就进入到 Linux 系统了：

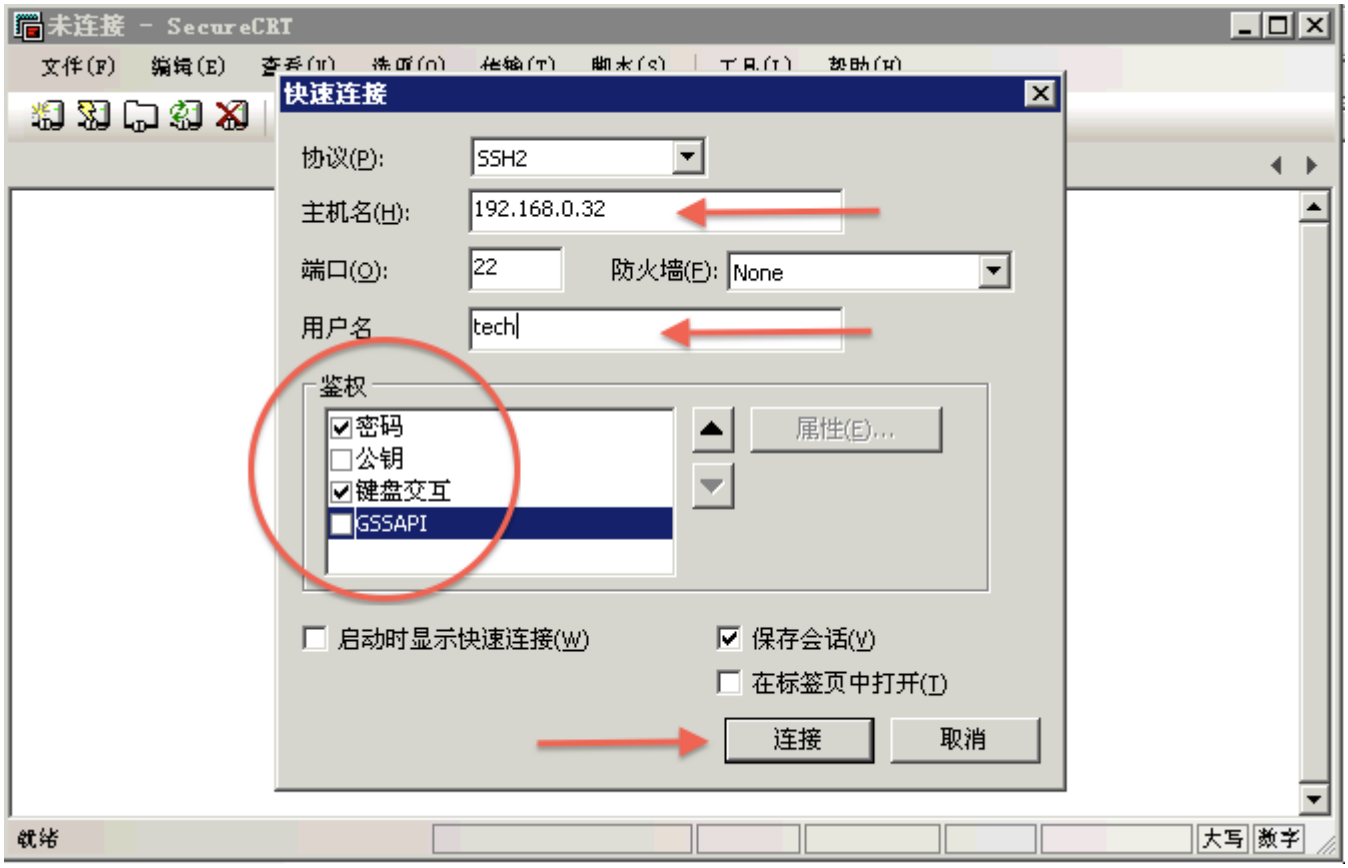


SecureCRT

SecureCRT 是一款支持 SSH（SSH1 和 SSH2）的终端仿真程序，同时支持 Telnet 和 rlogin 协议。SecureCRT 是一款用于连接运行包括 Windows、UNIX 和 VMS 的远程系统的理想工具。通过使用内含的 VCP 命令行程序可以进行加密文件的传输。有流行 CRTTelnet 客户机的所有特点，包括：自动注册、对不同主机保持不同的特性、打印功能、颜色设置、可变屏幕尺寸、用户定义的键位图和优良的 VT100，VT102，VT220 和 ANSI 竞争。能从命令行中运行或从浏览器中运行。其它特点包括文本手稿、易于使用的工具条、用户的键位图编辑器、可定制的 ANSI 颜色等.SecureCRT 的 SSH 协议支持 DES，3DES 和 RC4 密码和密码与 RSA 鉴别。

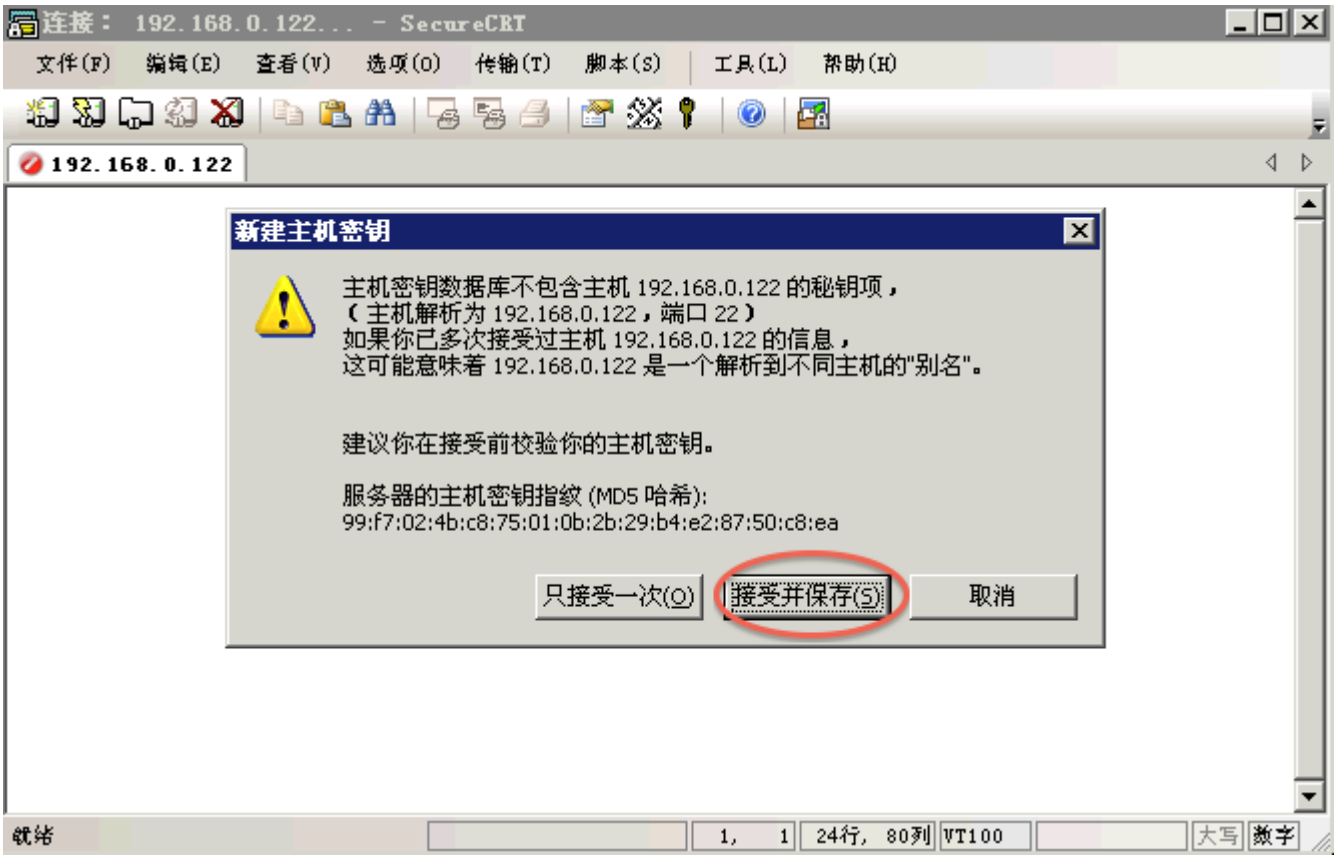
配置方法：

第一部分： 保存一个远程主机的连接信息，第一次启动后，会弹出如下画面：

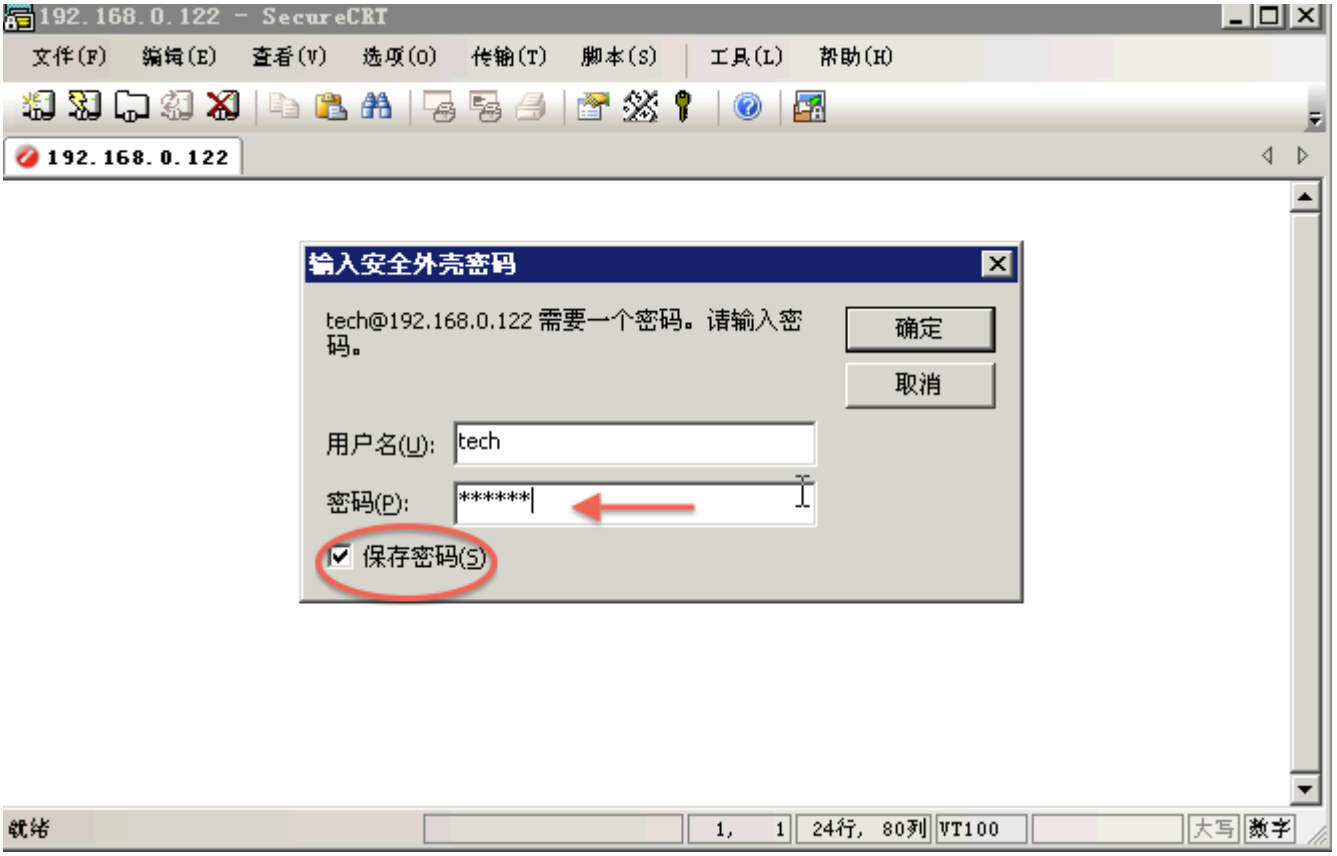


按照图上的例子，把主机名，用户名都填写上，鉴定 部分就留下密码和键盘交互两部分。

然后是如下的画面，按照图上选择接受，就可以了。



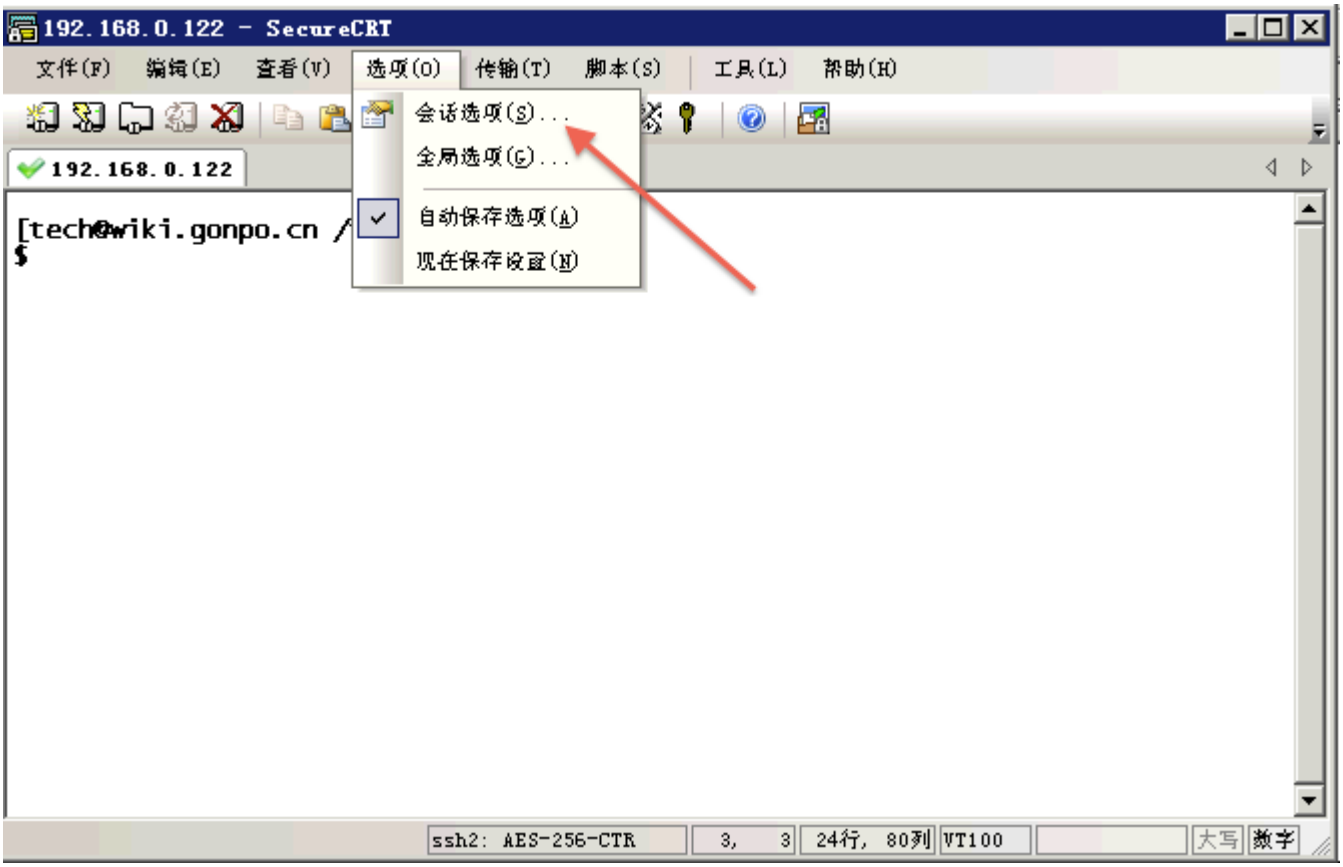
然后是如下的画面，按照图上的例子，输入密码，保存密码。



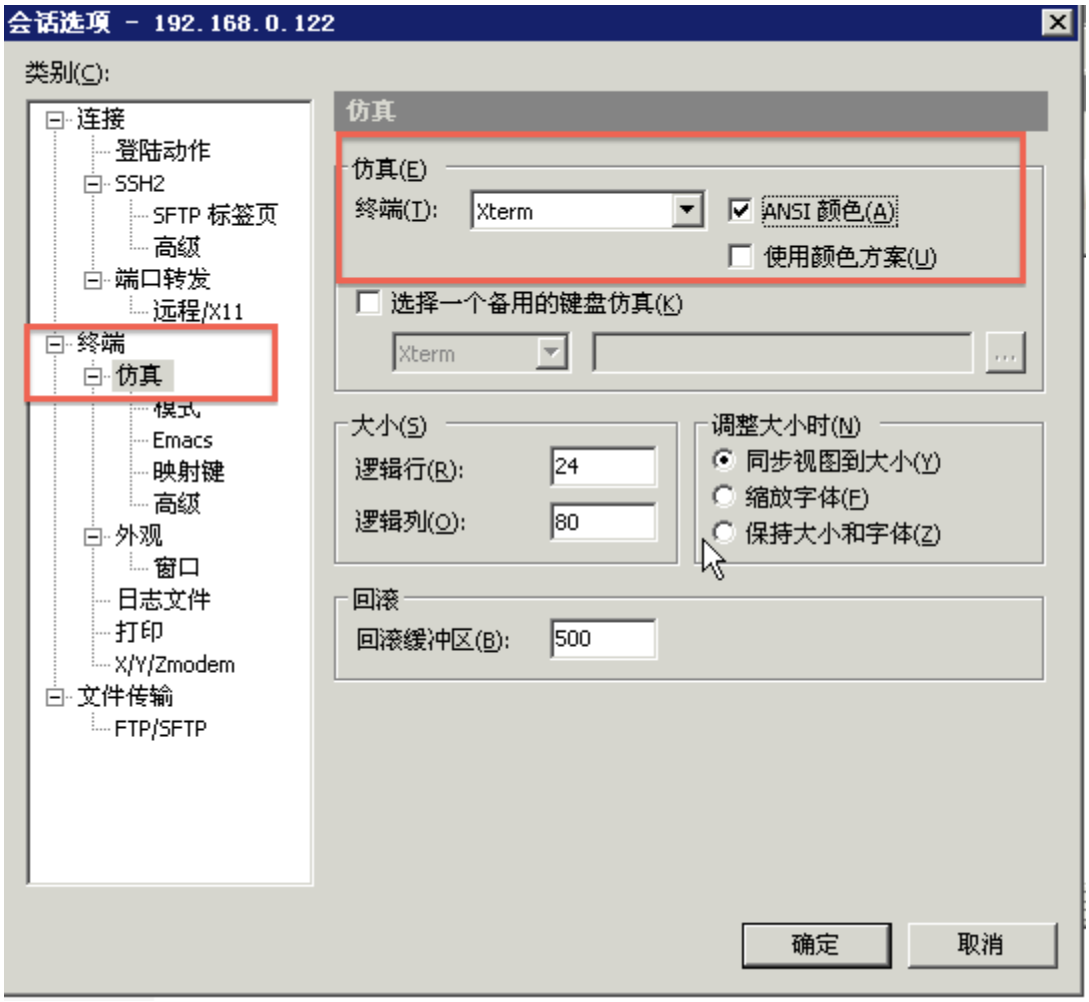
然后就可以登录到远程服务器了。

第二部分：调整远程登录的显示界面，让显示清晰，不容易看错。（**很重要**）

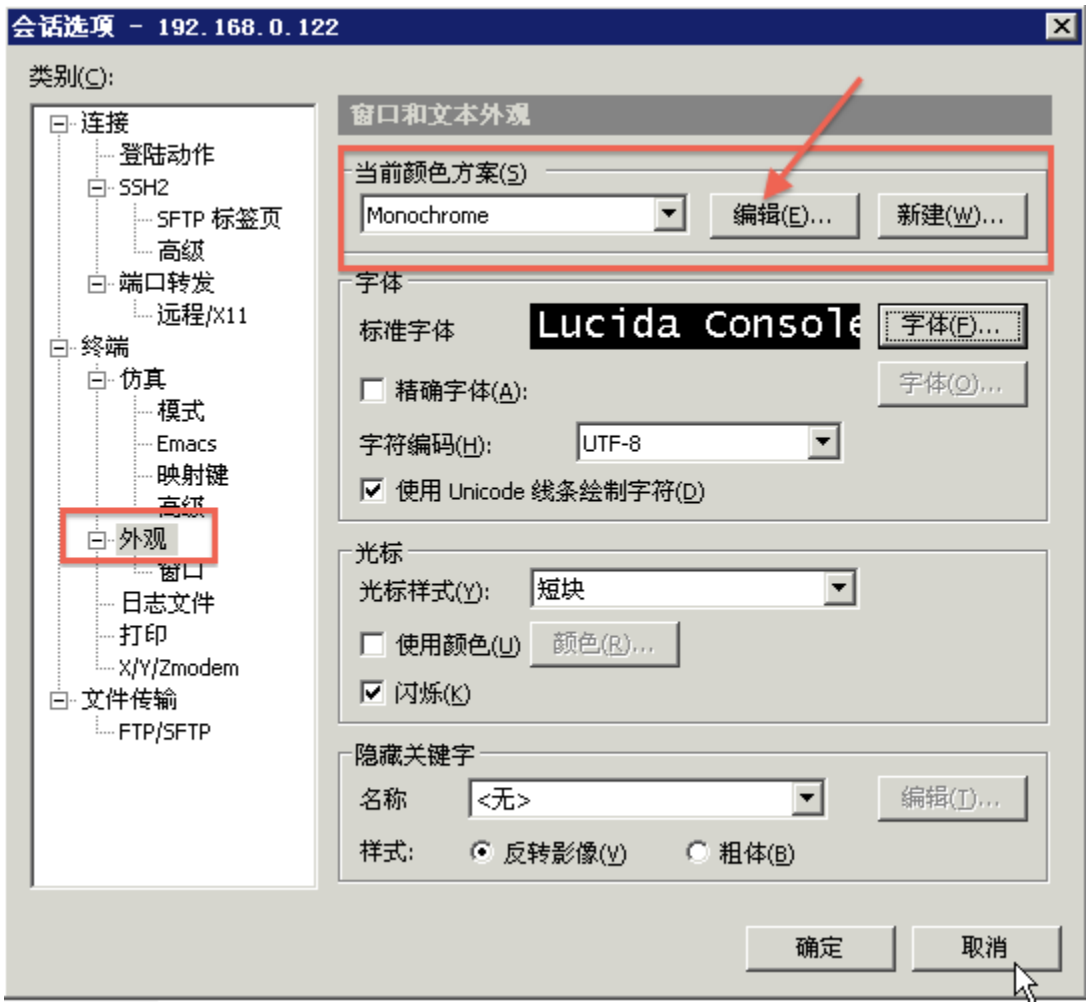
登录后，默认是白底，黑字，字体比较小，极不清晰，容易出错。下面进行相关的调整。
首先按照下图的示例，选择 选项-会话选项



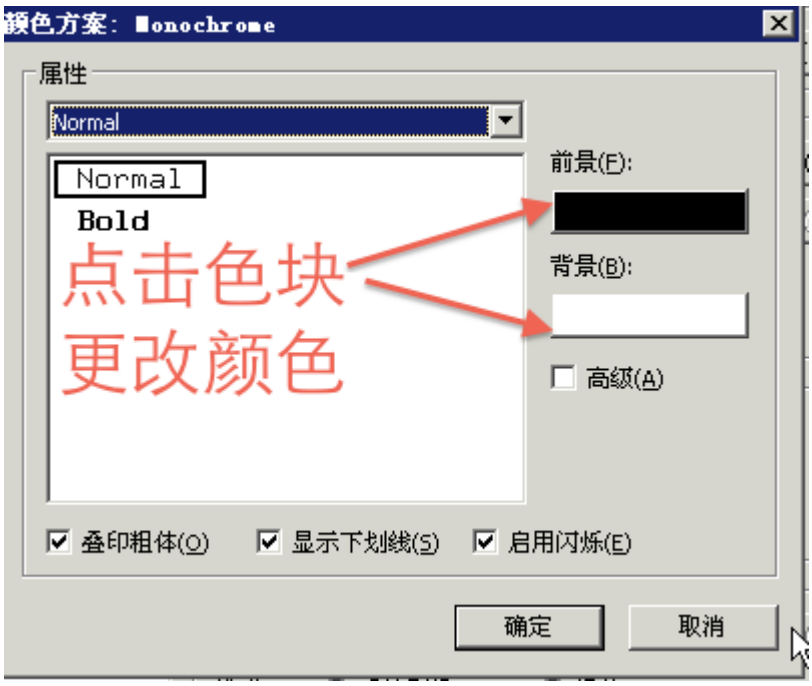
按照下图的要求，在 终端-仿真中，把终端类型改为 Xterm，勾选 “ANSI”颜色， 但是，“使用颜色方案”千万不要选择。

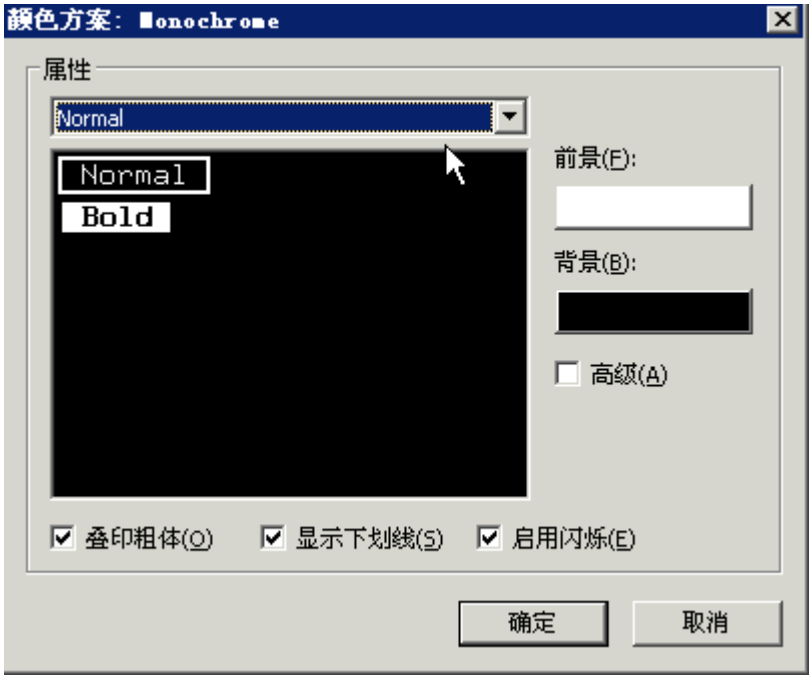


按照下图的要求，调整外观，首先 在 monochrome 中，选择编辑：

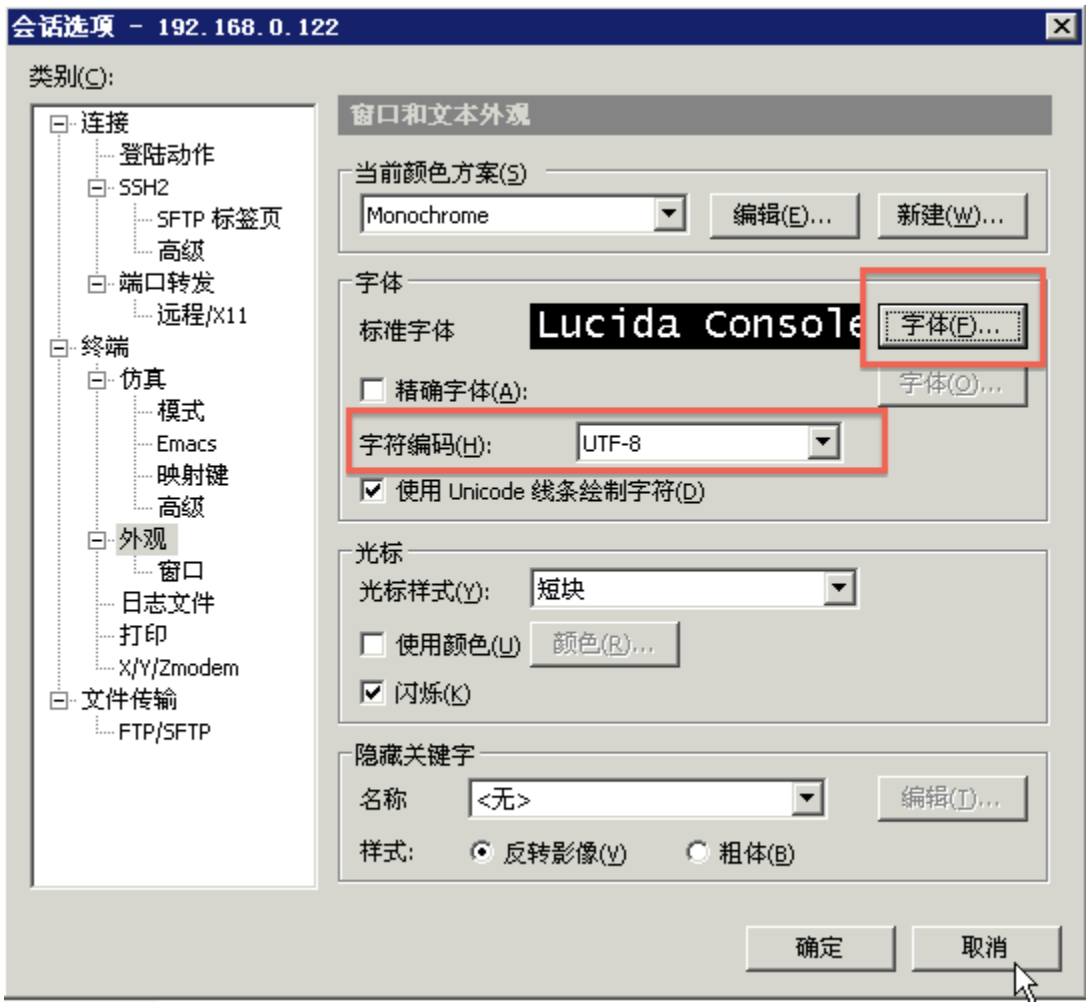


在编辑选择窗口中，把默认的白底黑字，调整为黑底白字。





再回到外观界面，更改字体和环境字符变量 ，这个很重要。把字体改为自己舒适的大小，字符环境一定要设置为 UTF-8 。



SSH 服务器设置

基本上，所有的 sshd 服务器详细设置都放在 /etc/ssh/sshd_config 里面！不过，每个 Linux distribution 的默认设置都不太相同，所以我们有必要来了解一下整个设置值的意义是什么才好！ 同时请注意，在默认的文件内，只要是默认有出现且被批注的配置值 (配置值前面加 #)，即为『默认值!』，你可以依据它来修改的哩。

```
[root@www ~]# vi /etc/ssh/sshd_config
# 1. 关于 SSH Server 的整体设置，包含使用的 port 啦，以及使用的密码演算方式
# Port 22
# SSH 默认使用 22 这个端口，也可以使用多个 port，即重复使用 port 这个设置项目！
# 例如想要开放 sshd 在 22 与 443 ，则多加一行内容为：『 Port 443 』
# 然后重新启动 sshd 这样就好了！不过，不建议修改 port number 啦！

Protocol 2
# 选择的 SSH 协议版本，可以是 1 也可以是 2 。

# ListenAddress 0.0.0.0
# 监听的主机适配器！举个例子来说，如果你有两个 IP，分别是 192.168.1.100 及
# 192.168.100.254，假设你只想要让 192.168.1.100 可以监听 sshd ，那就这样写：
# 『 ListenAddress 192.168.1.100 』默认值是监听所有接口的 SSH 要求

# PidFile /var/run/sshd.pid
# 可以放置 SSHD 这个 PID 的文件！上述为默认值

# LoginGraceTime 2m
# 当使用者连上 SSH server 之后，会出现输入密码的画面，在该画面中，
# 在多久时间内没有成功连上 SSH server 就强迫断线！若无单位则默认时间为秒！

# Compression delayed
# 指定何时开始使用压缩数据模式进行传输。有 yes, no 与登入后才将数据压缩 (delayed)

# 2. 说明主机的 Private Key 放置的文件，默认使用下面的文件即可！
# HostKey /etc/ssh/ssh_host_key              # SSH version 1 使用的私钥
# HostKey /etc/ssh/ssh_host_rsa_key       # SSH version 2 使用的 RSA 私钥
# HostKey /etc/ssh/ssh_host_dsa_key       # SSH version 2 使用的 DSA 私钥
# 还记得我们在主机的 SSH 联机流程里面谈到的，这里就是 Host Key ～

# 3. 关于登录文件的信息数据放置与 daemon 的名称！
SyslogFacility AUTHPRIV
# 当有人使用 SSH 登入系统的时候，SSH 会记录信息，这个信息要记录在什么 daemon
name
# 底下？默认是以 AUTH 来设置的，即是 /var/log/secure 里面！什么？忘记了！
#其他可用的 daemon name 为：DAEMON,USER,AUTH,
```

LOCAL0,LOCAL1,LOCAL2,LOCAL3,LOCAL4,LOCAL5,

LogLevel INFO
登录记录的等级！嘿嘿！任何信息！同样的，忘记了就回去参考！

4. 安全设置项目！极重要！
4.1 登入设置部分
PermitRootLogin yes
是否允许 root 登入！默认是允许的，但是建议设置成 no！

StrictModes yes
是否让 sshd 去检查用户家目录或相关文件的权限数据，
这是为了担心使用者将某些重要文件的权限设错，可能会导致一些问题所致。
例如使用者的 ~.ssh/ 权限设错时，某些特殊情况下会不许用户登入

PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys
是否允许用户自行使用成对的密钥系统进行登入行为，仅针对 version 2。
至于自制的公钥数据就放置于用户家目录下的 .ssh/authorized_keys 内

PasswordAuthentication yes
密码验证当然是需要的！所以这里写 yes 啰！

PermitEmptyPasswords no
若上面那一项如果设置为 yes 的话，这一项就最好设置为 no ，
这个项目在是否允许以空的密码登入！当然不许！

4.2 认证部分
RhostsAuthentication no
本机系统不使用 .rhosts，因为仅使用 .rhosts 太不安全了，所以这里一定要设置为 no

IgnoreRhosts yes
是否取消使用 ~/.ssh/.rhosts 来做为认证！当然是！

RhostsRSAAuthentication no
这个选项是专门给 version 1 用的，使用 rhosts 文件在 /etc/hosts.equiv
配合 RSA 演算方式来进行认证！不要使用啊！

HostbasedAuthentication no
这个项目与上面的项目类似，不过是给 version 2 使用的！

IgnoreUserKnownHosts no
是否忽略根目录内的 ~/.ssh/known_hosts 这个文件所记录的主机内容？
当然不要忽略，所以这里就是 no 啦！

ChallengeResponseAuthentication no
允许任何的密码认证！所以，任何 login.conf 规定的认证方式，均可适用！
但目前我们比较喜欢使用 PAM 模块帮忙管理认证，因此这个选项可以设置为 no 喔！

UsePAM yes
利用 PAM 管理使用者认证有很多好处，可以记录与管理。
所以这里我们建议你使用 UsePAM 且 ChallengeResponseAuthentication 设置为 no

4.3 与 Kerberos 有关的参数设置！因为我们没有 Kerberos 主机，所以底下不用设置！
KerberosAuthentication no
KerberosOrLocalPasswd yes
KerberosTicketCleanup yes
KerberosTgtPassing no

4.4 底下是有关在 X-Window 底下使用的相关设置！
X11Forwarding yes
X11DisplayOffset 10
X11UseLocalhost yes
比较重要的是 X11Forwarding 项目，他可以让窗口的数据透过 ssh 信道来传送喔！
在本章后面比较进阶的 ssh 使用方法中会谈到。

4.5 登入后的项目：
PrintMotd yes
登入后是否显示出一些信息呢？例如上次登入的时间、地点等等，默认是 yes
亦即是打印出 /etc/motd 这个文件的内容。但是，如果为了安全，可以考虑改为 no ！

PrintLastLog yes
显示上次登入的信息！可以啊！默认也是 yes ！

TCPKeepAlive yes
当达成联机后，服务器会一直传送 TCP 封包给客户端藉以判断对方式否一直存在联机。
不过，如果联机时中间的路由器暂时停止服务几秒钟，也会让联机中断喔！
在这个情况下，任何一端死掉后，SSH 可以立刻知道！而不会有僵尸程序的发生！
但如果你的网络或路由器常常不稳定，那么可以设置为 no 的啦！

UsePrivilegeSeparation yes
是否权限较低的程序来提供用户操作。我们知道 sshd 启动在 port 22 ，
因此启动的程序是属于 root 的身份。那么当 student 登入后，这个设置值
会让 sshd 产生一个属于 sutdent 的 sshd 程序来使用，对系统较安全

MaxStartups 10
同时允许几个尚未登入的联机画面？当我们连上 SSH ，但是尚未输入密码时，
这个时候就是我们所谓的联机画面啦！在这个联机画面中，为了保护主机，

所以需要设置最大值，默认最多十个联机画面，而已经建立联机的不计算在这十个当中

4.6 关于用户抵挡的设置项目：
DenyUsers *
设置受抵挡的使用者名称，如果是全部的使用者，那就是全部挡吧！
若是部分使用者，可以将该账号填入！例如下列！
DenyUsers test

DenyGroups test
与 DenyUsers 相同！仅抵挡几个群组而已！

5. 关于 SFTP 服务与其他的设置项目！
Subsystem sftp /usr/lib/ssh/sftp-server
UseDNS yes
一般来说，为了要判断客户端来源是正常合法的，因此会使用 DNS 去反查客户端的主机名
不过如果是在内网互连，这项目设置为 no 会让联机达成速度比较快。

基本上，CentOS 默认的 sshd 服务已经算是挺安全的了，不过还不够！建议你 (1)将 root 的登入权限取消；(2)将 ssh 版本设定为 2 。其他的设定值就请你依照自己的喜好来设置了。通常不建议进行随便修改啦！另外，如果你修改过上面这个档案(/etc/ssh/sshd_config)，那么就必需要重新启动一次 sshd 这个 daemon 才行！

第四章 Linux 文件与目录管理

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’，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。

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 /home/azureuser/.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/jexus 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 azureuser 此时用户所在的路径为 /home/azureuser 第一个 cd 命令后跟‘/home’第二个 cd 命令后跟‘azureuser’，并没有斜杠，这个‘azureuser’

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 ‘azureuser’ 账户，所以运行 cd 后，会进入 azureuser 账户的根目录 ‘/home/azureuser’. 后面跟目录名，则会直接切换到指定目录下：

```
[azureuser@mono ~]$ cd
[azureuser@mono ~]$ cd /usr/jexus
[azureuser@mono jexus]$ pwd
/usr/jexus
[azureuser@mono jexus]$
```

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：

```
[azureuser@mono jexus]$ cd /etc/passwd
-bash: cd: /etc/passwd: Not a directory
[azureuser@mono jexus]$
```

./ 表示当前目录，../ 表示当前目录的上一级目录：

```
[azureuser@mono jexus]$ pwd
/usr/jexus
[azureuser@mono jexus]$ cd ./
[azureuser@mono jexus]$ pwd
/usr/jexus
[azureuser@mono jexus]$ cd ../
[azureuser@mono usr]$ pwd
/usr
[azureuser@mono usr]$
```

上例中，首先在 /usr/jexus 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/ 目录下，即 /usr/jexus 目录的上一级目录。

创建和删除目录

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，这里不做重点解释。 ‘-p’ 这个选项很管用，先来做个试验，您会一目了然的：

```
[azureuser@mono tmp]$ mkdir /tmp/test/log
mkdir: cannot create directory `/tmp/test/log': No such file or directory
```

[azureuser@mono tmp]$ mkdir -p /tmp/test/log
[azureuser@mono tmp]$ ls /tmp/test
Log
当我们想创建 /tmp/test/log 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：

[azureuser@mono tmp]$ ls -ld /tmp/test/log
drwxrwxr-x. 2 azureuser azureuser 4096 Aug 3 05:14 /tmp/test/log
[azureuser@mono tmp]$ mkdir -p /tmp/test/log
[azureuser@mono tmp]$ ls -ld /tmp/test/log
drwxrwxr-x. 2 azureuser azureuser 4096 Aug 3 05:14 /tmp/test/log
ls 命令，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/log’ 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。该命令通常很少使用，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，通常用的比较多。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。
[azureuser@mono tmp]$ ls /tmp/test
log
[azureuser@mono tmp]$ rmdir /tmp/test
rmdir: failed to remove `/tmp/test': Directory not empty
[azureuser@mono tmp]$ rmdir /tmp/test/log
[azureuser@mono tmp]$ ls /tmp/test
[azureuser@mono tmp]$
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。在这里只介绍最常用的两个选项。

‘-r’ ：删除目录用的选项，等同于 rmdir.
‘-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。

关于 rm，通常使用最多便是 ‘-rf’ 两个选项组合用了。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，那就杯具了，要多加小心。

环境变量 PATH

不知道你意识到没有，为什么我们输入很多命令时是直接打出了命令，而没有去使用这些命令的绝对路径？我们可以用 Which 命令看下我们的上面执行的命令的文件：

[azureuser@mono tmp]$ which rm
/bin/rm
[azureuser@mono tmp]$ which ls
alias ls='ls --color=auto'
/bin/ls
我们看到了 rm 命令是执行的/bin/rm, ls 实际上执行的执行的是/bin/ls。‘which’ 这个命令只用来查询某个命令的绝对路径，不常使用。

我们直接输入命令而不要输入命令的绝对路径是因为环境变量 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。

[azureuser@mono tmp]$ echo $PATH
/usr/lib64/qt-3.3/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/azureuser/bin
因为/bin 在 PATH 的设置中，所以自然就可以找到 ls 了。如果您把 Mono 安装到 /usr/local/mono 底下的话，当您执行 mono 的时候，他就是不理您？怎么办？这是因为 PATH 没有 /usr/local/mono 这个目录，而您又将 mono 安装到 /usr/local/mono 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!’

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 /usr/local/mono 的路径加入 $PATH 当中！如何增加？可以使用命令 PATH=$PATH:/ usr/local/mono
另一种方法就是使用绝对路径。

文件操作相关的命令

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。
[azureuser@mono test]$ mkdir log
[azureuser@mono test]$ cp log log2

```
cp: omitting directory `log'
[azureuser@mono test]$ cp -r log log2
[azureuser@mono test]$ ls -l
total 8
drwxrwxr-x. 2 azureuser azureuser 4096 Aug  3 06:29 log
drwxrwxr-x. 2 azureuser azureuser 4096 Aug  3 06:30 log2
```

-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。面简单做一个小试验，很快您就会明白-i 选项的作用了。

```
[azureuser@mono test]$ cd log
[azureuser@mono log]$ ls
[azureuser@mono log]$ touch 111
[azureuser@mono log]$ touch 222
[azureuser@mono log]$ cp -i 111 222
cp: overwrite `222'? n
[azureuser@mono log]$ echo 'abc' > 111
[azureuser@mono log]$ echo 'def' > 222
[azureuser@mono log]$ cat 111 222
abc
def
[azureuser@mono log]$ cp 111 222
[azureuser@mono log]$ cat 111
abc
[azureuser@mono log]$ cat 222
abc
```

例子中的 ‘touch’命令，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。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。

```
[azureuser@mono test]$ mkdir dira dirb
[azureuser@mono test]$ ls
dira  dirb  log  log2
[azureuser@mono test]$ mv dira dirc
[azureuser@mono test]$ ls
dirb  dirc  log  log2
```

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’.

```
[azureuser@mono test]$ mv dirc dirb
[azureuser@mono test]$ ls
dirb  log  log2
[azureuser@mono test]$ 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。

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，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。

```
[azureuser@mono test]$ echo 'hello mono on centos' > dirb/hellomono

[azureuser@mono test]$ echo 'Jexus web server' >> dirb/hellomono

[azureuser@mono test]$ cat dirb/hellomono

hello mono on centos

Jexus web server

[azureuser@mono test]$ cat -n dirb/hellomono

     1  hello mono on centos

     2  Jexus web server
```

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

```
[azureuser@mono test]$ cat -An dirb/hellomono
     1  hello mono on centos$
     2  Jexus web server$
```

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。

[azureuser@mono test]$ tac dirb/hellomono
Jexus web server
hello mono on centos

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’ 肯定是看不到前面的内容的，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。

```
[azureuser@mono test]$ head /usr/etc/mono/config

<configuration>

        <dllmap dll="i:cygwin1.dll" target="libc.so.6" os="!windows" />

        <dllmap dll="libc" target="libc.so.6" os="!windows"/>

        <dllmap dll="intl" target="libc.so.6" os="!windows"/>

        <dllmap dll="intl" name="bind_textdomain_codeset"
target="libc.so.6" os="solaris"/>

        <dllmap dll="libintl" name="bind_textdomain_codeset"
target="libc.so.6" os="solaris"/>
```

```
        <dllmap dll="libintl" target="libc.so.6" os="!windows"/>

        <dllmap dll="i:libxslt.dll" target="libxslt.so"
os="!windows"/>

        <dllmap dll="i:odbc32.dll" target="libodbc.so" os="!windows"/>

        <dllmap dll="i:odbc32.dll" target="libiodbc.dylib" os="osx"/>
```

‘-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。

```
[azureuser@mono test]$ tail -n2  /usr/etc/mono/config

        <dllmap dll="gdiplus.dll" target="/usr/lib/libgdiplus.so" />

</configuration>
```

-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。

```
[azureuser@mono test]$ tail -f /usr/jexus/log/default

08-03 03:55:12:  100.43.83.154 mono.cloudapp.net GET
/post/2013/05/16/11.aspx

08-03 03:56:29:  100.43.83.154 mono.cloudapp.net GET
/post/2013/05/04/2.aspx

08-03 04:07:24:  100.43.83.154 mono.cloudapp.net GET
/post/2013/05/04/5.aspx

08-03 04:32:23:  66.249.74.227 mono.cloudapp.net GET /category/0.aspx

08-03 04:33:10:  66.249.74.227 mono.cloudapp.net GET /default.aspx
type=post&name=4

08-03 04:36:42:  100.43.83.154 mono.cloudapp.net GET
/category/SQLite.aspx

08-03 04:37:43:  100.43.83.154 mono.cloudapp.net GET /admin
```

08-03 04:46:02: 66.249.74.227 mono.cloudapp.net GET /default.aspx type=archive&date=201305

08-03 04:47:33: 100.43.83.154 mono.cloudapp.net GET /admin/-->/admin/default.aspx

08-03 04:57:36: 100.43.83.154 mono.cloudapp.net GET /author/admin.aspx

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。

```
[azureuser@mono ~]$ ls -la
total 44
drwx------. 6 azureuser azureuser 4096 Aug  2 14:24 .
drwxr-xr-x. 3 root      root      4096 Jul 25 11:01 ..
-rw-------. 1 azureuser azureuser 3490 Aug  3 04:28 .bash_history
-rw-r--r--. 1 azureuser azureuser   18 May 10  2012 .bash_logout
-rw-r--r--. 1 azureuser azureuser  176 May 10  2012 .bash_profile
-rw-r--r--. 1 azureuser azureuser  124 May 10  2012 .bashrc
drwxrwxr-x. 2 azureuser azureuser 4096 Aug  1 14:16 BddifySamples
drwxrwxr-x. 3 azureuser azureuser 4096 Aug  1 14:15 .mono
drwxrwxr-x. 2 azureuser azureuser 4096 Aug  1 13:28 RazorDemo
drwxrwxr-x. 3 azureuser azureuser 4096 Aug  2 14:46 RxDemo
-rw-------. 1 azureuser azureuser  625 Jul 25 14:50 .viminfo
```

linux 文件属性

上面的例子中，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；

```
[azureuser@mono ~]$ ls -l /etc/rc.local
lrwxrwxrwx. 1 root root 13 Dec 18  2012 /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。

‘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 .，要特别说一下，之前的 CentOS 5 是没有这个点的，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 以后再介绍，您只要了解一下即可。

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 列为该文件的创建日期或者最近的修改日期，分别为月份日期以及时间。
第 9 列，文件名。

修改文件的权限

上一节介绍的 Linux 文件属性的第一列是文件的权限，文件的权限如何进行修改呢，本节介绍相关的命令。

- 更改所属组 chgrp

语法：chgrp [组名] [文件名]

```
[azureuser@mono ~]$ sudo groupadd testgroup
[sudo] password for azureuser:
```

```
[azureuser@mono ~]$ touch test1
[azureuser@mono ~]$ ls –l test1
-rw-rw-r--. 1 azureuser azureuser        0 Aug    3 07:26 test1
[azureuser@mono ~]$ sudo chgrp testgroup test1
[azureuser@mono ~]$ ls -l test1
-rw-rw-r--. 1 azureuser testgroup 0 Aug    3 07:26 test1
[azureuser@mono ~]$
```

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。

```
[azureuser@mono test]$ ls -l dirb
total 8
drwxrwxr-x. 2 azureuser azureuser 4096 Aug    3 06:39 dirc
-rw-rw-r--. 1 azureuser azureuser      38 Aug    3 06:49 hellomono
[azureuser@mono test]$ ls -ld dirb/
drwxrwxr-x. 3 azureuser azureuser 4096 Aug    3 06:48 dirb/
[azureuser@mono test]$ sudo chgrp testgroup dirb/
[azureuser@mono test]$ ls -ld dirb/
drwxrwxr-x. 3 azureuser testgroup 4096 Aug    3 06:48 dirb/
[azureuser@mono test]$ ls -l dirb/
total 8
drwxrwxr-x. 2 azureuser azureuser 4096 Aug    3 06:39 dirc
-rw-rw-r--. 1 azureuser azureuser      38 Aug    3 06:49 hellomono
```

‘chgroup’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：

```
[azureuser@mono test]$ sudo chgrp -R testgroup dirb/
[azureuser@mono test]$ ls -l dirb/
total 8
drwxrwxr-x. 2 azureuser testgroup 4096 Aug    3 06:39 dirc
-rw-rw-r--. 1 azureuser testgroup      38 Aug    3 06:49 hellomono
```

- 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。

值得提一下的是，在 linux 系统中，默认一个目录的权限为 775，而一个文件的默认权限为 664.

当我们在系统中新建一个文件或目录 时，系统会自动赋予该文件或目录一个初始访问权限(Value)，我们称为默认权限，默认权限与文件系统的 umask 值有关。可以在终端下直接输入 umask 来查看当前系统的 umask 值。例如：

[azureuser@mono log]$ umask

0002

也许你会问为什么自己系统的 umask 值只有三位数？这里第一位是特殊权限位，我们暂且不考虑它，先看后面三位数就可以了。那么这个值具体是怎样运用的呢？其实 umask 本质是一个掩码，为此我们有必要先了解 Linux 文件系统权限的表示方法及文 件与目录的约定权限。

1．文件系统的权限表示方法有两种。

一是直接用 r、w、x 来代表文件的所有者(u)、用户组(g)、其他用 户(o)对某一文件或目录的读、写、执行(x)权限，称为字符表示法，例如上面的-rw-rw-rw-。二是用一组(三位)八进制数来间接表示文件或目录 的权属,称为数字表示法，例如某文件的权限为 755。

注：所谓数字表示法是指将读取(r)，写入 (w) 和执行(x) 分别以 4、2、l 来代表，没有授予的部份就表示值为 0，然后再把所授予的权限相加而成。

2．在不考虑 umask 的情况下，新建一个文件或目录的权限应该都是 rwxrwxrwx，但为了提高安全性，会默认去除新建文件的可执行权限（x），而对于新 建的目录，可执行位 x 与可否被允许进入该目录有关，因此 Linux 约定：

新建文件的权属是-rw-rw-rw-，权限值是 666。

新建目录的权属是 drwxrwxrwx，权限值是 777。

那么，在给定系统 umask 的情况下，新建文件或目录的默认权限如下赋予：

新建文件的约定权限 — UMASK 表示的权限 = 文件的默认权限

新建目录的约定权限 — UMASK 表示的权限 = 目录的默认权限

这 里的减号(-)更确切地说是屏蔽的意思。

例如当前系统的 UMASK 值为 002，或者表示为--- --- -w-。那么新建一个文件或目录的默认权限为：

新建文件的约定权限 — UMASK 表示的权限 = 文件的默认权限

-rw-rw-rw- — --- --- -w- = -rw-rw-r-

表示在约定权限的基础上屏蔽除所 有者(u)、同组用户(g)之外其他用户(o)的可写权限。此时当你在系统中新建一个文件时，该文件的默认权限就是-rw-rw-r-了，这样你就会理解 为什么 UMASK 是一个掩码值了。

- chmod 命令

功能：chmod 命令是非常重要的，用于改变文件或目录的访问权限.用户用它控制文件或目录的访问权限.

语法：该命令有两种用法。一种是包含字母和操作符表达式的文字设置法；另一种是包含数字的数字设置法。

1. 文字设置法

chmod [who] [+ | - | =] [mode] 文件名?

参数：
操作对象 who 可是下述字母中的任一个或者它们的组合：
u 表示“用户（user）”，即文件或目录的所有者。
g 表示“同组（group）用户”，即与文件属主有相同组 ID 的所有用户。
o 表示“其他（others）用户”。
a 表示“所有（all）用户”。它是系统默认值。
操作符号可以是：
+ 添加某个权限。
- 取消某个权限。
= 赋予给定权限并取消其他所有权限（如果有的话）。

设置 mode 所表示的权限可用下述字母的任意组合：
r 可读。
w 可写。
x 可执行。
X 只有目标文件对某些用户是可执行的或该目标文件是目录时才追加 x 属性。
s 在文件执行时把进程的属主或组 ID 置为该文件的文件属主。方式“u＋s”设置文件的用户 ID 位，“g＋s”设置组 ID 位。
t 保存程序的文本到交换设备上。
u 与文件属主拥有一样的权限。
g 与和文件属主同组的用户拥有一样的权限。
o 与其他用户拥有一样的权限。
文件名：以空格分开的要改变权限的文件列表，支持通配符。
在一个命令行中可给出多个权限方式，其间用逗号隔开。例如：chmod g+r，o+r example 使同组和其他用户对文件 example 有读权限。
2. 数字设置法
我们必须首先了解用数字表示的属性的含义：0 表示没有权限，1 表示可执行权限，2 表示可写权限，4 表示可读权限，然后将其相加。所以数字属性的格式应为 3 个从 0 到 7 的八进制数，其顺序是（u）（g）（o）。
例如，如果想让某个文件的属主有"读/写"二种权限，需要把 4（可读）+2（可写）＝6（读/写).
数字设置法的一般形式为：
chmod [mode] 文件名?
例子：
（1）文字设置法：
例 1：$ chmod a+x sort
即设置文件 sort 的属性为：
文件属主（u） 增加执行权限
与文件属主同组用户（g） 增加执行权限
其他用户（o） 增加执行权限
例 2：$ chmod ug+w，o-x text
即设置文件 text 的属性为：
文件属主（u） 增加写权限
与文件属主同组用户（g）增加写权限

其他用户（o） 删除执行权限
例 3：$ chmod u+s a.out
假设执行 chmod 后 a.out 的权限为（可以用 ls － l a.out 命令来看）：
－rws--x--x 1 inin users 7192 Nov 4 14:22 a.out
并且这个执行文件要用到一个文本文件 shiyan1.c，其文件存取权限为“－rw-------”，即该文件只有其属主具有读写权限。
当其他用户执行 a.out 这个程序时，他的身份因这个程序暂时变成 inin（由于 chmod 命令中使用了 s 选项），所以他就能够读取 shiyan1.c 这个文件（虽然这个文件被设置为其他人不具备任何权限），这就是 s 的功能。
因此，在整个系统中特别是 root 本身，最好不要过多的设置这种类型的文件（除非必要）这样可以保障系统的安全，避免因为某些程序的 bug 而使系统遭到入侵。
例 4：$ chmod a－x mm.txt
$ chmod –x mm.txt
$ chmod ugo–x mm.txt
以上这三个命令都是将文件 mm.txt 的执行权限删除，它设置的对象为所有使用者。
（2）数字设置法：
例 1： $ chmod 644 mm.txt
$ ls –l
即设置文件 mm.txt 的属性为：
-rw-r--r-- 1 inin users 1155 Nov 5 11:22 mm.txt
文件属主（u）inin 拥有读、写权限
与文件属主同组人用户（g）拥有读权限
其他人（o） 拥有读权限
例 2： $ chmod 750 wch.txt
$ ls –l
-rwxr-x--- 1 inin users 44137 Nov 12 9:22 wchtxt
即设置 wchtxt 这个文件的属性为：
文件主本人（u）inin 可读/可写/可执行权
与文件主同组人（g）可读/可执行权
其他人（o） 没有任何权限

文件搜索

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。
[azureuser@mono log]$ which mono
/usr/bin/mono
2. ‘find’ 这个搜索工具是用的最多的一个，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

‘-name filename’直接查找该文件名的文件，这个选项使用很多

下面我们实例来学习一下：

```
[azureuser@mono log]$ find /tmp -mtime -1
/tmp
/tmp/root-temp-aspnet-0
/tmp/root-temp-aspnet-0/ea5f23af_2d14
/tmp/root-temp-aspnet-0/ea5f23af_2d14/assembly
/tmp/root-temp-aspnet-0/ea5f23af_2d14/assembly/shadow
/tmp/root-temp-aspnet-0/ea5f23af_2d14/assembly/shadow/fa3c43db
/tmp/root-temp-aspnet-0/ea5f23af_2d14/assembly/shadow/fa3c43db/ef4e9b91_1572d84a_00000001
/tmp/root-temp-aspnet-0/ea5f23af_2d14/assembly/shadow/fa3c43db/ef4e9b91_1572d84a_00000001/__AssemblyInfo__.ini
/tmp/root-temp-aspnet-0/ea5f23af_2d14/assembly/shadow/2685600c
/tmp/root-temp-aspnet-0/ea5f23af_2d14/assembly/shadow/2685600c/33f7b846_1572d84a_00000001
/tmp/root-temp-aspnet-0/ea5f23af_2d14/assembly/shadow/2685600c/33f7b846_1572d84a_00000001/__AssemblyInfo__.ini
/tmp/root-temp-aspnet-0/ea5f23af_2d14/assembly/shadow/04751168
/tmp/root-temp-aspnet-0/ea5f23af_2d14/assembly/shadow/04751168/1107c922_1572d84a_00000001
/tmp/root-temp-aspnet-0/ea5f23af_2d14/assembly/shadow/04751168/1107c922_1572d84a_00000001/__AssemblyInfo__.ini
/tmp/root-temp-aspnet-0/ea5f23af_2d14/assembly/shadow/bbc95d93
/tmp/root-temp-aspnet-0/ea5f23af_2d14/assembly/shadow/bbc95d93/aebb85d9_1572d84a_00000001
/tmp/root-temp-aspnet-0/ea5f23af_2d14/assembly/shadow/bbc95d93/aebb85d9_1572d84a_00000001/__AssemblyInfo__.ini
/tmp/root-temp-aspnet-0/ea5f23af_2d14/assembly/shadow/b7cb5d00
/tmp/root-temp-aspnet-0/ea5f23af_2d14/assembly/shadow/b7cb5d00/a2b9854a_1572d84a_00000001
/tmp/root-temp-aspnet-0/ea5f23af_2d14/assembly/shadow/b7cb5d00/a2b9854a_1572d84a_00000001/__AssemblyInfo__.ini
/tmp/root-temp-aspnet-0/ea5f23af_2d14/assembly/shadow/03b043c1
/tmp/root-temp-aspnet-0/ea5f23af_2d14/assembly/shadow/03b043c1/16c29b8b_1572d84a_00000001
/tmp/root-temp-aspnet-0/ea5f23af_2d14/assembly/shadow/03b043c1/16c29b8b_1572d84a_00000001/__AssemblyInfo__.ini
/tmp/root-temp-aspnet-0/9274a0c1
```

/tmp/root-temp-aspnet-0/9274a0c1/App_Web_3efcdf85_0.cs
/tmp/root-temp-aspnet-0/9274a0c1/App_Web_3efcdf85.dll.mdb
/tmp/root-temp-aspnet-0/92

看到这里，您对这三个 time 是不是有些晕了，那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。

[azureuser@mono log]$ stat /tmp/test
  File: `/tmp/test'
  Size: 4096 Blocks: 8 IO Block: 4096 directory
Device: 801h/2049d Inode: 1057537 Links: 6
Access: (0775/drwxrwxr-x) Uid: (500/azureuser) Gid: (500/azureuser)
Access: 2013-08-06 13:09:54.881626761 +0000
Modify: 2013-08-03 07:34:24.000000000 +0000
Change: 2013-08-06 03:06:02.561627161 +0000
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, 这三个 time stamp 都放在 inode 中。若 mtime, atime 修改 inode 就一定会改, 既然 inode 改了, 那 ctime 也就跟着要改了。

直接查找该文件名的文件，例如我们要查下 jws 文件。
[azureuser@mono log]$ sudo find / -name jws
[sudo] password for azureuser:
/usr/jexus/jws
/usr/local/src/jexus-5.4/data/jws

Linux 文件系统

大家都知道 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，不会再有如 Ext3 例文件最大为 16TB 的限制。

Ext2

最老的 Linux 文件系统还在使用 ext2 文件系统，在该类型下不采用任何日志记录。因此，文件系统崩溃后的 fsck 需要很长的恢复时间，这并不适合用于许多用途。用户不应该把数据库放在 ext2 的卷上。虽然理论上可以工作，但在许多已知的情况下，例如在十分复杂的 fsck 进程中用户做了些错误操作，这样就会打破数据库原本的写入顺序。不过由于没有任何日志的写入，所以 ext2 卷的速度较快，常用于较少需要日志的任务。

Ext3

Ext3 在 ext2 文件系统的基础上，增加了日志文件。如果日志是空的（服务器正常关闭的情况下），用户可以将 ext3 文件系统设置为 ext2 模式。它向后兼容 ext2，并且可以相互转换，从 ext2 转换成 ext3 或者 ext3 转换成 ext2。
在 Ext3 中，有三种可用的日志级别，在挂载文件系统的时候以选项的方式指定。

- data = writeback: 数据变更不进行任何日志记录。记录元数据的变更，但是与数据块有关的写顺序不能保证。在文件系统崩溃之后，文件在完成写操作的部分后有额外的内容，此时用户具有新旧混合的数据。
- data = ordered: 记录元数据，但不记录数据变化。不过在所有的情况下，元数据的写入只发生于与其相关联的数据已被写入后，即“ordered”状态。在崩溃发生后，像回写模式一样可以在文件中找到新老混合的数据，但用户不能以不正确的长度结束文件。与完全日志模式相比，最重要的不同是用户修改的块已经在磁盘上，而数据变更时与其关联的元数据没有变更。当文件是新的或者块最后已写入，增大了文件大小，ordered 模式的行为和功能上相当于日志。
- data = journal：文件数据和文件系统元数据在写入主文件系统前已写入日志。

在这些模式中进行选择并不想起初看起来那么容易。在所有的情况下，可能会认为 writeback 模式会是最快的，但是在具有写缓存功能的系统尤其是在 ordered 模式和 writeback 模式中，速度的差别很小。但是从性能的角度看，当基础磁盘硬件比较优秀的时候不大可能去选择 writeback 模式，因为这会使得所面临的风险更加让人难以接受。
Ext3 的一个局限性就是在 Intel 和 AMD 处理器环境下，越来越与当前的磁盘驱动器大小有关，ext3 文件系统的最大大小为 16TB，单个文件大小最大不得超过 2TB。

Ext4

在 Linux 内核 2.6.28 中，公布了作为 ext3 换代产品的 ext4 标准。到了 2.6.30 版本之间发布了较多关于延迟分配等问题的修复补丁，直到 2.6.32 一些涉及到 fsync 处理缺陷都得到了完全解决。CentOS6 和 Ubuntu 10.04 都采用这个版本的 ext4,并将长期支持 ext4。
从 ext3 更新为 ext4 最大的提升在于能够更好的处理些屏障（Write Barriers）和 fsync 操作。

XFS

XfS 文件系统是 SGI 开发的高级日志文件系统，XFS 极具伸缩性，非常健壮。所幸的是 SGI 将其移植到了 Linux 系统中。
主要特性包括以下几点：

- 数据完全性

采用 XFS 文件系统，当意想不到的宕机发生后，首先，由于文件系统开启了日志功能，所以你磁盘上的文件不再会意外宕机而遭到破坏了。不论目前文件系统上存储的文件与数据有多少，文件系统都可以根据所记录的日志在很短的时间内迅速恢复磁盘文件内容。

- 传输特性

XFS 文件系统采用优化算法，日志记录对整体文件操作影响非常小。XFS 查询与分配存储空间非常快。xfs 文件系统能连续提供快速的反应时间。笔者曾经对 XFS、JFS、Ext3、ReiserFS 文件系统进行过测试，XFS 文件文件系统的性能表现相当出众。

- 可扩展性

XFS 是一个全 64-bit 的文件系统，它可以支持上百万 T 字节的存储空间。对特大文件及小尺寸文件的支持都表现出众，支持特大数量的目录。最大可支持的文件大 小为 263 = 9 x 1018 = 9 exabytes，最大文件系统尺寸为 18 exabytes。

XFS 使用高的表结构(B+树)，保证了文件系统可以快速搜索与快速空间分配。XFS 能够持续提供高速操作，文件系统的性能不受目录中目录及文件数量的限制。

- 传输带宽

XFS 能以接近裸设备 I/O 的性能存储数据。在单个文件系统的测试中，其吞吐量最高可达 7GB 每秒，对单个文件的读写操作，其吞吐量可达 4GB 每秒。

CentOS 7 使用 XFS 替代 ext4 作为默认的文件系统。虽然在 CentOS 6 中已经提供了 XFS 的选项，但是默认还是使用 ext4。XFS 支持高达 500TB 的容量，而 ext4 仅支持 50TB。不幸的是,除了备份和恢复之外目前还没有方法可以让用户从 ext4 或 btrfs 文件系统上迁移到 XFS。

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。

- 纯文本档(ASCII)：这是 Linux 系统中最多的一种文件类型， 称为纯文本档是因为内容为我们人类可以直接读到的数据，例如数字、字母等等。 几乎只要我们可以用来做为设置的文件都属于这一种文件类型。 举例来说，你可以下达『 cat ~/.bashrc 』就可以看到该文件的内容。 (cat 是将一个文件内容读出来的指令)
- 二进制文件(binary)：，我们的系统其实仅认识且可以执行二进制文件(binary file) 在 Linux 当中的可执行文件(scripts, 文字型批处理文件不算)就是这种格式的 举例来说，我们经常使用的 cat 命令就是一个 binary file。
- 数据格式文件(data)： 有些程序在运作的过程当中会读取某些特定格式的文件，那些特定格式的文件可以被称为数据文件 (data file)。举例来说，我们的 Linux 在使用者登入时，都会将登录的数据记录在 /var/log/wtmp 那个文件内，该文件是一个 data file，他能够透过 last 这个指令读出来！ 但是使用 cat 时，会读出乱码 因为它是属于一种特殊格式的文件。

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。在后续章节会介绍。

4）字符设备或块设备文件：与系统周边相关的一些文件，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’.

5) 套接口文件: 这种类型的文件通常被用在网络上的数据承接了。我们可以启动一个程序来监听客户端的要求， 而客户端就可以透过这个 socket 来进行数据的沟通了。第一个属性为 [s]， 最常在/var/run 这个目录中看到这种文件类型了。当我们启动 Jexus 服务器时，会产生两个 socket 的文件，一个是日志服务的，一个是子进程与父进程的消息通道。

```
[root@mono azureuser]# ls -lh /var/run/jexus
total 4.0K
srwxr-xr-x. 1 root root 0 Aug  3 23:39 jwslog_socket
srwxrwxrwx. 1 root root 0 Aug  3 23:39 jwsmsg_socket
-rw-r--r--. 1 root root 4 Aug  3 23:39 jws.pid
```

注意这个文件的属性的第一个字符是 s。我们了解一下就行了。

Linux 文件扩展名

对于文件扩展名，windows 底下大家都非常熟悉了，能被执行的文件扩展名通常是 .exe .bat 等等，而在 Linux 底下，只要你的权限当中具有 x 的话，例如[-rwx-r-xr-x] 即代表这个文件可以被执行。在 linux 系统中，文件的扩展名并没有具体意义，也就是说，您加或者不加，都无所谓，主要看文件的属性。但是为了容易区分，我们习惯给文件加一个扩展名，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。底下有数种常用的扩展名：

- *.sh ： 脚本或批处理文件 (scripts)，因为批处理文件为使用 shell 写成的，所以扩展名就编成 .sh ；

- •*Z, *.tar, *.tar.gz, *.zip, *.tgz： 经过打包的压缩文件。这是因为压缩软件分别为 gunzip, tar 等等的，由于不同的压缩软件，而取其相关的扩展名！
- •*.html, *.aspx：网页相关文件，分别代表 HTML 语法与 c# 语法的网页文件！.html 的文件可使用网页浏览器来直接开启，至于 .php 的文件， 则可以通过客户端的浏览器来 服务端浏览，以得到运算后的网页结果！

基本上，Linux 系统上的文件名真的只是让你了解该文件可能的用途而已， 真正的执行与否仍然需要权限的规范才行！例如虽然有一个文件为可执行文件， 如常见的/bin/ls 这个显示文件属性的指令，不过，如果这个文件的权限被修改成无法执行时， 那么 ls 就变成不能执行！

Linux 文件长度与文件名的限制

在 Linux 底下，使用默认的 Ext2/Ext3 文件系统时，针对文件的档名长度限制为：

- 单一文件或目录的最大容许文件名为 255 个字符；
- 包含完整路径名称及目录 (/) 之完整档名为 4096 个字符。

由于 Linux 在文字接口下的一些指令操作关系，一般来说，你在设置 Linux 底下的文件名时，最好可以避免一些特殊字符比较好！例如底下这些：

* ? > < ; & ! [ ] | \ ' " ` ( ) { }

因为这些符号在文字接口下，是有特殊意义的！另外，文件名的开头为小数点『.』时， 代表这个文件为『隐藏文件』! 同时，由于指令下达当中，常常会使用到 -option 之类的选项，所以最好也避免将文件档名的开头以 - 或 + 来命名！

Linux 链接文件

Linux 链接文件类型有两种，类似于 windows 系统下的快捷方式。但是 Linux 链接文件类型又与 windows 系统的不同。

Linux 文件系统最重要的特点之一是它的文件链接。链接是对文件的引用，这样您可以让文件在文件系统中多处被看到。不过，在 Linux 中，链接可以如同原始文件一样来对待。链接可以与普通的文件一样被执行、编辑和访问。对系统中的其他应用程序而言，链接就是它所对应的原始文件。当您通过链接对文件进行编辑时，您编辑的实际上是原始文件。链接不是副本。有两种类型的链接：硬链接和符号链接(软链接)。

硬链接

硬链接只能引用同一文件系统中的文件。它引用的是文件在文件系统中的物理索引（也称为 inode）。当您移动或删除原始文件时，硬链接不会被破坏，因为它所引用的是文件的物理数据而不是文件在文件结构中的位置。硬链接的文件不需要用户有访问原始文件的权限，也不会显示原始文件的位置，这样有助于文件的安全。如果您删除的文件有相应的硬链接，那么这个文件依然会保留，直到所有对它的引用都被删除。

符号链接(软链接)

符号链接(软链接)是一个指针，指向文件在文件系统中的位置。符号链接可以跨文件系统，

甚至可以指向远程文件系统中的文件。符号链接只是指明了原始文件的位置，用户需要对原始文件的位置有访问权限才可以使用链接。如果原始文件被删除，所有指向它的符号链接也就都被破坏了，而去使用 ‘ll’ 查看发现颜色也变了。它们会指向文件系统中并不存在的一个位置。目录也是可以软链接的。

两种链接都可以通过命令 ln 来创建。ln 默认创建的是硬链接。使用 -s 开关可以创建符号链接。

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。

例：$ ln - s lunch /home/lun

用户为当前目录下的文件 lunch 创建了一个符号链接/home/lun。

第五章 Linux 系统用户及用户组管理

Linux 系统是一个多用户多任务的分时操作系统，任何一个要使用系统资源的用户，都必须首先向系统管理员申请一个账号，然后以这个账号的身份进入 系统。用户的账号一方面可以帮助系统管理员对使用系统的用户进行跟踪，并控制他们对系统资源的访问；另一方面也可以帮助用户组织文件，并为用户提供安全性 保护。每个用户账号都拥有一个惟一的用户名和各自的口令。用户在登录时键入正确的用户名和口令后，就能够进入系统和自己的主目录。
实现用户账号的管理，要完成的工作主要有如下几个方面：

- 用户账号的添加、删除与修改。
- 用户口令的管理。
- 用户组的管理。

用户账号的管理工作主要涉及到用户账号的添加、修改和删除。添加用户账号就是在系统中创建一个新账号，然后为新账号分配用户号、用户组、主目录和登录 Shell 等资源。刚添加的账号是被锁定的，无法使用。

认识账号管理的灵魂文件（passwd 和 shadow）

关于某个用户的帐号的管理，最重要的就是/etc/passswd 和/etc/shadow，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。；而关于用户群组的管理，最重要的就是/etc/group 和/etc/gshadow 两个文件。
下面来介绍一下关于用户管理的/etc/passwd 和/etc/shadow 这两个个文件。

[root@mono azureuser]# 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
[root@mono azureuser]#

/etc/passwd 是一个纯文本文件，每行采用了相同的格式：由 ‘:’ 分割成 7 个字段，每个字段的具体含义是:

1）“用户名”是代表用户账号的字符串。通常长度不超过 8 个字符，并且由大小写字母和/或数字组成。登录名中不能有冒号(，因为冒号在这里是分隔符。为了兼容起见，登录名中最好不要包含点字符(.)，并且不使用连字符(-)和加号(+)打头。

2） “口令”一些系统中，存放着加密后的用户口令字。虽然这个字段存放的只是用户口令的加密串，不是明文，但是由于/etc/passwd 文件对所有用户都可读，所以这仍是一个安全隐患。因此，现在许多 Linux 系统（如 SVR4）都使用了 shadow 技术，把真正的加密后的用户口令字存放到/etc/shadow 文件中，而在/etc/passwd 文件的口令字 段中只存放一个特殊的字符，例如“x”或者“*”。

3）“用户标识号”是一个整数，系统内部用它来标识用户，也叫做 uid。一般情况下它与用户名是一一对应的。如果几个用户名对应的用户标识号是一样的，系统内部将把它们视为同一个用户，但是它们可以有不同的口令、不同的主目录以及不同的登录 Shell 等。 通常用户标识号的取值范围是 0～65 535。0 是超级用户 root 的标识号，1～99 由系统保留，作为管理账号，普通用户的标识号从 100 开始。在 Linux 系统中，这个界限是 500。

4）“组标识号”字段记录的是用户所属的用户组，也叫做 gid。它对应着/etc/group 文件中的一条记录，其实/etc/group 和/etc/passwd 基本上类似。

5) “注释性描述”字段记录着用户的一些个人情况，例如用户的真实姓名、电话、地址等，这个字段并没有什么实际的用途。在不同的 Linux 系统中，这个字段的格式并没有统一。在许多 Linux 系统中，这个字段存放的是一段任意的注释性描述文字，用做 finger 命令的输出。

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

7) 用户登录后，要启动一个进程，负责将用户的操作传给内核，这个进程是用户登录到系统后运行的命令解释器或某个特定的程序，即 Shell。Shell 是用户与 Linux 系统之间的接口。Linux 的 Shell 有许多种，每种都有不同的特点。常用的有 sh(Bourne Shell), csh(C Shell), ksh(Korn Shell), tcsh(TENEX/TOPS-20 type C Shell), bash(Bourne Again Shell)等。系统管理员可以根据系统情况和用户习惯为用户指定某个 Shell。如果不指定 Shell，那么系统使用 sh 为默认的登录 Shell，即这 个字段的值为/bin/sh。

再来看看/etc/shadow 这个文件，和/etc/passwd 一样，每行代表一个用户，使用符号“：”分割成 9 个字段。

[root@mono azureuser]# cat /etc/shadow | head -n 5
root:*LOCK*:14600::::::
bin:*:15513:0:99999:7:::
daemon:*:15513:0:99999:7:::
adm:*:15513:0:99999:7:::
lp:*:15513:0:99999:7:::

1、 username:用户名，跟/etc/passwd 对应，这个字段不用过多解释，不过系统管理员需要知道一点，在系统安装完毕后，除了 root 特权帐户外，系统还会自动建立不少的用户。这些用户往往用来完成一些特殊的作业。为此在没有了解这些帐户的用途之前，不要轻易更改系统创建的用户。而系统管理员的用户通常情况下都是保存在最后面。也就是说，其记录的顺序是按照帐号创建的先后顺序来保存的。这对于系统管理员迅速定位帐户信息能够提供一定的帮助。

2、 passwd：密码，如果为空，表示用户密码为空，如果是“*”，表示该用户有效但不能登录。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。为此在一些安全性要求比较高的企业中，建议大家采用 MD5 编码。如此的话，这里显示的字符串长度就会有 256 个字符，其破解的难度就会大的多。有时会这个字符串为空白，就表示这个帐户没有设置密码。在设置密码的时候，用户最好不要依靠密码的长度来提高安全性;而最好是通过密码的复杂性，如字符与阿拉伯数字混用等手段来提高操作系统的安全。

3、 lastchg：表示从 1970 年 1 月 1 日起到上次修改口令所经过的天数。这个数字是这样计算得来的，例如上次更改密码的日期为 2013 年 1 月 1 日，则这个值就是 ‘365 x (2013-1970) + 1 = 15696’.

4、 min：表示两次修改口令之间至少经过的天数，默认是 0，即不限制。

5、 max：这个字段表示密码更改后，距离下次一定要更改口令的天数。有时候系统管理员为了安全考虑，会给密码设置一个有效期限。当这个期限到后，就会提示用户更改密码。特别是当 Linux 服务器暴露在互联网上时，频繁更改密码确实是提高其安全性的一个重要手段。如输入 60，就代表距离上次更改密码 60 天后，用户就必须更改一个新的口令。如果逾期不进行更改的话，就会受到警告，最后这个帐户可能就会被注销。

6、 warn：表示口令失效前多少天内系统向用户发出警告若这个值设置成 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、 inactive：表示禁止登陆前用户名还有效的天数，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、 expire：账号的生命周期，表示用户被禁止登陆的时间，其算法而 3 相同，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、 flag：无意义，未使用，留做扩展。

上面的分析可以看出 shadow 文件中基本上包含了帐户与密码管理中的所有内容。作为系统管理员，除了要了解这个文件的基本结构之外，还需要了解这个文件中哪些字段可以更改。通常情况下，前面三个字段最好不要更改。后面的内容可以根据需要来调整。不过在账号有效期限这个字段中，不要为 root 帐户设置账号期限。因为 root 账号主要用来做管理用。如果为其设置有效期限，而期限过期后，那么就没有用户能够来完成系统的维护工作了。通过配置文件来维护帐户、密码等管理策略，要比通过其他方式简单的多。

新增/删除用户和用户组

- 新增一个组

命令 : groupadd

语法 : groupadd [-g GID] groupname

[root@mono azureuser]# groupadd grptest1

[root@mono azureuser]# tail -n1 /etc/group

grptest1:x:503:

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。可以用-g 选项自定义 gid

[root@mono azureuser]# groupadd -g 510 grptest2

- 删除组

命令 : groupdel

[root@mono azureuser]# groupdel grptest2

[root@mono azureuser]# tail -n3 /etc/group

testgroup:x:501:

user1:x:502:

grptest1:x:503:

这个命令没有特殊选项，但有一种情况不能删除组：用户组中还包含有用户。

- 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@mono azureuser]# useradd test1

[root@mono azureuser]# tail -n1 /etc/passwd

test1:x:502:504::/home/test1:/bin/bash

[root@mono azureuser]# tail -n1 /etc/group

test1:x:504:

‘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@mono azureuser]# useradd -u 510 -g 513 -M -s /sbin/nologin user11

useradd: group '513' does not exist

[root@mono azureuser]# useradd -u 510 -g 503 -M -s /sbin/nologin user11

[root@mono azureuser]# useradd -u 511 -g grptest1 user12
[root@mono azureuser]# tail -n2 /etc/passwd
user11:x:510:503::/home/user11:/sbin/nologin
user12:x:511:503::/home/user12:/bin/bash
[root@mono azureuser]# tail -n2 /etc/group
grptest1:x:503:
test1:x:504:
‘-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。

- 删除账户

命令 : userdel

语法 : userdel [-r] username
[root@mono azureuser]# ls -ld /home/user12
drwx------. 2 user12 grptest1 4096 Aug 7 23:45 /home/user12
[root@mono azureuser]# userdel user12
[root@mono azureuser]# ls -ld /home/user12
drwx------. 2 511 grptest1 4096 Aug 7 23:45 /home/user12
‘-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。

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：

- 长度大于 10 个字符；
- 密码中包含大小写字母数字以及特殊字符 ‘*’, ‘&’, ‘%’ 等；
- 不规则性（不要出现 root, happy, love, linux, jexus, mono，11111111 等等单词或者数字）；
- 不要带有自己名字、公司名字、自己电话、自己生日等。

我们也可以偷懒，安装一个命令 mkpasswd，这个命令在包“expect”里，用命令 yum install -y expect 即可完成安装。安装好后，输入命令：
[azureuser@mono ~]$ sudo mkpasswd
y7tb9r^CE
[azureuser@mono ~]$

“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。

只有 root 才可以修改其他账户的密码，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。

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。

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。

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，系统就可能被别人控制了，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 预先设置的，有很多系统是没有安装 sudo 命令的，可以通过 yum install -y sudo 安装。安装后配置文件在 /etc/sudoers，默认 root 能够 sudo 是因为这个文件中有一行

```
## Allow root to run any commands anywhere
root        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。

使用密码记录工具 keepass 来保存密码

在第一章，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？下面介绍给您一款记录密码的软件，使用.NET 编写的软件，通过 Mono 可以支持 Linux，Mac

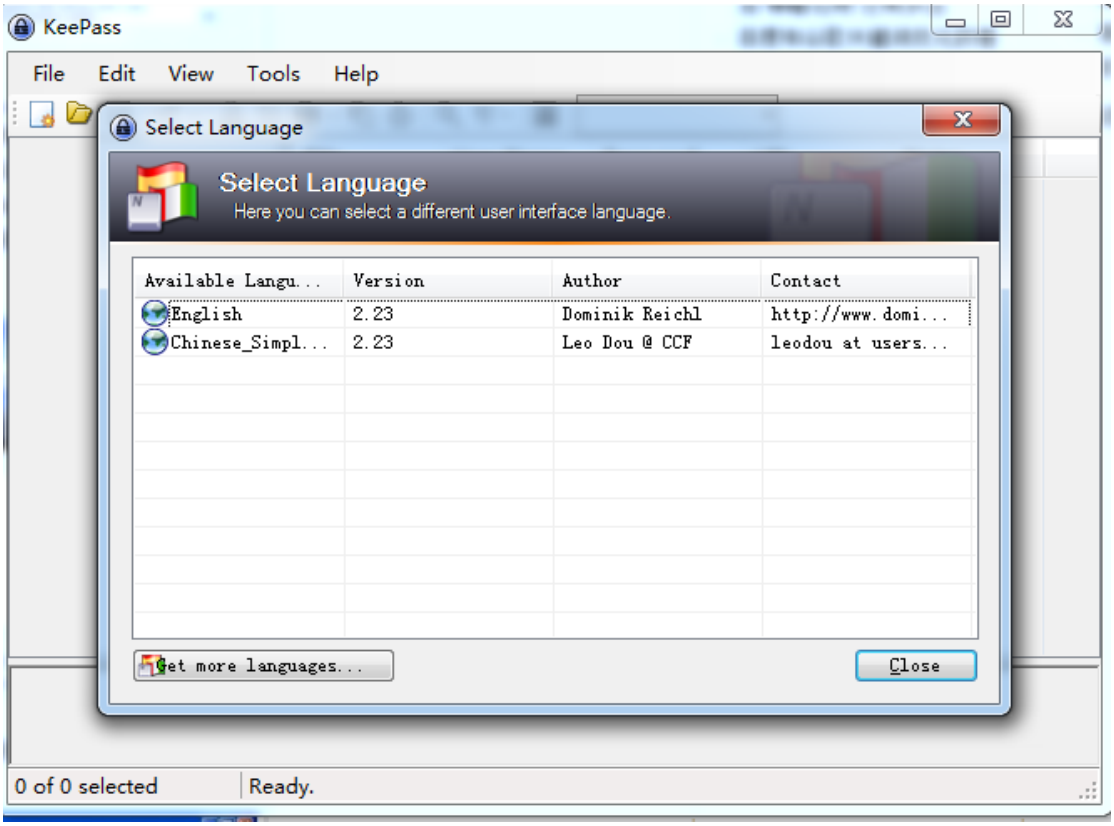
等。而且还有 Android 手机版本[https://keepass2android.codeplex.com/]。

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. KeePass（KeePass Password Safe）密码管理系统是一套类似数据库管理的密码管理软件，通过密码和密钥，它能够提供一个足够安全的密码存储空间。只要你记住这一个管理密码，妥善保管好密钥文件和数据库文件，基本上可以安枕无忧了。同时 KeePass 也有强大的密码生成功能，绝对比你自己想的要安全。它的操作方式也极为 简单，没有复杂的步骤。

- 下载及安装

下载最新版本的 keepass，当前最新的发行版本为 2.23, 下载地址 http://www.keepass.info/download.html，下载安装包解压后双击安装文件根据安装向导就可以安装。

选择 I accept 意思也就是我同意，然后 next（下一步），根据提示继续 next 就可以安装成功。软件是英文的，安装好之后我们看到界面仍旧都是英文的，我们可以去 http://keepass.info/translations.html，中下载简体中文插件复制放到安装目录下，运行软件“KeePass”，点击“View”菜单中“Change Language”命令，选择“简体中文”重新启动程序即可。

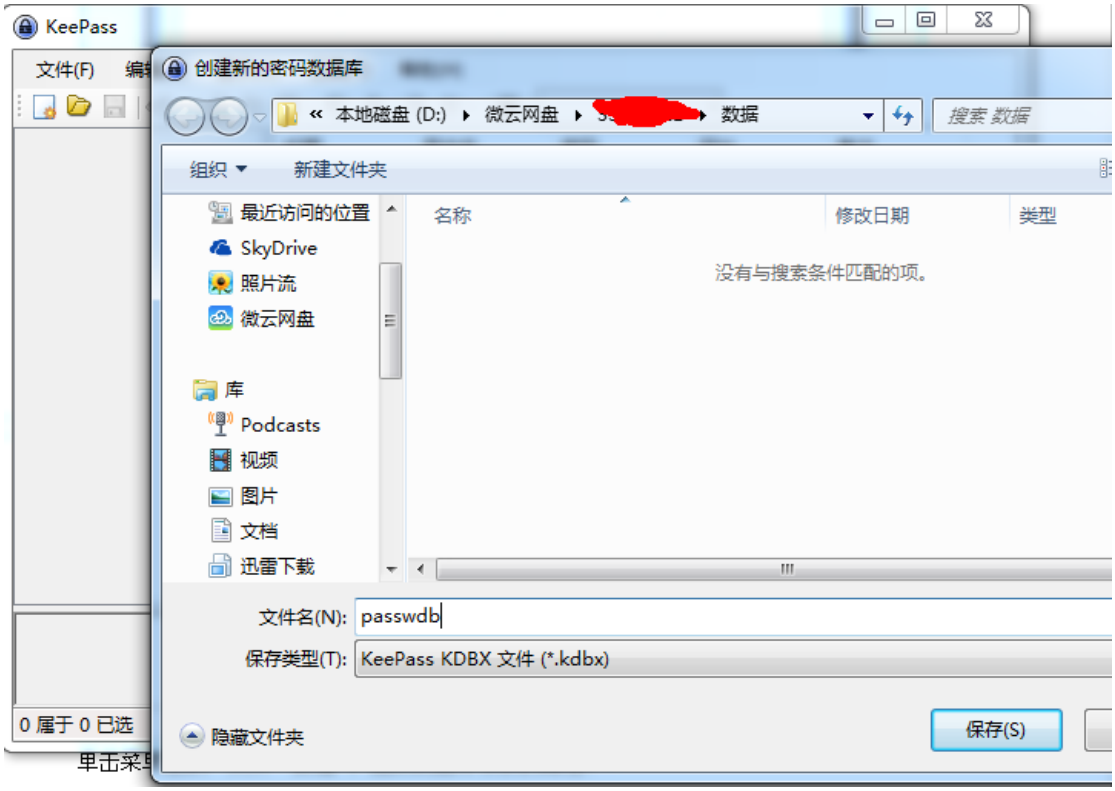


- 使用软件

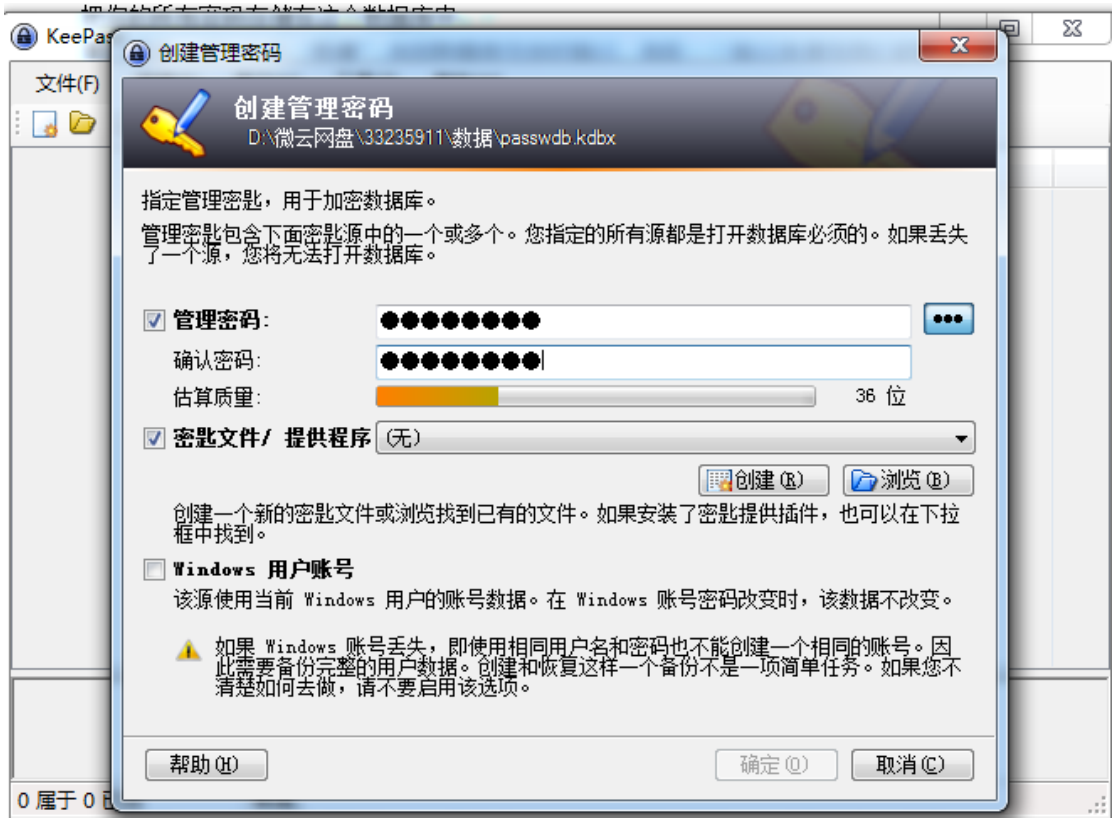
经过上面的步骤，我们返回到桌面双击 keepass 的快捷键，就可以看到软件的语言已经变成中文的了。

如果是第一次使用，我们首先的是要创建数据库，也就是存放密码的数据库，KeePass 将会把你的所有密码存储在这个数据库中。

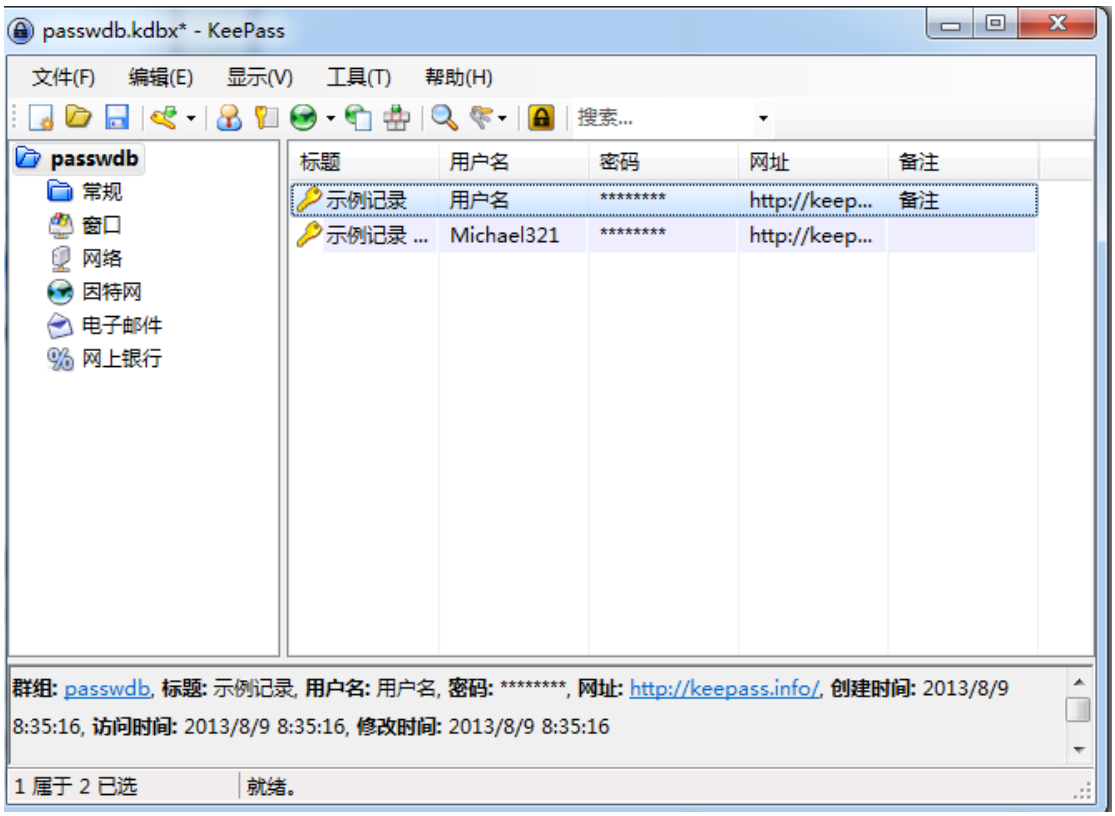
单击菜单上的“文件—新建”，选择数据库存放的路径，选择一个路径来保存我们的数据库，然后点击保存：



这里需要填入数据库管理密码，并要确认密码，我们同时要勾选上密匙文件，并选择密钥的保存位置（可以选择不用密钥）。这样将来在开启数据库时就要“主密码”和“密钥文件”同时具备才行。输入完毕后点击“确定”然后点击创建

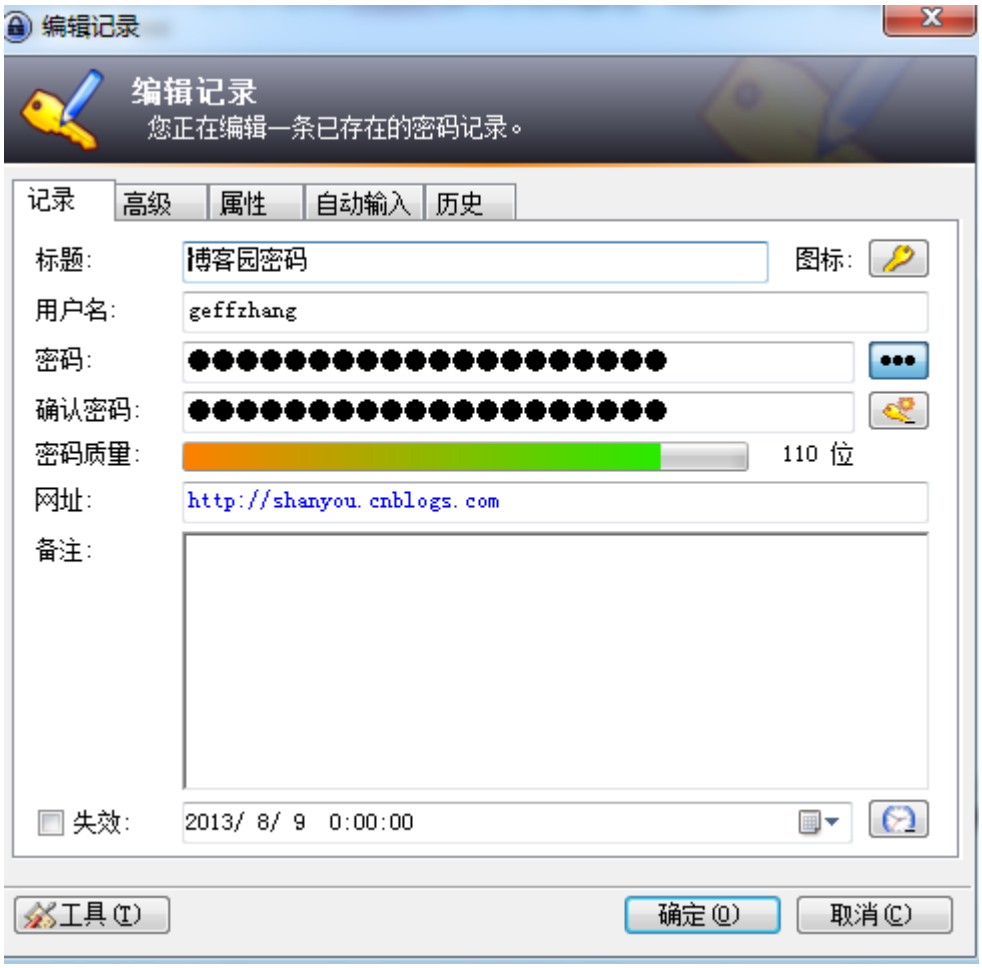


进入数据库配置参数，在这里描述或者不描述都可以，点击确定后就进入主窗口并可以看到一条示例记录，点击示例记录在下方可以看到网址和密码并且也创建时间。



主窗口的左边是密码群组，各个群组又可建立子群组。右边是你的密码记录。密码记录收纳于不同的密码群组中。你可以使用 Keepass 默认的密码群组，或删除它们，创建自己的密码组。

在主窗口右边单击右键，选择“添加记录…”，就可以编辑你的记录了：记录标题，用户名，网址，密码，备注等。可以空着不填。确认后就完成了一条记录了，在主窗口右边的子窗口中看到刚才新增的密码条目。
当你再次使用这个文件时，选中该条记录，在记录上右击鼠标就可以真正使用它了。你可以将用户名拷贝后，就可以在其它的任何软件中粘贴了，还可以将用户名拖放到其它窗口中。最后，一定要记住保存密码，点击“文件->保存”，或工具栏上的保存按钮。

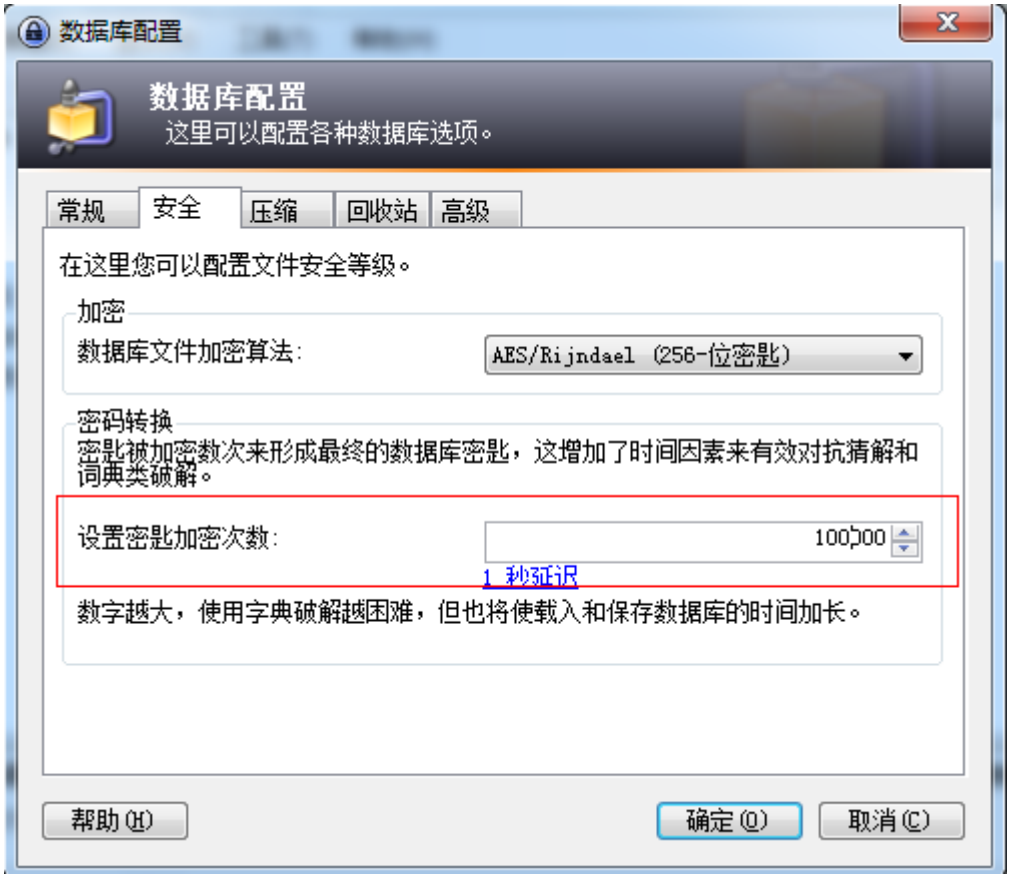


标题自定义，方便我们以后查找；用户名 用来记录密码的用户是谁；密码这个默认就存在了，也可以更改，点一下后面的 "..." 图标可以查看密码的内容，再点一下变为不可见状态；网址用来记录网址，方便我们跳转，比如这个密码为某个网站的某个会员的密码，那如果在这里填写了该网址地址，则可以直接跳转到那个网站，可以留空；Notes 用来写一些与这个密码相关的信息，方便我们记忆，可以留空。

提示：随着数据库里的记录数目的逐步增多，通过“编辑”菜单中的“在数据库中查找”或“在此群组内查找”命令可以迅速的找到要查的信息。另外，我们可以把数据库导出为 TXT、HTML、XML、CSV 等格式，导入 CSV、TXT、KeePass 数据库等格式的文件。

- 数据库的配置

点击“文件->数据库设置…”，在弹出的窗口的选择“安全”按钮创建一个加密算法，并设置密钥的加密次数（如果选择生成密钥的话）。通常加密次数的数量级达到 10 万，也就足够了。



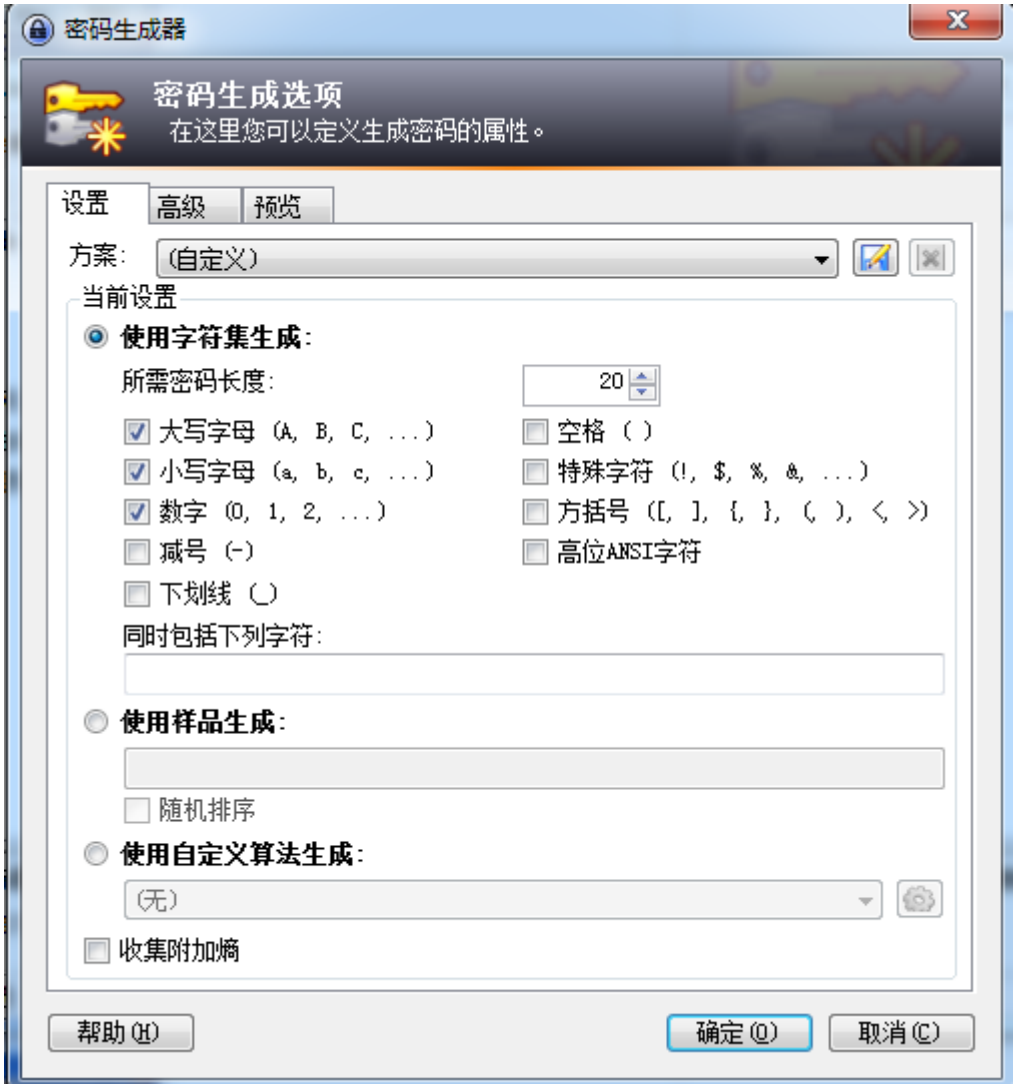
- Keepass 数据的备份

备份你所保存的密码数据库文件（*.kdb）。如果你生在了密钥，还要保存密钥文件。以上文件打包后，可以放入多动存储装置中，或放入网络空间，现在网盘很多，可以把文件之间放到网盘上，推荐用微云。

提示：如果用户的密码记录经常变动，那么相应的 KeePass 数据也要经常备份。保存好你的数据。尽管你有数不清的密码，但只需看好一个 KeePass 数据备份就可以了。

- 密码生成器

在添加记录时，我们可以利用密码生成器来生成复杂的密码，点击“密码”右侧的“生成”按钮，调出“密码生成器”属性框，然后，点击“确定”按钮即可自动生成随机密码。通过点击“***”按钮将密码明文显示，清楚的看到由生成器所产生的字符串。



我们现在可以把众多的 Linux 服务器密码管理工作了，而且还可以把生活中各个方面都需要用到密码，网站，邮箱，论坛，银行卡等等，使用 keepass 可以方便的为我们服务记录各种不同的密码，使我们的操作更加省时高效，管理更方便安全。

第六章 Linux 磁盘管理

在日常的 Linux 管理工作中,磁盘管理的工作还是很多的，包括查看状态，分区，格式化磁盘，挂载/卸载磁盘，对文件系统进行扫描等等。

获取硬盘的属性信息

硬盘上有些什么、文件们都有多大，在有些时候我们是需要关心的，所以这里简单介绍两个命令：df、du。

命令 ：df

df 用来检查文件系统的磁盘空间占用情况。可以利用该命令来获取硬盘被占用了多少空间，目前还剩下多少空间等信息。
df 的参数有：

- -a 显示所有文件系统的磁盘使用情况，包括 0 块（block）的文件系统，如/proc 文件系统。
- -k 以 k 字节为单位显示。
- -i 显示 i 节点信息，而不是磁盘块。
- -t 显示各指定类型的文件系统的磁盘空间使用情况。
- -x 列出不是某一指定类型文件系统的磁盘空间使用情况（与 t 选项相反）。
- -T 显示文件系统类型。

```
[azureuser@mono ~]$ sudo df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda1             29954008   5015480  23416916  18% /
tmpfs                   794380        16    794364   1% /dev/shm
/dev/sdb1             72246600    184084  68392604   1% /mnt/resource
[azureuser@mono ~]$
```

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，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]
它的参数如下：

- -s 对每个 Names 参数只给出占用的数据块总数。
- -a 递归地显示指定目录中各文件及子孙目录中各文件占用的数据块数。若既不指定-s，也不指定-a，则只显示 Names 中的每一个目录及其中的各子目录所占的磁盘块数。
- -b 以字节为单位列出磁盘空间使用情况（系统缺省以 k 字节为单位）。
- -k 以 1024 字节为单位列出磁盘空间使用情况。
- -c 最后再加上一个总计（系统缺省设置）。
- -l 计算所有的文件大小，对硬链接文件，则计算多次。
- -x 跳过在不同文件系统上的目录不予统计。
- -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。

通常习惯用 du -sh filename 这样的形式：

```
[azureuser@mono ~]$ du -sh
8.1M    .
[azureuser@mono ~]$ du -h
2.0M    ./RxDemo/lib
```

2.1M	./RxDemo
3.7M	./QuartzDemo
1.1M	./RazorDemo
4.0K	./.mono/registry/CurrentUser
8.0K	./.mono/registry
12K	./.mono
64K	./wcf/TestService
44K	./wcf/TestClient
112K	./wcf
1.1M	./BddifySamples
80K	./NPinyin
8.1M	.

磁盘的分区

首先谈谈基础知识，一个硬盘可以划分 4 个区，3 个主要分区，一个扩展分区，而扩展分区里可以划分 n 个逻辑分区，扩展分区本身不能储存任何东西，也不能格式化成某种文件系统，只能用于区分逻辑分区。
fdisk 是 Linux 下硬盘的分区工具，是一个非常实用的命令。
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。
[azureuser@mono ~]$ sudo fdisk -l

Disk /dev/sda: 32.2 GB, 32212254720 bytes
255 heads, 63 sectors/track, 3916 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: 0x00072ca8

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1	*	1	3789	30432256	83	Linux
/dev/sda2		3789	3917	1024000	82	Linux swap / Solaris

Disk /dev/sdb: 75.2 GB, 75161927680 bytes
255 heads, 63 sectors/track, 9137 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: 0xf2e41048

Device	Boot	Start	End	Blocks	Id	System
/dev/sdb1	*	1	9138	73398272	83	Linux

磁盘格式化

分完区还不算完事，格式化后才能正常使用。这里介绍下 linux 的格式化命令：

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’ 分区时设置每个数据区块占用空间大小，目前支持 1024, 3576 以及 4376 bytes 每个块。
‘-i’ 设置 inode 的大小
‘-N’ 设置 inode 数量，有时使用默认的 inode 数不够用，所以要自定设置 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’ 默认该分区的标签 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.

挂载文件系统

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。

挂载文件系统有两个方法，一种是利用 mount 命令，另外一种是修改/etc/fstab，因为修改 /etc/fstab 并不方便，所以我只介绍 mount。

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

```
[azureuser@mono ~]$ mount
/dev/sda1 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,rootcontext="system_u:object_r:tmpfs_t:s0")
```

none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
/dev/sdb1 on /mnt/resource type ext4 (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。

对文件系统进行扫描

各位朋友对 Windows 中的 scandisk 不寞生吧，在 Linux 中就有类似这样的工具 fsck，不过 fsck 可不仅仅是扫描，还能修正文件系统的一 些问题。一定要注意的是 fsck 扫描文件系统时一定要在修复模式或把设备 umount 后进行，否则会有重大的问题发生。

创建 SWAP 文件

Linux 支持虚拟内存（virtual memory），虚拟内存是指使用磁盘当作 RAM 的扩展，这样可用的内存的大小就相应地增大了。内核会将暂时不用的内存块的内容写到硬盘上，这样一来，这块内存就可用于其他目的。当需要用到原始的内容时，他们被重新读入内存。这些操作对用户来说是完全透明的；Linux 下运行的程式只是看到有大量的内存可供使用而并没有注意到时不时他们的一部分是驻留在硬盘上的。当然，读写硬盘要比直接使用真实内存慢得多（要慢数千倍），所以程式就不会象一直在内存中运行的那样快。用作虚拟内存的硬盘部分被称为交换空间（Swap Space）。
Linux 系统常常动不动就使用交换空间，以保持尽可能多的空闲物理内存。即使并没有什么事情需要内存，Linux 也会交换出暂时不用的内存页面。这能避免等待交换所需的时间：当磁盘闲着，就能提前做好交换。
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。
[azureuser@mono ~]$ dd if=/dev/zero of=/tmp/newdisk bs=4k count=102400
102400+0 records in
102400+0 records out
419430400 bytes (419 MB) copied, 8.29287 s, 50.6 MB/s
“dd”这个命令经常用到，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. 用 dd 建了一个大小为 400M 的文件，然后格式化成 swap 格式：
[azureuser@mono ~]$ mkswap -f /tmp/newdisk
Setting up swapspace version 1, size = 409596 KiB

no label, UUID=c334b9c2-f9bc-4ef9-8168-d60d836d3180

[azureuser@mono ~]$ free -m

	total	used	free	shared	buffers	cached
Mem:	1551	1168	383	0	170	694
-/+ buffers/cache:		303	1248			
Swap:	0	0	0			

格式化完后，就可以挂载上使用了：

[azureuser@mono ~]$ sudo swapon /tmp/newdisk

[sudo] password for azureuser:

[azureuser@mono ~]$ free -m

	total	used	free	shared	buffers	cached
Mem:	1551	1168	382	0	170	694
-/+ buffers/cache:		303	1247			
Swap:	399	0	399			

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。

第七章文档的压缩与打包

本章介绍 Linux 环境中处理 rar、zip、gz、bz2、tar 等常见压缩/归档文件的方法。在 Linux 中，有很多的压缩命令。利用这些压缩命令，可以方便的从网络上下载大型的文件。同时，我们知道，Linux 文件的扩展名是没有特殊意义的，不过，因为 Linux 下存在着许多压缩命令，所以为了方便记忆，会对这些命令所压缩的文件进行一些特殊的命名方式。

rar 解压缩

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，如果只是解压，可安装 unrar。unrar 是 RARLAB 公开了源码的工具：

wget http://pkgs.repoforge.org/unrar/unrar-4.2.3-1.el6.rf.x86_64.rpm

[azureuser@mono src]$ rpm -i unrar-4.2.3-1.el6.rf.x86_64.rpm

用法: unrar <命令> -<开关 1> -<开关 N> <压缩文件> <文件...>
<@列表文件...> <解压路径\>

<注释>

e	解压压缩文件到当前目录
l[t,b]	列出压缩文件[技术信息,简洁]
p	打印文件到标准输出设备
t	测试压缩文件

v[t,b] 详细列出压缩文件[技术信息,简洁]
x 用绝对路径解压文件

<开关>
- 停止扫描
ac 压缩或解压后清除存档属性
ad 添加压缩文件名到目标路径
ap<路径> 添加路径到压缩文件中
av- 禁用用户身份校验
c- 禁用注释显示
cfg- 禁用读取配置
cl 名称转换为小写
cu 名称转换为大写
dh 打开共享文件
ep 从名称中排除路径
ep3 扩展路径为包含盘符的完全路径
f 刷新文件
id[c,d,p,q] 禁止信息显示
ierr 发送所有消息到标准错误设备
inul 禁用所有消息
ioff 完成一个操作后关闭 PC 电源
kb 保留损坏的已解压文件
n<文件> 仅包括指定文件
n@ 从标准输入设备读取要包括的文件名称
n@<列表> 在指定列表文件包括文件
o+ 覆盖已存在文件
o- 不覆盖已存在文件
oc 设置 NTFS 压缩属性
or 自动重命名文件
ow 保存或恢复文件所有权和组
[密码] 设置密码
p- 不询问密码
r 包含子目录
ri<P>[:<S>] 设置优先级 (0-默认,1-最小..15-最大) 和休眠时间(毫秒)
sl<大小> 处理小于指定大小的文件
sm<大小> 处理大于指定大小的文件
ta<日期> 添加日期 <日期> 后修改的文件,日期格式 YYYYMMDDHHMMSS
tb<日期> 添加日期 <日期> 前修改的文件,日期格式 YYYYMMDDHHMMSS
tn<时间> 添加 <时间> 以后的文件
to<时间> 添加 <时间> 以前的文件
ts<m,c,a>[N] 保存或恢复文件时间(修改,创建,访问)
u 更新文件
v 列出所有卷
ver[n] 文件版本控制

vp 每卷之前暂停
x<文件> 排除指定的文件
x@ 从标准输入设备读取要排除的文件名
x@<列表> 排除指定列表文件中的文件
y 假设对全部询问都回答是

如果需要制作 rar 文件，可以安装 rar 软件包并购买相应许可。rar 也具备 unrar 功能，用法同上，只需把 unrar 换成 rar 即可。

Zip

info-zip 提供了自由的 zip 文件处理方案，一般可以从系统中安装 zip 或 unzip：

值得一提的是，如果 zip 中包括非 UTF-8 编码的中文文件名文件，解压结果中会出现乱码，解决该乱码问题请参考常见故障诊断与排除的文件名编码转换。

同样由于潜在的文件名编码问题，不建议在 Linux 中打 zip 包供 Windows 用户使用。

zip 命令详解
功能说明：压缩文件。
语 法：zip [-AcdDfFghjJKlLmoqrSTuvVwXyz$][-b <工作目录>][-ll][-n <字尾字符串>][-t <日期时间>][-<压缩效率>][压缩文件][文件...][-i <范本样式>][-x <范本样式>]
补充说明：zip 是个使用广泛的压缩程序，文件经它压缩后会另外产生具有".zip"扩展名的压缩文件。
参 数：
-A 调整可执行的自动解压缩文件。
-b<工作目录> 指定暂时存放文件的目录。
-c 替每个被压缩的文件加上注释。
-d 从压缩文件内删除指定的文件。
-D 压缩文件内不建立目录名称。
-f 此参数的效果和指定"-u"参数类似，但不仅更新既有文件，如果某些文件原本不存在于压缩文件内，使用本参数会一并将其加入压缩文件中。
-F 尝试修复已损坏的压缩文件。
-g 将文件压缩后附加在既有的压缩文件之后，而非另行建立新的压缩文件。
-h 在线帮助。
-i<范本样式> 只压缩符合条件的文件。
-j 只保存文件名称及其内容，而不存放任何目录名称。
-J 删除压缩文件前面不必要的数据。
-k 使用 MS-DOS 兼容格式的文件名称。
-l 压缩文件时，把 LF 字符置换成 LF+CR 字符。
-ll 压缩文件时，把 LF+CR 字符置换成 LF 字符。
-L 显示版权信息。
-m 将文件压缩并加入压缩文件后，删除原始文件，即把文件移到压缩文件中。
-n<字尾字符串> 不压缩具有特定字尾字符串的文件。
-o 以压缩文件内拥有最新更改时间的文件为准，将压缩文件的更改时间设成和该文件相同。
-q 不显示指令执行过程。

-r 递归处理，将指定目录下的所有文件和子目录一并处理。
-S 包含系统和隐藏文件。
-t<日期时间> 把压缩文件的日期设成指定的日期。
-T 检查备份文件内的每个文件是否正确无误。
-u 更换较新的文件到压缩文件内。
-v 显示指令执行过程或显示版本信息。
-V 保存 VMS 操作系统的文件属性。
-w 在文件名称里假如版本编号，本参数仅在 VMS 操作系统下有效。
-x<范本样式> 压缩时排除符合条件的文件。
-X 不保存额外的文件属性。
-y 直接保存符号连接，而非该连接所指向的文件，本参数仅在 UNIX 之类的系统下有效。
-z 替压缩文件加上注释。
-$ 保存第一个被压缩文件所在磁盘的卷册名称。
-<压缩效率> 压缩效率是一个介于 1-9 的数值。

unzip 命令详解

功能说明：解压缩 zip 文件

语 法：unzip [-cflptuvz][-agCjLMnoqsVX][-P <密码>][.zip 文件][文件][-d <目录>][-x <文件>] 或 unzip [-Z]
补充说明：unzip 为.zip 压缩文件的解压缩程序。
参 数：
-c 将解压缩的结果显示到屏幕上，并对字符做适当的转换。
-f 更新现有的文件。
-l 显示压缩文件内所包含的文件。
-p 与-c 参数类似，会将解压缩的结果显示到屏幕上，但不会执行任何的转换。
-t 检查压缩文件是否正确。
-u 与-f 参数类似，但是除了更新现有的文件外，也会将压缩文件中的其他文件解压缩到目录中。
-v 执行是时显示详细的信息。
-z 仅显示压缩文件的备注文字。
-a 对文本文件进行必要的字符转换。
-b 不要对文本文件进行字符转换。
-C 压缩文件中的文件名称区分大小写。
-j 不处理压缩文件中原有的目录路径。
-L 将压缩文件中的全部文件名改为小写。
-M 将输出结果送到 more 程序处理。
-n 解压缩时不要覆盖原有的文件。
-o 不必先询问用户，unzip 执行后覆盖原有文件。
-P<密码> 使用 zip 的密码选项。
-q 执行时不显示任何信息。
-s 将文件名中的空白字符转换为底线字符。

-V 保留 VMS 的文件版本信息。
-X 解压缩时同时回存文件原来的 UID/GID。
[.zip 文件] 指定.zip 压缩文件。
[文件] 指定要处理.zip 压缩文件中的哪些文件。
-d<目录> 指定文件解压缩后所要存储的目录。
-x<文件> 指定不要处理.zip 压缩文件中的哪些文件。
-Z unzip -Z 等于执行 zipinfo 指令

范例：
zip 命令可以用来将文件压缩成为常用的 zip 格式。unzip 命令则用来解压缩 zip 文件。
1. 我想把一个文件 abc.txt 和一个目录 dir1 压缩成为 yasuo.zip：
zip -r yasuo.zip abc.txt dir1

2.我下载了一个 yasuo.zip 文件，想解压缩：
unzip yasuo.zip

3.我当前目录下有 abc1.zip，abc2.zip 和 abc3.zip，我想一起解压缩它们：
unzip abc\?.zip

注释：?表示一个字符，如果用*表示任意多个字符。
4.我有一个很大的压缩文件 large.zip，我不想解压缩，只想看看它里面有什么：

unzip -v large.zip
5.我下载了一个压缩文件 large.zip，想验证一下这个压缩文件是否下载完全了

unzip -t large.zip
6.我用-v 选项发现 music.zip 压缩文件里面有很多目录和子目录，并且子目录中其实都是歌曲 mp3 文件，我想把这些文件都下载到第一级目录，而不是一层一层建目录：
unzip -j music.zip

bzip2 压缩工具

bzip2 的压缩效果通常不错，但速度要慢一些[3]，后缀为 bz2。

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：

[azureuser@mono test]$ ls

dirb log log2 test test.txt

[azureuser@mono test]$ bzip2 test.txt

[azureuser@mono test]$ ls

dirb log log2 test test.txt.bz2

[azureuser@mono test]$ bzip2 -d test.txt.bz2

[azureuser@mono test]$ ls

dirb log log2 test test.txt

[azureuser@mono test]$ bzip2 -z test.txt

[azureuser@mono test]$ ls

dirb log log2 test test.txt.bz2

bzip2 不可以压缩目录：

[azureuser@mono test]$ bzip2 test/

bzip2: Input file test/ is a directory.

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

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。

[azureuser@mono test]$ gzip test.txt

[azureuser@mono test]$ ls

dirb log log2 test test.txt.gz

[azureuser@mono test]$ gzip -d test.txt.gz

[azureuser@mono test]$ ls

dirb log log2 test test.txt

tar 压缩工具

tar 实际上是个打包工具，将一些文件存放在一个单独的文件中，并不做压缩。[2]而真正的压缩是其与 gzip 或 bzip2 等工具配合完成的，故常见的后缀名为.tar.gz, .tar.bz2。

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

“-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）

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。如需在 Linux 中创建压缩文件，推荐使用自由的 tar 与 gzip 组合，生成的文件同样兼容 Windows 中的常见压缩软件。例：

[azureuser@mono tmp]$ tar cvfz test.tar.gz test
test/
test/log/
test/log/222
test/log/111
test/log2/
test/dirb/
test/dirb/dirc/

```
test/dirb/hellomono
test/test/
test/test.txt
[azureuser@mono tmp]$ ls
newdisk  nunit20  root-temp-aspnet-0  state.pid  test  test.tar.gz
```

通常用“-zxvf” 用来解压.tar.gz 的压缩包

```
[azureuser@mono tmp]$ rm -rf test
[azureuser@mono tmp]$ ls
newdisk  nunit20  root-temp-aspnet-0  state.pid  test.tar.gz
[azureuser@mono tmp]$ tar -zxvf test.tar.gz
test/
test/log/
test/log/222
test/log/111
test/log2/
test/dirb/
test/dirb/dirc/
test/dirb/hellomono
test/test/
test/test.txt
[azureuser@mono tmp]$ ls
newdisk  nunit20  root-temp-aspnet-0  state.pid  test  test.tar.gz
```

第八章 安装 RPM 包或者安装源码包

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

在 Linux 系统下，对于软件包的管理有多种机制，有源代码方式、RPM 软件包管理方式以及 YUM 软件管理方式。

源代码形式

在开源的环境下，大多数的开源软件都是以源代码的形式来发布，通常将源代码打包成 tar.gz 的归档压缩文件发布到网上供我们下载使用。但是我们下载下来的源代码方式我们还不能够直接使用，不像在 windows 系统上直接下载下来可执行的二进制文件，我们需要将下载好的源码编译成可执行的二进制文件才能运行使用。

在 linux 下面安装一个源码包是最常用的，我在日常的管理工作中，大部分软件都是通过源码安装的，比如本书中的 Mono 的安装都是采用源代码形式安装。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 文件中默认的参数进行编译，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。

基本上所有以源代码形式发布的软件都是按照以上流程来进行安装，大家可能觉得其实步骤就这三个，安装过程非常简单!其实不然，通常以源代码形式发布的软件，在对其进行①、②步操作时往往会出很多很多的问题，比如说编译过程中缺少所需的库文件，或者说编译源码时又需要依赖 A 文件，编译 A 文件时可能又需要用到 B 文件，这样我们需要花大量时间去找这些依赖的问题，同时如果一个软件特别大的话，我们在将其编译过程中要等待非常久的时间。所以说其实源代码形式的安装软件并不是我们想象的那么简单，但是为什么源代码形式的软件管理还一直存在呢？必然也有其优点，因为我们的源码都要通过编译成可执行的二进制文件才行，所以说它适用于各种操作系统平台，我只需要在各个操作系统平台上对其源码进行编译即可运行起来了。下面我们来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。

RPM 软件包管理

因为源代码方式安装软件终究还是比较麻烦，所以说现在出现了许多代替源码方式安装的软件管理机制。RPM 就是其中一个，RPM（redhat Package Manager）是为了方便软件管理使用所开发的一套开源软件格式。
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。
每一个 rpm 包的名称都由 - 和 . 分成了若干部分。就拿“convmv-1.15-1.el6.rf.noarch.rpm”这个包来解释一下，“convmv”为包名，“1.15”则为版本信息，“1.el6.rf”为发布版本号，“noarch”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，noarch，这代表这个 rpm 包没有硬件平台限制。下面介绍一下 rpm 常用的命

令。

- **安装一个 rpm 包**

[azureuser@mono src]$ sudo rpm -ivh convmv-1.15-1.el6.rf.noarch.rpm
warning: convmv-1.15-1.el6.rf.noarch.rpm: Header V3 DSA/SHA1 Signature, key ID 6b8d79e6: NOKEY
Preparing... ### [100%]
1:convmv ### [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

- **升级一个 rpm 包**

命令 rpm -Uvh filename
“-U” ：即升级的意思

- **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。

[azureuser@mono src]$ rpm -qa | grep convmv
convmv-1.15-1.el6.rf.noarch
[azureuser@mono src]$ sudo rpm -e convmv-1.15-1.el6.rf.noarch

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。

- **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)

[azureuser@mono src]$ sudo rpm -q convmv-1.15-1.el6.rf.noarch
package convmv-1.15-1.el6.rf.noarch is not installed
[azureuser@mono src]$ sudo rpm -ivh convmv-1.15-1.el6.rf.noarch.rpm
warning: convmv-1.15-1.el6.rf.noarch.rpm: Header V3 DSA/SHA1 Signature, key ID 6b8d79e6: NOKEY
Preparing... ### [100%]
1:convmv ### [100%]
[azureuser@mono src]$ rpm -q convmv-1.15-1.el6.rf.noarch
convmv-1.15-1.el6.rf.noarch

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

[azureuser@mono src]$ rpm -qa | head
gdb-7.2-56.el6.x86_64
at-3.1.10-43.el6_2.1.x86_64
ca-certificates-2010.63-3.el6_1.5.noarch
cvs-1.11.23-11.el6_2.1.x86_64
basesystem-10.0-4.el6.noarch
alsa-utils-1.0.22-3.el6.x86_64

logrotate-3.7.8-15.el6.x86_64
fontpackages-filesystem-1.41-1.1.el6.noarch
tcsh-6.17-19.el6_2.x86_64
ncurses-base-5.7-3.20090208.el6.x86_64
[azureuser@mono src]$

- **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）

[azureuser@mono src]$ rpm -qi convmv-1.15-1.el6.rf.noarch
Name : convmv Relocations: (not relocatable)
Version : 1.15 Vendor: Dag Apt Repository, http://dag.wieers.com/apt/
Release : 1.el6.rf Build Date: Mon 05 Sep 2011 04:37:50 PM UTC
Install Date: Sat 10 Aug 2013 04:11:13 AM UTC Build Host: lisse.hasselt.wieers.com
Group : Applications/File Source RPM: convmv-1.15-1.el6.rf.src.rpm
Size : 42999 License: GPL
Signature : DSA/SHA1, Mon 05 Sep 2011 04:52:11 PM UTC, Key ID a20e52146b8d79e6
Packager : Dries Verachtert <dries@ulyssis.org>
URL : https://www.j3e.de/linux/convmv/
Summary : Convert filenames to a different encoding
Description :
convmv converts filenames (not file content), directories, and even whole
filesystems to a different encoding. This comes in very handy if, for example,
one switches from an 8-bit locale to an UTF-8 locale or changes charsets on
Samba servers. It has some smart features: it automagically recognises if a
file is already UTF-8 encoded (thus partly converted filesystems can be fully
moved to UTF-8) and it also takes care of symlinks. Additionally, it is able
to convert from normalization form C (UTF-8 NFC) to NFD and vice-versa. This
is important for interoperability with Mac OS X, for example, which uses NFD,
while Linux and most other Unixes use NFC. Though it's primary written to
convert from/to UTF-8 it can also be used with almost any other charset
encoding. Convmv can also be used for case conversion from upper to lower case
and vice versa with virtually any charset. Note that this is a command line
tool which requires at least Perl version 5.8.0.

- **列出一个 rpm 包安装的文件**

命令 rpm -ql 包名

[azureuser@mono src]$ rpm -ql convmv-1.15-1.el6.rf.noarch
/usr/bin/convmv
/usr/share/doc/convmv-1.15
/usr/share/doc/convmv-1.15/CREDITS
/usr/share/doc/convmv-1.15/Changes
/usr/share/doc/convmv-1.15/TODO
/usr/share/man/man1/convmv.1.gz

通过上面的命令可以看出文件“/usr/bin/convmv”是通过安装“convmv-1.15-1.el6.rf.noarch”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？

- **列出某一个文件属于哪个 rpm 包**

命令 rpm -qf 文件的绝对路径

```
[azureuser@mono src]$ rpm -qf /usr/bin/convmv
convmv-1.15-1.el6.rf.noarch
```

- **RPM 软件验证命令**

rpm -K software.rpm 验证 rpm 文件

rpm -V softname 验证已安装的软件

RPM 软件包管理还具有验证功能，因为在开源的软件里，源代码都是开放的，我们从网上下载的软件可以被一些不法分子在里面植入了一些木马程序，这样就会损害我们的操作系统。所以为了安全起见现代操作系统都加入了对软件的验证功能。

验证 rpm 文件我们可以使用 rpm -K software.rpm 命令，例如我们要验证刚才的 convmv-1.15-1.el6.rf.noarch.rpm：

```
[azureuser@mono src]$ sudo rpm -K convmv-1.15-1.el6.rf.noarch.rpm
[sudo] password for azureuser:
convmv-1.15-1.el6.rf.noarch.rpm: (SHA1) DSA sha1 md5 (GPG) NOT OK (MISSING KEYS: GPG#6b8d79e6)
```

验证以后发现该软件是没有问题的。

验证已安装的软件我们可以使用 rpm -V softname 命令，例如我要验证一下安装的 convmv 软件，就可以使用如下命令：

```
[azureuser@mono src]$ rpm -V convmv
```

如果没有出现任何错误，就表示该软件是完整的，没有被修改。我们使用的 RHEL 以及 CentOS 等 Linux 系统，其软件包的安装维护都是通过 RPM 软件包来进行管理的，我们也看到使用 RPM 软件包来对软件进行管理非常的方便。

yum 工具

在前面的章节中，多次提到 yum 工具。yum（全 称为 Yellow dog Updater, Modified）是一个在 Fedora 和 RedHat 以及 SUSE 中的 Shell 前端软件包管理器。基於 RPM 包管理，能够从指定的服务器自动下载 RPM 包并且安装，可以自动处理依赖性关系，并且一次安装所有依赖的软体包，无须繁琐地一次次下载、安装。yum 提供了查找、安装、删除某一个、一组甚至 全部软件包的命令，而且命令简洁而又好记。

下面介绍常用的 yum 命令：

- 列出所有可用的 rpm 包 “yum list”

```
[azureuser@mono src]$ yum list | head -n 10
Loaded plugins: security
Installed Packages
ConsoleKit.x86_64                                    0.4.1-3.el6
@anaconda-CentOS-201207061011.x86_64/6.3
    ConsoleKit-libs.x86_64                           0.4.1-3.el6
@anaconda-CentOS-201207061011.x86_64/6.3
    MAKEDEV.x86_64                                   3.24-6.el6
@anaconda-CentOS-201207061011.x86_64/6.3
```

SDL.x86_64 1.2.14-3.el6
@anaconda-CentOS-201207061011.x86_64/6.3
WALinuxAgent.noarch 1.3.3-1
@openlogic
abrt.x86_64 2.0.8-6.el6.centos.2
@updates
abrt-addon-ccpp.x86_64 2.0.8-6.el6.centos.2 @updates
abrt-addon-kerneloops.x86_64 2.0.8-6.el6.centos.2 @updates

限于篇幅，只列举出来前 8 个包信息。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”，“@openlogic” 或者 “@updates”，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)

- 搜索一个 rpm 包

命令 yum search [相关关键词]

```
[azureuser@mono src]$ yum search vim
Loaded plugins: security
============================== N/S Matched: vim ======================
vim-X11.x86_64 : The VIM version of the vi editor for the X Window Syste
vim-common.x86_64 : The common files needed by any version of the VIM ed
vim-enhanced.x86_64 : A version of the VIM editor which includes recent
                    : enhancements
vim-minimal.x86_64 : A minimal version of the VIM editor

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

除了这样搜索外，常用的是利用 grep 来过滤

```
[azureuser@mono src]$ yum search vim
Loaded plugins: security
============================== N/S Matched: vim ======================
vim-X11.x86_64 : The VIM version of the vi editor for the X Window Syste
vim-common.x86_64 : The common files needed by any version of the VIM ed
vim-enhanced.x86_64 : A version of the VIM editor which includes recent
                    : enhancements
vim-minimal.x86_64 : A minimal version of the VIM editor

  Name and summary matches only, use "search all" for everything.
[azureuser@mono src]$ yum list | grep vim
vim-common.x86_64 2:7.2.411-1.8.el6
@anaconda-CentOS-201207061011.x86_64/6.3
vim-enhanced.x86_64 2:7.2.411-1.8.el6
@anaconda-CentOS-201207061011.x86_64/6.3
vim-minimal.x86_64 2:7.2.411-1.8.el6
```

@anaconda-CentOS-201207061011.x86_64/6.3
vim-X11.x86_64 2:7.2.411-1.8.el6 base
我们同样可以找到相应的 rpm 包。

- 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。如果嫌这样太麻烦，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。

```
[azureuser@mono src]$ sudo yum install vim-X11
[sudo] password for azureuser:
Loaded plugins: security
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package vim-X11.x86_64 2:7.2.411-1.8.el6 will be installed
--> Finished Dependency Resolution
Dependencies Resolved
================================================================================
 Package          Arch          Version                          Repository   Size
================================================================================
Installing:
 vim-X11          x86_64        2:7.2.411-1.8.el6                base         1.0 M

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

Total download size: 1.0 M
Installed size: 2.2 M
Is this ok [y/N]: y
Downloading Packages:
vim-X11-7.2.411-1.8.el6.x86_64.rpm                       | 1.0 MB     00:00
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
Warning: RPMDB altered outside of yum.
  Installing : 2:vim-X11-7.2.411-1.8.el6.x86_64                             1/1
  Verifying  : 2:vim-X11-7.2.411-1.8.el6.x86_64                             1/1
Installed:
  vim-X11.x86_64 2:7.2.411-1.8.el6
Complete!
```

- 卸载一个 rpm 包

命令 yum remove [-y] [rpm 包名]

```
[azureuser@mono src]$ sudo yum remove -y vim-X11
Loaded plugins: security
Setting up Remove Process
Resolving Dependencies
--> Running transaction check
---> Package vim-X11.x86_64 2:7.2.411-1.8.el6 will be erased
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package          Arch          Version                        Repository    Size
================================================================================
Removing:
 vim-X11          x86_64        2:7.2.411-1.8.el6              @base         2.2 M

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

Installed size: 2.2 M
Downloading Packages:
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Erasing    : 2:vim-X11-7.2.411-1.8.el6.x86_64                              1/1
  Verifying  : 2:vim-X11-7.2.411-1.8.el6.x86_64                              1/1
Removed:
  vim-X11.x86_64 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。

- 升级一个 rpm 包

命令 yum update [-y] [rpm 包]

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，这里不再列举出来，如果您感兴趣就去 man 一下吧。除此之外，下面还会教您一些关于 yum 的小应用。

- 自动选择最快的源

由于 yum 中有的 mirror 速度是非常慢的，如果 yum 选择了这个 mirror，这个时候 yum 就会非常慢，对此，可以下载 fastestmirror 插件，它会自动选择最快的 mirror：

yum install yum-fastestmirror
配置文件：/etc/yum/pluginconf.d/fastestmirror.conf（一般不用改动），你的 yum 镜像的速度测试记录文件：/var/cache/yum/timedhosts.txt

- 配置 yum 源

yum 源的配置文件：/etc/yum.repos.d/CentOS-Base.repo，例如下面我们把源配置为网易的：

首先备份/etc/yum.repos.d/CentOS-Base.repo
mv /etc/yum.repos.d/CentOS-Base.repo /etc/yum.repos.d/CentOS-Base.repo.backup 下载 Centos 6 repo 文件 http://mirrors.163.com/.help/CentOS6-Base-163.repo, 放入/etc/yum.repos.d/(操作前请做好相应备份)，然后替换 CentOS-Base.repo, 运行 yum makecache 生成缓存。

- 添加特定软件的 Yum 源，例如在 yum 仓库中找不到 MonogoDB，MonogoDB 提供了 yum 源安装方法。可以通过下面的方式操作。

在/etc/yum.repos.d/目录中增加 *.repo yum 源配置文件，例如 10gen.repo
vi /etc/yum.repos.d/10gen.repo，输入下面的语句：

```
[10gen]
name=10gen Repository
baseurl=http://downloads-distro.mongodb.org/repo/redhat/os/x86_64
gpgcheck=0
```

做好 yum 源的配置后，如果配置正确执行下面的命令便可以查询 MongoDB 相关的信息：
查看 mongoDB 的服务器包的信息

```
[azureuser@mono src]$ yum info mongo-10gen-server
Loaded plugins: security
10gen                                                    |  951 B     00:00
10gen/primary                                            |  11 kB     00:00
10gen                                                                   9
Available Packages
Name        : mongo-10gen-server
Arch        : x86_64
Version     : 2.4.5
Release     : mongodb_1
Size        : 12 M
Repo        : 10gen
Summary     : mongo server, sharding server, and support scripts
URL         : http://www.mongodb.org
License     : AGPL 3.0
Description : Mongo (from "huMONGOus") is a schema-free
            : document-oriented database.
            :
            : This package provides the mongo server software, mong
            : sharding server softwware, default configuration file
            : and init.d scripts.
```

利用 yum 工具下载一个 rpm 包

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

a）首选要安装 yum-downloadonly

[azureuser@mono src]$ sudo yum install -y yum-plugin-downloadonly.noarch

b）下载一个 rpm 包而不安装，并且下载到指定的目录

[azureuser@mono src]$ sudo yum install -y yum-presto.noarch --downloadonly --downloaddir=/usr/local/src

Loaded plugins: downloadonly, security

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.x86_64 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	x86_64	3.5-0.5.20090913git.el6	base	71 k

Transaction Summary

Install 2 Package(s)

Total download size: 102 k

Installed size: 252 k

Downloading Packages:

```
(1/2): deltarpm-3.5-0.5.20090913git.el6.x86_64.rpm                    |  71 kB     00:00
(2/2): yum-presto-0.6.2-1.el6.noarch.rpm                              |  32 kB     00:00
--------------------------------------------------------------------------------
Total                                                      194 kB/s | 102 kB
00:00


exiting because --downloadonly specified

[azureuser@mono src]$ ls /usr/local/src
convmv-1.15-1.el6.rf.noarch.rpm                 libgdiplus-2.10.tar.bz2
deltarpm-3.5-0.5.20090913git.el6.x86_64.rpm     mono
jexus-5.4                                       mono32
jexus-5.4.tar.gz                                unrar-4.2.3-1.el6.rf.x86_64.rpm
libgdiplus-2.10                                 yum-presto-0.6.2-1.el6.noarch.rpm
```

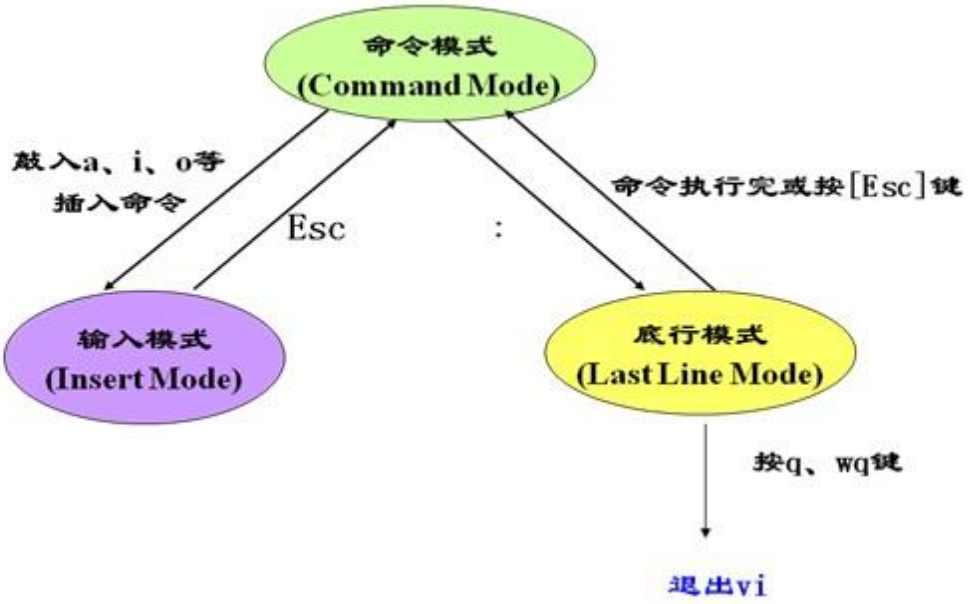
第九章 文本编辑工具 vim

Vim 是从 vi 发展出来的一个文本编辑器。代码补完、编译及错误跳转等方便编程的功能特别丰富，在程序员中被广泛使用。和 Emacs 并列成为类 Unix 系统用户最喜欢的编辑器。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。

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。

基本上 Vim 共分为 3 种模式，分别是一般模式，编辑模式和命令行模式，这三种模式的作用分别如下简述：

- 一般模式：默认模式。打开 vim 直接进入的是一般模式，在这个模式下，可以进行的操作有：移动光标，复制，粘贴，删除。
- 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
- 命令行模式：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。



Vim 下的常用命令

1、命令模式和编辑模式

启动 VIM 之后，就进行了命令模式。在命令模式下按“i”键进入编辑模式，在编辑模式下按“esc”键退回命令模式。

2、移动光标与删除单个字符

移动光标，方向键、home、end、PageUp、PageDown 等几个键就够用了，而退格键、Delete 键可以删除光标前、后的字符。

3、定义块

在命令模式下按一下“v”键开始定义块，再移动光标到块的结束位置一个块就定好了。要取消定义，行，连按两下“esc”键或者按一下“i”键进入编辑模式那个块的定义就被取消了。

4、复制、粘贴、块删除

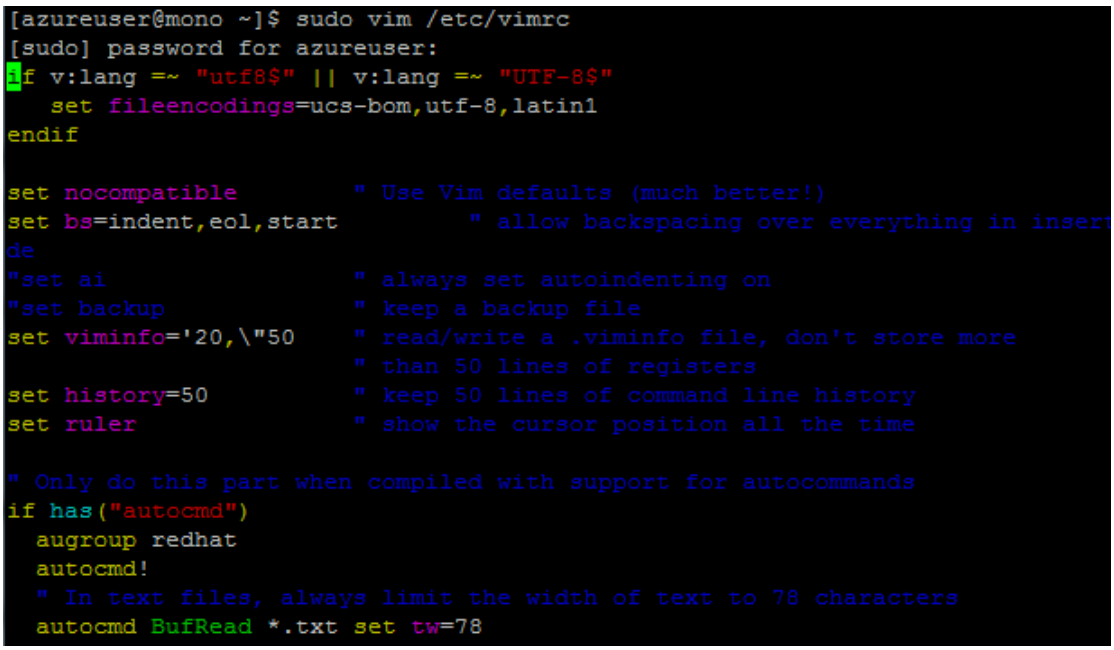
在命令模式下定好块，按“d”键，这个块的内容就会拜拜，按“y”键复制，按“p”键粘贴。

5、保存、退出

在底行模式下按 q 退出 q！强调退出

Vim 环境设置

在目录 /etc/ 下面，有个名为 vimrc 的文件，这是系统中公共的 vim 配置文件，对所有用户都有效。而在每个用户的主目录下，都可以自己建立私有的配置文件，命名 为：“.vimrc”



vim 环境变量的设置实在太多了，没必要全都自行配置。文件内容很少，只有两行：

set mouse=a

set backupdir=~/tmp/backup

第一行的作用是使得 vim 在编辑时支持鼠标操作，如选择等；第二行的作用是把 vim 生成的备份文件（比原文件名多一个”~"）全都放置在主目录下的 tmp/backup 文件夹下，这样就不会每次都看到烦人的一堆”～“文件了，而且如果要清理也方便许多。

在用户的主目录下还有一个文件信息文件 .viminfo 被设计为储存状态信息:

- 命令行和模式搜索的历史记录
- 寄存器内文本
- 各种文件的标记
- 缓存器列表
- 全局变量

你每次退出 Vim，它就把此种信息存放在一个文件内。即 viminfo 信息文件。当 Vim 重新起动时，就读取这个信息文件，而那些信息就被还原了。

http://www.cnblogs.com/lhb25/p/130-essential-vim-commands.html

第十章 Shell 脚本

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。

Shell 基础知识

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。

什么是 shell

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。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，简称 sh。Bash 有以下特点：

- 记录命令历史

我们敲过的命令，linux 是会有记录的，默认可以记录 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；

```
[azureuser@mono tsar-master]$ pwd
/usr/local/src/tsar-master
[azureuser@mono tsar-master]$ !!
pwd
/usr/local/src/tsar-master
```

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 !12 表示执行命令历史中第 12 个命令；

```
[azureuser@mono tsar-master]$ history | grep 120
  120  ls /usr/lib/mono
  558  history | grep 120
[azureuser@mono tsar-master]$ !120
ls /usr/lib/mono
2.0  4.0  compat-2.0  mono-configuration-crypto  xbuild
3.5  4.5  gac         monodoc                    xbuild-frameworks
```

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。

```
[azureuser@mono tsar-master]$ !pw
pwd
/usr/local/src/tsar-master
```

- 指令和文件名补全

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。

- 别名

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 会看到目前系统默认的 alias.

系统默认的 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']

```
[azureuser@mono tsar-master]$ alias dotnet='mono'
[azureuser@mono tsar-master]$ dotnet
Usage is: mono [options] program [program-options]

Development:
    --aot[=<options>]          Compiles the assembly to native code
    --debug[=<options>]        Enable debugging support, use --help-debug for details
    --debugger-agent=options Enable the debugger agent
    --profile[=profiler]      Runs in profiling mode with the specified profiler module
    --trace[=EXPR]             Enable tracing, use --help-trace for details
    --jitmap                    Output a jit method map to /tmp/perf-PID.map
    --help-devel               Shows more options available to developers

Runtime:
    --config FILE              Loads FILE as the Mono config
    --verbose, -v              Increases the verbosity level
    --help, -h                 Show usage information
    --version, -V              Show version information
    --runtime=VERSION          Use the VERSION runtime, instead of autodetecting
    --optimize=OPT             Turns on or off a specific optimization
                                  Use --list-opt to get a list of optimizations
    --security[=mode]          Turns on the unsupported security manager (off by default)
                                  mode is one of cas, core-clr, verifiable or validil
    --attach=OPTIONS           Pass OPTIONS to the attach agent in the runtime.
                                  Currently the only supported option is 'disable'.
    --llvm, --nollvm           Controls whenever the runtime uses LLVM to compile code.
    --gc=[sgen,boehm]          Select SGen or Boehm GC (runs mono or mono-sgen)
[azureuser@mono tsar-master]$ unalias dotnet
[azureuser@mono tsar-master]$ dotnet
-bash: dotnet: command not found
```

使用 ualias 命令别名 就可以把设置的别名给解除了。

- 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。

- 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>>

- 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。

- 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）。

环境变量

Linux 是一个多用户的操作系统。多用户意味着每个用户登录系统后，都有自己专用的运行环境。而这个环境是由一组变量所定义,这组变量被称为环境变量。用户可以对自己的环境变量进行修改以达到对环境的要求。
环境变量分为系统环境变量和用户环境变量。系统环境变量，对所有用户起作用，而用户环境变量只对当前用户起作用。

系统环境变量:
/etc/profile:此文件为系统的每个用户设置环境信息,当用户第一次登录时,该文件被执行.并从/etc/profile.d 目录的配置文件中搜集 shell 的设置.
/etc/bashrc:为每一个运行 bash shell 的用户执行此文件.当 bash shell 被打开时,该文件被读取.
当前用户变量:
~/.bash_profile: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!默认情况下,他设置一些环境变量,执行用户的.bashrc 文件.
~/.bashrc:该文件包含专用于你的 bash 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.
~/.bash_logout:当每次退出系统(退出 bash shell)时,执行该文件。

定制环境变量

环境变量是和 Shell 紧密相关的，它是通过 Shell 命令来设置的。环境变量又可以被所有当前用户所运行的程序所使用。对于 bash 来说，可以通过变量名来访问相应的环境变量。
下面通过几个实例来说明

1.显示环境变量 HOME
[azureuser@mono tmp]$ echo $PATH
/usr/lib64/qt-3.3/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/azureuser/bin
2.设置一个新的环境变量 NAME
[azureuser@mono tmp]$ export NAME="geffzhang"
[azureuser@mono tmp]$ echo $NAME
Geffzhang

3.使用 env 命令显示所有的环境变量
[azureuser@mono tmp]$ env
HOSTNAME=mono
SELINUX_ROLE_REQUESTED=
TERM=xterm
SHELL=/bin/bash
HISTSIZE=1000
SSH_CLIENT=183.17.170.60 13550 22
SELINUX_USE_CURRENT_RANGE=

QTDIR=/usr/lib64/qt-3.3
OLDPWD=/usr
QTINC=/usr/lib64/qt-3.3/include
SSH_TTY=/dev/pts/0
NAME=geffzhang
USER=azureuser
……
用 Export 命令可以设置环境变量，但是如果每回进入系统之后都要重新设置一遍环境变量就很烦人。Linux 给大家提供了自动设置环境变量的方法，那就 是更改 .bashrc 文件。 一般说来，有两个文件可以提供这种“进入系统时自动设置”的功能，一个是 /etc/bashrc，另一个是 ~/.bashrc。其中/etc/bashrc 是被每个用户执行的，而~/.bashrc 只被当前用户执行。所以/etc/bashrc 只有 root 用 户能更改，而~/.bashrc 是各个用户私有的文件。 "~"指的是用户的 home 目录。

4.使用 set 命令显示所有本地定义的 Shell 变量
$ set
BASH=/bin/bash
BASHOPTS=checkwinsize:cmdhist:expand_aliases:extquote:force_fignore:hostcomplete:interactive_comments:login_shell:progcomp:promptvars:sourcepath
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]="x86_64-redhat-linux-gnu")
BASH_VERSION='4.1.2(1)-release'
COLORS=/etc/DIR_COLORS
COLUMNS=78
…

5.使用 unset 命令来清除环境变量，我们把上面设置的 NAME 变量清除：
[azureuser@mono tmp]$ unset NAME
[azureuser@mono tmp]$ echo $NAME

Shell 特殊符号

在 shell 中常用的特殊符号罗列如下：
; ;; . , / \ 'string'| ! $ ${} $?
$$ $* "string"* ** ? : ^ $# $@
`command`{} [] [[]] () (()) || && {xx,yy,zz,...}~
~+ ~- & \<...\< + - %= == !=
就说一说常用到的特殊字符。

- * 代表零个或多个字符或数字。

```
[azureuser@mono ~]$ ls
BddifySamples  NPinyin  QuartzDemo  RazorDemo  RxDemo  test1  wcf
[azureuser@mono ~]$ ls -d test*
test1
```

- ？只代表一个任意的字符

```
[azureuser@mono ~]$ touch test2
[azureuser@mono ~]$ ls -d test?
test1  test2
```

不管是数字还是字母，只要是一个都能匹配出来。

- # 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。

```
[azureuser@mono ~]$ abc=123 #test
[azureuser@mono ~]$ echo $abc
123
```

- \ 脱意字符，将后面的特殊符号（例如”*” ）还原为普通字符。

```
[azureuser@mono ~]$ ls -d test\*
ls: cannot access test*: No such file or directory
```

- | 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。

```
[azureuser@mono ~]$ cat test1 | wc -l
0
```

- $ 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。

```
[azureuser@mono ~]$ ls test1
test1
[azureuser@mono ~]$ ls !$
ls test1
test1
```

- ; ：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。
- ~：用户的家目录，如果是 root 则是 /root ，普通用户则是 /home/username
- &：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。
- >, >>, 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>>
- [ ] 中括号，中间为字符组合，代表中间字符中的任意一个。
- && 与 ||，用于多条命令中间的特殊符号

Shell 脚本的基本结构

我们先来写一个 Hello World 的最简单的 Shell 脚本：

```
[azureuser@mono ~]$ cd /usr/local/sbin
```

```
[azureuser@mono sbin]$ sudo vi helloworld.sh
#! /bin/bash

## This is first bash shell on centos
## Written by Geffzhang 2013.8.18

date '+%Y-%m-%d %H:%M:%S'
echo "Hello world!"
```

Shell 脚本通常都是以.sh 为后缀名的，大家一看到.sh 文件就知道这是个 Shell 脚本。helloworld.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。脚本的注释虽然不是必须的，建议不要省略了。因为随着您写的 shell 脚本越来越多，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。而且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:

```
[azureuser@mono sbin]$ sh helloworld.sh
2013-08-18 07:26:59
Hello world!
```

Shell 脚本还有一种执行方法:

```
[azureuser@mono sbin]$ sudo ./helloworld.sh
sudo: ./helloworld.sh: command not found
[azureuser@mono sbin]$ sudo chmod +x helloworld.sh
[azureuser@mono sbin]$ sudo ./helloworld.sh
2013-08-18 07:29:03
Hello world!
```

用这种方法运行 shell 脚本，脚本本身要有执行权限，所以需要给脚本加一个 ‘x’权限，用 chmod 命令使 Shell 程序成为可执行的。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:

```
[azureuser@mono sbin]$ sh -x helloworld.sh
+ date '+%Y-%m-%d %H:%M:%S'
2013-08-18 07:33:08
+ echo 'Hello world!'
Hello world!
```

Shell 脚本中的本地变量

上面我们写的第一个脚本里面没有变量，前面我们已经介绍过环境变量，Shell 脚本还有本地变量。本地变量就如同局部变量一样，只在本 SHELL 中起作用。它不会影响到其他 SHELL 中的变量。使用变量无需事先声明，同时变量名的命名须遵循如下规则：

1. 变量名称只能是英文字母与数字，但是数字不能是开头字符，首个字符必须为字母（a-z，A-Z）
2. 中间不能有空格，可以使用下划线（_）

3. 不能使用标点符号
4. 不能使用 bash 里的关键字（可用 help 命令查看保留关键字）
5. 等号两边不能直接接空格符；

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：

```
[azureuser@mono sbin]$ sudo vi shellvar.sh
#! /bin/bash

## We will use shell local variables in this script
## Wirtten by geffzhang 2013.8.18

d=`date +%H:%M:%S`
echo "The script begin at $d."
echo "We'll sleep 4 secondes"
sleep 4
d1=`date +%H:%M:%S`
echo "The script end date $d1."
```

脚本中使用到了反引号，‘d’ 和‘d1’ 在脚本中作为变量出现，运行下结果如下：

```
[azureuser@mono sbin]$ sudo sh shellvar.sh
The script begin at 08:20:10.
We'll sleep 4 secondes
The script end date 08:20:14.
```

Shell 脚本还可以和用户交互:

```
[azureuser@mono sbin]$ sudo vi read.sh
#! /bin/bash

## Using 'read' in shell script
## Written by geffzhang 2013.8.18

read -p "Please input a number:" x
read -p "Please input a number:" y
sum=$[$x+$y]
echo "The sum of two numbers is: $sum"

[azureuser@mono sbin]$ sudo sh read.sh
Please input a number:1
Please input a number:2
The sum of two numbers is: 3
```

read 命令用于和用户交互，把用户输入的字符串作为变量值。数学计算要用[ ]括起来并且外头要带一个 ‘$’

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 脚本的默认变量。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。

```
[azureuser@mono sbin]$ sudo vi opetion.sh
#! /bin/bash

## Shell script option args
## Written by geffzhang 2013.8.18

sum=$[$1+$2]
echo "The sum of the two numbers is :$sum"
```

执行结果：

```
[azureuser@mono sbin]$ sudo sh -x opetion.sh 1 2
+ sum=3
+ echo 'The sum of the two numbers is :3'
The sum of the two numbers is :3
```

在脚本中，$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 脚本的默认变量是没有限制的。另外还有一个$0，不过它代表的是脚本本身的名字。

Shell 控制结构

Shell 有一个结构控制集合,我们再一次说明他们与其他的程序语言非常相像.

If 判断

if 语句是相当简单的:他测试一个命令的结果,并且有选择的执行一组语句:

```
if condition; then
    statements
else
    statements
fi
```

使用 if 命令:

下面的这个例子中显示 if 的普通用法,他会询问一个问题并依据这个问题来进行回答:

循环结构

Shell 中有 3 中循环，分别是 for,while,until，until 与 while 正好相反，前者条件为假则执行，后者条件为真时执行。基本和 C 语言中额循环没什么区别 ，只是语法上有些差异。我们将从标准的"for" 循环开始。这里有一个简单的例子。

```
#! /bin/bash
```

```
for x in one two three four
do
    echo number $x
done

[azureuser@mono ~]$ sudo sh forsh.sh
number one
number two
number three
number four
```

"for" 循环中的"for x" 部分定义了一个名为"$x" 的新环境变量（也称为循环控制变量），它的值被依次设置为"one"、"two"、"three" 和"four"。每一次赋值之后，执行一次循环体（"do" 和"done" 之间的代码）。在循环体内，象其它环境变量一样，使用标准的变量扩展语法来引用循环控制变量"$x"。还要注意，"for" 循环总是接收"in" 语句之后的某种类型的字列表。在本例中，指定了四个英语单词，但是字列表也可以引用磁盘上的文件，甚至文件通配符。

更多的循环结构："while" 和"until"

只要特定条件为真，"while" 语句就会执行，其格式如下：

```
while [ condition ]

do
      statements
done
```

通常使用"While" 语句来循环一定次数，比如，下例将循环 10 次：

```
myvar=0
while [ $myvar -ne 10 ]
do
      echo $myvar
      myvar=$(( $myvar + 1 ))
done
```

可以看到，上例使用了算术表达式来使条件最终为假，并导致循环终止。
"Until" 语句提供了与"while" 语句相反的功能：只要特定条件为假，它们就重复。下面是一个与前面的"while" 循环具有同等功能的"until" 循环：

```
myvar=0
until [ $myvar -eq 10 ]
do
      echo $myvar
      myvar=$(( $myvar + 1 ))
```

done

循环控制语句
break 命令不执行当前循环体内 break 下面的语句从当前循环退出.
continue 命令是程序在本循体内忽略下面的语句,从循环头开始执行

函数

在代码复用及可维护性方面，函数有着巨大的优势，因此，把常用功能封装成函数是一件非常平常的事。
BASH 是一个相对简单的脚本语言，不过为了方便结构化的设计，BASH 中也提供了函数定义的功能。BASH 中的函数定义很简单，只要向下面这样写就可以了：

```
function my_funcname
{
    code block
}
```

或者

```
my_funcname() {
     code block
}
```

上面的第二种写法更接近于 C 语言中的写法。BASH 中要求函数的定义必须在函数使用之前，这是和 C 语言用头文件说明函数方法的不同。

更进一步的问题是如何给函数传递参数和获得返回值。BASH 中函数参数的定义并不需要在函数定义处就制定，而只需要在函数被调用时用 BASH 的保留变量 $1 $2 … 来引用就可以了；BASH 的返回值可以用 return 语句来指定返回一个特定的整数，如果没有 return 语句显式的返回一个返回值，则返回值就是该函数最后一条语句执行的结果（一般为 0，如果执行失败返回错误码）。函数的返回值在调用该函数的程序体中通过 $? 保留字来获得。

第十一章 linux 系统日常管理

网络管理

监控系统的状态

Linux 的防火墙

系统服务管理

系统进程管理

http://www.cnblogs.com/haoshikui/archive/2011/12/29/2306434.html

数据备份工具 rsync

系统日志

screen 工具介绍

第十二章 crontab 计划任务

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，Windows 上大家都把它叫做计划任务。在 Linux 中，我们经常用到 cron 服务来完成这项工作。cron 服务可以根据配置文件约定的时间来执行特定的作务。比如我们可以在配置文件中约定每天早上 4 点，对 httpd 服务器重新启动，这就是一个计划任务；后续我们还会介绍 Quartz.NET 这个调度框架在我们的应用中实现 crontab 的类似功能。
crontab 命令常见于 Unix 和 Linux 的操作系统之中，用于设置周期性被执行的指令。该命令从标准输入设备读取指令，并将其存放于"crontab"文件中，以供之后读取和执行。
在 Linux 系统中，Linux 任务调度的工作主要分为以下两类：
1、系统执行的工作：系统周期性所要执行的工作，如备份系统数据、清理缓存。
2、个人执行的工作：某个用户定期要做的工作，例如每隔 10 分钟检查邮件服务器是否有新信，这些工作可由每个用户自行设置。

系统调度服务和配置文件

我们可以 chkconfig --list | grep cron 命令来查看 cron 服务的启动情况：
[azureuser@mono ~]$ chkconfig --list | grep cron
crond 0:off 1:off 2:on 3:on 4:on 5:on 6:off
我们可以看到，系统启动级别如果是 2-5，cron 服务都会开机自动启动的，我们可以通过如下命令来设置该服务的启动项：
service crond start
service crond status

service crond stop
service crond restart
service crond reload
以上 1-4 行分别为启动、服务状态、停止、重启服务和重新加载配置。
要把 cron 设为在开机的时候自动启动，在 /etc/rc.d/rc.local 脚本中加入 service crond start 即可。

系统调度的任务一般存放在/etc/crontab 这个文件下，里面存放了一些系统运行的调度程序，通过命令我们可以看一下里面的内容：

```
[azureuser@mono ~]$ cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root //如果出现错误，或者有数据输出，数据作为邮件发给这个帐号
HOME=/   // 使用者运行的路径,这里是根目录

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
```

这些任务都会是系统在运行起来后自动进行调度的。同时在/etc 目录下还存放了/etc/cron.deny 和 /etc/cron.allow 文件。
/etc/cron.deny 表示不能使用 crontab 命令的用户。
/etc/cron.allow 表示能使用 crontab 的用户。
如果两个文件同时存在，那么/etc/cron.allow 优先。
如果两个文件都不存在，那么只有 root 用户可以安排作业。

crontab 命令

crontab 命令的语法为：
crontab [-e [UserName]|-l [UserName]|-r [UserName]|-v [UserName]|File]
注意：crontab 是用来让使用者在固定时间或固定间隔执行程序之用，换句话说，也就是类似使用者的时程表。-u user 是指设置指定 user 的时程表，这个前提是你必须要有其权限(比如说是 root)才能够指定他人的时程表。如果不使用 -u user 的话，就是表示设置自己的时程表。
-e [UserName]: 执行文字编辑器来设置时程表，内定的文字编辑器是 vi
-r [UserName]: 删除目前的时程表
-l [UserName]: 列出目前的时程表

-v [UserName]:列出用户 cron 作业的状态

我们如果要创建自己的一个任务调度，可以使用 crontab -e 这个命令，如：
[azureuser@mono ~]$ sudo crontab –e
此时会进入到 vi 编辑界面，来编写我们要调度的任务，crontab 调度命令的格式如下：
* * * * * command path
// 前五个字段可以取整数值，指定何时开始工作，第六个域是字符串，即命令字段，其中包括了 crontab 调度执行的命令。 各个字段之间用 spaces 和 tabs 分割。
调度命令的规则：

字段名称	说明	取值范围
分钟	每小时的第几分钟执行	0-59
小时	每日的第几个小时执行	0-23
日期	每月的第几天执行	1-31
月历	每年的第几月执行	1-12
星期	每周的第几天执行	0-6
命令名称	欲执行的命令及参数	

crontab 命令中的一些常用特殊符号：

符号	说明
*	表示任何时刻
,	表示分割
—	表示一个段，如第二段里：1-5，就表示 1 到 5 点
n	表示每个 n 的单位执行一次，如第二段里，*/1, 就表示每隔 1 个小时执行一次命令。也可以写成 1-23/1

如果我希望的每天 10 点 01 分执行命令 echo "ok" > /root/cron.log，则输入下面的命令，则输入下面的命令：
[azureuser@mono ~]$ crontab -e
no crontab for azureuser - using an empty one
01 10 * * * echo "ok" >/home/azureuser/cron.log
crontab: installing new crontab
保存以后如果出现 crontab: installing new crontab 这句话，则表示该调度命令已经成功。
每次编辑完某个用户的 cron 设置后，cron 自动在/var/spool/cron 下生成一个与此用户同名的文件，此用户的 cron 信息都记录在这个文件中，这个文件是不可以直接编辑的，只可以用 crontab -e 来编辑。cron 启动后每过一份钟读一次这个文件，检查是否要执行里面的命令。因此此文件修改后不需要重新启动 cron 服务。查看已经设置的任务计划使用 crontab -l 命令:
[azureuser@mono ~]$ crontab -l
01 10 * * * echo "ok" >/home/azureuser/cron.log

如果我们需要终止自己刚创建的任务调度，则使用 crontab -r 命令即可

附录 Linux 常用命令

下面是格式说明，你现在可以跳过，直到遇到疑问时再来查询。

$ 命令行提示符

粗体表示命令

*斜体*表示参数

filename, file1, file2 都是文件名。有时文件名有后缀，比如 file.zip

command 命令名

dir 文件夹名

string 字符串

username 用户名

groupname 组名

regex 正则表达式

path 路径

device 设备名

partition 分区名

IP IP 地址

domain 域名

ID 远程用户 ID

host 主机名，可以为 IP 地址或者域名

var 变量名

value 变量值

命令帮助

$**man** *command*

查询命令 command 的说明文档

$**man** -k *keyword*

查询关键字

$**info** *command*

更加详细的说明文档

$**whatis** *command*

简要说明

$**which** *command*

command 的 binary 文件所在路径

$**whereis** *command*

在搜索路径中的所有 command

这里只是以 command (binary file) 为例。比如 man 还可以用于查询系统函数、配置文件等。

用户

$**finger** *username*

显示用户 username 的信息

$**who**

显示当前登陆用户

    $**who** am I

    一个有趣的用法

$**write** username

向用户发送信息 (用 EOF 结束输入)

$**su**

成为 root 用户

$**sudo** *command*

以 root 用户身份执行

$**passwd**

更改密码

SHELL (BASH)

$**history**

显示在当前 shell 下命令历史

$**alias**

显示所有的命令别称

$**alias** *new_command*='*command*'

将命令 *command* 别称为 *new_command*

$**env**

显示所有的环境变量

$**export** *var*=*value*

设置环境变量 *var* 为 *value*

$**expr** 1 + 1

计算 1+1

文件系统

$**du** -sh *dir*

文件夹大小，-h 人类可读的单位，-s 只显示摘要

$**find** . -name *filename*

从当前路径开始，向下寻找文件 *filename*

$**locate** *string*

寻找包含有 *string* 的路径

$**updatedb**

与 **find** 不同，**locate** 并不是实时查找。你需要更新数据库，以获得最新信息。

$**ln** -s *filename path*

为文件 *filename* 在 *path* 位置创建软链接

$**pwd**

显示当前路径

$**cd** *path*

更改当前工作路径为 *path*

$**cd** -

更改当前路径为之前的路径

文件

$**touch** *filename*

如果文件不存在，创建一个空白文件；如果文件存在，更新文件读取和修改时间。

$**rm** *filename*

删除文件

$**cp** *file1 file2*

复制 *file1* 为 *file2*

$**ls** -l *path*

显示文件和文件相关信息

$**mkdir** *dir*

创建 dir 文件夹

$**mkdir** -p *path*

递归创建路径 path 上的所有文件夹

$**rmdir** *dir*

删除 *dir* 文件夹，*dir* 必须为空文件夹。

$**rm** -r *dir*

删除 *dir* 文件夹，以及其包含的所有文件

$**file** *filename*

文件 filename 的类型描述

$**chown** *username*:*groupname filename*

更改文件的拥有用会为 *owner*，拥有组为 *group*

$**chmod** *755 filename*

更改文件的权限为 755: owner r+w+x, group: r+x, others: r+x

$**od** -c *filename*

以 ASCII 字符显示文件

文件显示

$**cat** *filename*

显示文件

    $**cat** *file1 file2*

    连接显示 file1 和 file2

$**head** -1 *filename*

显示文件第一行

$**tail** -5 *filename*

显示文件倒数第五行

$**diff** *file1 file2*

显示 *file1* 和 *file2* 的差别

$**sort** *filename*

对文件中的行排序，并显示

    $**sort** -f *filename*

    排序时，不考虑大小写

    $**sort** -u *filename*

    排序，并去掉重复的行

$**uniq** *filename*

显示文件 *filename* 中不重复的行 （内容相同，但不相邻的行，不算做重复）

$**wc** *filename*

统计文件中的字符、词和行数

    $**wc** -l *filename*

    统计文件中的行数

文本

$**echo** *string*

显示 *string*

$**echo** *string* | **cut** -c5-7

截取文本的第 5 到第 7 列

$**echo** *string* | **grep** *regex*

显示包含正则表达式 *regex* 的行

$**echo** *string* | **grep** -o *regex*

显示符合正则 *regrex* 的子字符串

时间与日期

$**date**

当前日期时间

$**date** +"%Y-%m-%d_%T"

以 YYYY-MM-DD_HH:MM:SS 的格式显示日期时间 (格式可参考$man date)

$**date** --date="1999-01-03 05:30:00" 100 days

显示从 1900-01-03 05:30:00 向后 100 天的日期时间

$**sleep** 300

休眠 300 秒

进程

$**top**

显示进程信息，并实时更新

$**ps**

显示当前 shell 下的进程

$**ps** -lu *username*

显示用户 *username* 的进程

$**ps** -ajx

以比较完整的格式显示所有的进程

$**kill** *PID*

杀死 PID 进程 (PID 为 Process ID)

$**kill** %*job*

杀死 job 工作 (job 为 job number)

$**lsof** -u *username*

用户 *username* 的进程所打开的文件

$**dmesg**

显示系统日志

$**time** *a.out*

测试 a.out 的运行时间

硬件

$**uname** -a

显示系统信息

$**df** -lh

显示所有硬盘的使用状况

$**mount**

显示所有的硬盘分区挂载

$**mount** *partition path*

挂在 *partition* 到路径 *path*

$**umount** *partition*

卸载 *partition*

$**sudo fdisk** -l

显示所有的分区

$**sudo fdisk** *device*

为 device(比如/dev/sdc)创建分区表。 进入后选择 n, p, w

$**sudo mkfs** -t ext3 *partition*

格式化分区 *patition*(比如/dev/sdc1)

修改 /etc/fstab，以自动挂载分区。增加行：

/dev/sdc1 path(mount point) ext3 defaults 0 0

$**arch**

显示架构

$**cat** /proc/cpuinfo

显示 CPU 信息

$**cat** /proc/meminfo

显示内存信息

$**free**

显示内存使用状况

$**pagesize**

显示内存 page 大小（以 KByte 为单位）

网络

$**ifconfig**

显示网络接口以及相应的 IP 地址。ifconfig 可用于设置网络接口

> $**ifup** *eth0*
>
> 运行 *eth0* 接口
>
> $**ifdown** *eth0*
>
> 关闭 *eth0* 接口

$**iwconfig**

显示无线网络接口

$**route**

显示路由表。route 还可以用于修改路由表

$**netstat**

显示当前的网络连接状态

$**ping** *IP*

发送 ping 包到地址 *IP*

$**traceroute** *IP*

探测前往地址 *IP* 的路由路径

$**dhclient**

向 DHCP 主机发送 DHCP 请求，以获得 IP 地址以及其他设置信息。

$**host** *domain*

DNS 查询，寻找域名 *domain* 对应的 IP

> $**host** *IP*
>
> 反向 DNS 查询

$**wget** *url*

使用 **wget** 下载 *url* 指向的资源

$**wget** -m *url*

镜像下载

SSH 登陆与文件传输

$**ssh** *ID@host*

ssh 登陆远程服务器 *host*，*ID* 为用户名。

$**sftp** *ID@host*

登陆服务器 *host*，*ID* 为用户名。sftp 登陆后，可以使用下面的命令进一步操作：

```
get filename     # 下载文件
put filename     # 上传文件
ls               # 列出 host 上当前路径的所有文件
cd               # 在 host 上更改当前路径
lls              # 列出本地主机上当前路径的所有文件
lcd              # 在本地主机更改当前路径
```

$**scp** *localpath ID@host:path*

将本地 *localpath* 指向的文件上传到远程主机的 *path* 路径

$**scp** -r *ID@site:path localpath*

以 ssh 协议，遍历下载 *path* 路径下的整个文件系统，到本地的 *localpath*

压缩与归档

$**zip** *file.zip file1 file2*

将 *file1* 和 *file2* 压缩到 *file.zip*

$**unzip** *file.zip*

解压缩 *file.zip*

$**gzip** -c *filename > file.gz*

将文件 *filename* 压缩到 *file.gz*

$**gunzip** *file.gz*

解压缩 *file.gz* 文件

$**tar** -cf *file.tar file1 file2*

创建 tar 归档

$**tar** -zcvf *file.tar file1 file2*

创建 tar 归档，并压缩

$**tar** -xf *file.tar*

释放 tar 归档

$**tar** -zxf *file.tar.gz*

解压并释放 tar 归档

打印

$**lpr** *filename*

打印文件

$**lpstat**

显示所有打印机的状态