- 欢迎来到 Ubuntu 部落

索引

        - tar -c|x|u|r|t[z|j][v] -f <归档文件>[未打包文件]
- zip[参数]<压缩包> <源文件>
- unzip[参数]<压缩文件>[压缩包中将被释放的文件]
- 7z|7za <子命令>[参数]<压缩包>[文件]
- rar <子命令>[参数]<压缩包>[文件|文件列表路径]
    - 搜索
        - whereis <程序名称>
- locate <文件名称>
- find[路径]<表达式>
- grep <字符串>|"<正则表达式>"[文件名]
    - 其它
        - echo <字符串>
- clear
- alias <输入内容> <实际内容>
- export <变量名称>
- shutdown
- halt
- reboot
- chroot <路径>
    - 用户管理
        - 一些细节
- su[用户名]
- sudo[命令]
- passwd[用户名]
- chsh[-s <Shell>][用户名]
- usermod <用户名>
- useradd <用户名>
- userdel <用户名>
- id[用户名]
- finger[用户名]
    - 进程管理
        - 一些细节
- ps
- pstree
- pgrep <进程名>
- xkill
- pkill <进程名>
- kill[信号代码]<进程 PID>
- renice <优先级表达式> <进程表达式>
- top

Ubuntu 的发音

Ubuntu，源于非洲祖鲁人和科萨人的语言，发作 *oo-boon-too* 的音。 了解发音是有意义的，您不是第一个为此困惑的人，当然，也不会是最后一个：）

大多数的美国人读 *ubuntu* 时，将 u 作为元音发音， 类似单词 who 或者 boo ， 重音在第二个音节即 *u'buntu* ， *oo-boon-too* 。

如果您喜欢撒哈拉，喜欢它令人窒息的温柔、梦幻般的寂寥还有张扬恣肆的旷远， 您大可在第一个 u，后面带些嗡嗡声： *oom-boon-too*。

Ubuntu 的中文发音大约为： **乌班图**

Ubuntu 的涵义

Ubuntu 这个单词源自非洲，意谓“班图精神”————谁都不是一座孤岛，自成一体。每个人都包孕于人类，因他人存在而存在，因他人幸福而幸福。

Ubuntu 当前版本

Ubuntu Linux v6.06 LTS (Dapper Drake)

LTS: Long Term Support

Dapper Drake: 当前版本的开发代号

Ubuntu 的特点

Ubuntu 完全基于 Linux 操作系统， 可以免费得到社区及专业机构的支持。庞大的社区是它成长的沃土，请向这片动人的热忱敞开心扉。

Ubuntu 社区恪守 Ubuntu 理念：自由！软件应是自由的，应尊重人类的自由意志，它与人类之间不应有任何隔膜。本地语种，功能限制，自主改进的权利……都不应成为使用的障碍或负担。

自由，让 Ubuntu 与传统的私有软件从根本上不同：免费不能用来遮羞，您有权修正它，直到满意为止。

Ubuntu 适合桌面和服务器。当前 Ubuntu 发布版支持 PC (Intel x86), 64-bit PC (AMD64) 和 PowerPC (Apple iBook 和 Powerbook, G4 和 G5) 架构。

Ubuntu 包涵了超过 16,000 种软件， 核心的桌面系统却只有一张光盘，Ubuntu 覆盖了所有的桌面应用程序，从文字处理，电子表格到 web 服务器和开发设计环境一应俱全。 详情查看 Ubuntu 桌面 和 Ubuntu 服务器的介绍。

Ubuntu 相对其它 Linux 发行版的主要特点

基于 Debian/Linux，使用 APT 包管理系统。

相对于 Fedora Code： APT 包管理系统优雅地解决了依赖问题，并且可以从容的在线安装升级

相对于 Debian：软件更新积极，而 Debian 较保守。

相对于 Gentoo：基本无需编译，省力、省时、省心。

基本操作

进入系统

在登录界面中输入您的用户名，然后系统将提问您的密码

（图）

输入您的密码后，回车，稍事等待，您便可以进入 Ubuntu 系统 （图）

点击桌面左上角的图标，您可以打开一个菜单（或者使用 A1t+F1 组合键）（图）

如果您想退出系统，可以点击该图标 （图）

在桌面上方启动栏中，包含了一些常用程序的启动图标 （图） 这些图标也可以在开始菜单中找到

现在点击 FireFox 图标，您便可以使用 FireFox 浏览器冲浪 （图）

或者按下 A1t+F2 组合键，弹出一个运行命令对话框。输入 firefox 后回车，同样可以启动 FireFox （图）

在菜单中找到 终端 （图）

点击它便开启了一个终端窗口，您可以在终端窗口中运行命令 （图）

也可以在控制台中输入命令。使用 Ctr1+A1t+[F1~F6] ，您可以切换到 1~6 号控制台

使用 Ctr1+A1t+F7 可以返回图形界面 （您可以使用 Ctr1+A1t+BackSpace 将图形界面关闭） [1]

命令行提示符

如图， user@ubuntu:~$ 为命令提示符， @ 之前的部分为当前用户 ID， @ 与 ： 之间的部分，为您的主机名称， ：与 $ 之间的部分，为当前的路径。 （图）

退出系统

您可以点击这一个图标来退出系统 （图）

也可以在终端或者控制台中输入命令

```
sudo halt
```

系统会提问您密码，输入正确密码，便可以退出系统

[1] 在以后的章节中，如果我们提示您输入命令，那么您即可以在终端中输入，也可以在控制台中输入。如果只是启动应用程序，还可以使用 Alt+F2 组合键。

安装

分区概念

首先我们需要知道，硬盘分区的存在，是由硬盘的物理特性决定的，并不会因为不同的操作系统而有所改变。

请您把一块硬盘想象为一本书……即便您不喜欢读书，您也一定非常熟悉它，所有的书都是相同的， 包括我们使用的课本……您肯定非常熟悉:)

一本完整的书，通常包括书名、索引和正文。

如果您需要 Linux，您首先需要找到一本书名为《linux》的书，书名相当于硬盘中的 MBR， 也就是主引导纪录。不同的是，MBR 可以是几个书名合在一起，类似于《XX 合订本》。 这部分内容暂时还没有什么实用价值，您只需要大概的了解。

而正文，就是硬盘中纪录的数据，这也非常容易理解，且对于安装系统并没有什么影响， 所以现在我们来了解索引:)

索引相当于硬盘中的分区表，书中的每一个章节，相当于硬盘中的一个分区，它起始和结束的页次， 都可以在索引中找到。试想，如

果阅读一本撕掉索引的书，您将很难找到您想阅读的部分。 同样，如果没有分区表，操作系统也不能够在硬盘上定位数据的位置。

由于历史的原因，硬盘中的分区表大小受到了限制，最多只可以容纳四个分区（主分区）。 如果一本书，它的索引最多只能有四个章节，那不是太可怕了么？很多书的内容远远不止四个章节啊！

于是聪明的人们想到了一个变通的办法，就是利用其中的一个章节，来存储其它部分的索引。 比如第一章是前言，第二章是其它部分的索引，我们翻到第二章，呵呵，这里是第二个索引， 因为只有第一个索引受四个章节的限制，所以这个索引的内容可以非常的详尽。第二个索引就是分区表中的扩展分区了，其中定义的章节，就是硬盘中的逻辑分区，不是很难理解吧?

明白了这一点，我们来看看 Linux 和 Windows 对于分区不同的表示方法:

可能您已经很熟悉 Windows 了，它使用盘符来表示分区，比如 C: D: E:， 每一个分区使用一个盘符来标识，而且顺序可以颠倒， D: 并不一定就是您系统中的第二个分区。 （如果您给第二个分区分配最后一个硬盘盘符，把所有的盘符按顺序排列好，并且重装一次系统，您就会理解什么叫作“头疼”了：）

而在 Linux 中，分区是这样表示的

```
/dev/hda
/dev/hda1
/dev/hda2
/dev/hda5
/dev/sdb1
```

以 /dev/hda5 为例:

因为在 Linux 中，每一个设备都是用 /dev/ 文件夹下的一个文件来表示， 所以 /dev/hda5 中， /dev/ 表示的是根目录下的 dev 目录，我们来看剩下的部分 hda5 。

前两位的字母 hd 表示这是一块 IDE 硬盘，如果是 sd ，则代表 SATA 硬盘，或者闪存等外设。

第三位的字母 a 表示这是该类型接口上的第一个设备。 同理，b、c、d…… 分别代表该类型接口上的第二三四……个设备。 例如 hdc 表示第二个 IDE 接口上的主硬盘（每个 IDE 接口上允许一个主设

备和一个从设备）。

第四位的数字 5 ，并不表示这是该硬盘中的第 5 个分区，而是第一个逻辑分区。 因为在 Linux 中，为了避免不必要的混乱，分区的顺序是不能改变的， 分区标识则由它们在硬盘中的位置决定。 系统又要为所有可能的主分区预留标识，所以 1-4 一定不会是逻辑分区， 5 则是第一个逻辑分区，以此类推。

安装中的注意事项

在 Ubuntu 系统的安装过程中，您需要选择系统目录的挂载点。

我们知道，安装 Windows 时，我们可以选择把系统安装在哪一个分区，把系统挂载到分区上。而在 Ubuntu/Linux 中则相反，我们要把分区挂载到系统中。

当我们使用 Windows 的安装方式，把系统挂载到分区上，我们就不可能把 Windows 目录放在 C 盘，而把 MyDocuments 目录放到其它分 区。您或者出于习惯，或者出于数据安全方面的考虑，通常把文档放到其它分区中。但是 Windows 下很多软件保存文件的默认目录就是 MyDocument 目录，这就比较不方便。

在系统安装完成后，我们还是可以将 MyDocuments 目录转移到其它分区中，不过有点麻烦，可能许多朋友还不知道怎么去作……

而任何一种 Linux 系统时，当然包括 Ubuntu，我们可以在系统安装时就把分区挂载到目录下， /home 目录相当于 Windows 的 MyDocuments ，我们可以把 /dev/hda5 挂载到此目录下，这样我们往 /home 目录里存东西的时候，其实保存在第一个扩展分区中。如果再一次安装系统，只要把这个分区挂载到 /home 目录下，那么进入新系统就像回家一样，真是太棒了。

理论上来讲，您可以将分区挂载到任何目录下面，您可以自定义挂载的路径。但是我们并不推荐您这么作，因为那没有任何意义。

系统安装程序向您建议的挂载目录，通常也是我们向您建议的，现在我们来了解一下，这些目录通常都是用来作什么的:

/

根目录，唯一必须挂载的目录。不要有任何的犹豫，选一个分区，挂载它！（在绝大多数情况下，有 2G 的容量应该是够用了。当然了，很多东西都是多多益善的: ）

swap

交换分区，可能不是必须的，不过按照传统，并且照顾到您的安全感，还是挂载它吧。它的容量只要大于您的物理内存就可以了，如果超过了您物理内存两倍的容量，那绝对是一种浪费。

/home

前面已经介绍过了，这是您的家目录，通常您自己创建的文件，都保存在这里，您最好给它分配一个分区

/usr

应用程序目录。大部分的软件都安装在这里。如果您计划安装许多软件，建议您也给它分配一个分区

/var

如果您要作一些服务器方面的应用，可以考虑给它分配一个较大的分区

/boot

如果您的硬盘不支持 LBA 模式（我想那不太可能:），您最好挂载它，如果挂载硬盘的第一个分区，应该比较稳妥。一般来说，挂载的分区只要 100M 大小就足够了

[2]

在文件系统这一环节中，我们建议您选择： ReiserFS

[2] 也许您注意到了，Windows 中，盘符既用于表示硬件（硬盘上的分区）,又用于表示系统中的路径。而 Linux 中，硬件就是硬件，路径就是路径，不会混淆在一起，简单直接！

开始安装

这里 的教程图文并茂，内容详尽权威

Linux 基础

She11

可能您早已能够熟练的使用 GUI（图形用户界面），例如您可以使用鼠标双击一个图标，来打开或者执行它。

我们来看这个过程：您使用鼠标定位桌面上的一个程序图标，按下左键两次。系统读取鼠标指针的位置，并且判断该位置下图标的涵义，根据预设的双击动作，运行程序或者打开文件。

这一套 GUI 系统，便是一种 She11，它的作用是实现人机交互。如果我们不能够控制电脑，那么电脑还不如电视机好玩，不是么？电视机也可以选择频道（电视机的遥控器，也是一种人机交互的界面，不过相对于电脑，确实是相当简单了：）

易于上手、界面直观是 GUI 的优点，但是 GUI 为不意味着简单！或许您有类似经历：桌面上有几十个程序的启动图标，也知道它们的名字，但是翻出一个来，并不是一件轻松的事情。

我的 Windows 系统中，桌面上摆满了各种图标，每当启动一个程序的时候，我都很是困扰。后来尝试了 *音速启动* 这类的程序启动管理器，效果还是差强人意。

在我的不懈努力下，这个难道最终得到了解决： 将快捷方式名称简化，放到特定目录下，使用 Win+R 组合键呼出 *运行* 对话框，键入快捷方式的名称来运行该程序。比如 反恐精英 的快捷方式为 cs ，我把它放在 Windows 目录下；运行 cs 命令，就可以去维护世界和平了。

这么多快捷方式，统统放到 Windows 目录下，非常混乱。因此，我在 D 盘建立了一个名为 path 的目录，并把它的路径加入到环境变量的 path 项中，快捷方式放在 D:\path 目录中。即便重装系统，只要在环境变量中重新加入此路径，原来的程序大多可以直接以命令来运行……我的许多朋友强烈要示我帮他们设定这种启动方式，因为这确实很方便：）

其实在 Linux 下，所有的程序都可以通过命令运行。虽然 Linux 也有 GUI，但是它并不比 Windows 的 GUI 更优秀！上面只是简单的介绍了 CLI（命令行界面）相对 GUI 的优越之处，使用 CLI 还有更多的好处，您会慢慢体会到的。

当然了，在您的印象中，CLI 一定非常的不友善，缺少亲和力，冷漠而拒人于千里之外……您和 CLI 之间甚至有代沟的存在：）

命令

坦白的说，冷不丁见到那么老长的一串命令，谁都会毛骨悚然。

也许您使用过 DOS，留下这种印象：命令先放一边，光是正确的输入目录、文件名都够瞧的。而且 DOS 不区分大小写，要是像 Linux 一样区分大小写，那多恐怖啊！！！

其实 Linux 命令行具有补全功能，非常实用。假设有这样一个命令：

```
command path/file
```

如果只有一个以 c 起始的命令，键入 c ，再按一次 tab 键，系统将自动补全该命令余下的部分。 只要 c tab 两次按键，就可以完成 command 的输入。

如果不只一个 c 起始的命令，那么您可以按两次 tab 键，系统会列出所有符合条件的选项，也就是以 c 起始的所有命令。进一步输入 o ，如果只有一个以 co 起始的命令（一直输入，直到项符合条件的选项唯一），再按一次 tab ，命令就被补全完整。

路径和文件名也可以通过 tab 键来补全。还有一种 遍历补全 的方式，如果您的文件名是中文，而您不想切换输入法；甚至您的文件名中出现乱码，无法输入，这时 遍历补全 就可以大显身手了。这部分内容我们稍后再谈：）

现在我们来了解命令的语法结构，这一部分相当重要，您可得看仔细。

我们知道，任何语言都有特定的语法结构，以我们的中文为例：

```
我们郑重地推荐您 Ubuntu/Linux！
```

这个句子的语法尽管简单，却是大部分的命令行采用的句型。让我们看一下，这个句子里都有些什么：

我们

主语，Linux 命令的执行者只有一个，所以主语一概省略。

推荐

一个动词，作为谓语而存在。Linux 命令中，这一部分是必须的。这一部分也是不同命令之间最根本的区别方式，所以它通常作为命令名，写在最前面。键入 date 命令，您可以查看当前的时间日期。（ 应用程序->附件->终端 ）

郑重的

状语，用来修饰谓语。与之相对应，Linux 命令可以使用参数来精细调节程序的行为。为了与命令的操作对象相区别，参数前通常要加 - 或者 -- 符号。原则上，在命令名之后，参数的位置可以随意，但是为了养成一个良好的习惯，我们建议您在命令名后直接跟参数。

您 Ubuntu/linux

这两个部分都是宾语，它们是命令的操作对象。大部分的命令只有一个操作对象，也有一些命令是双宾语结构的，具有一个直接宾语和一个间接宾语。比如 拷贝 这个命令 cp（copy）

分隔符

我们的汉语是象形文字，没有分隔符。但是所有的拼音文字中都有分隔符，来分隔单词。Linux 命令中同样使用空格作分隔符。

cp a /home 表示把当前目录下的 a 文件，拷贝到 /home 目录下。（命令的不同部分使用空格分隔，连续的空格视为一个空格）

上面的那句话，翻译成 Linux 的命令，应该是这个样子的：

推荐 --郑重的 您 Ubuntu/Linux （按照传统，“-”后跟简写为单个字母的参数，“--”后跟完整单词的参数。不过也有例外：）

哈，Linux 的命令也蛮简单吧？

Linux 程序、进程

或许您会这样想，Linux 命令的句型确实不难，但是那么多命令，我怎么知道它们都是作什么的呢？而且不同的系统中，可以使用的命令似乎也不太一样，

这真让人困惑……

其实 Linux 的命令，运行的是 Linux 系统中的程序。只要您已安装了程序，您就可以通过命令来运行它，并且可以使用参数来精细的调整它的运行状态。也可以通过点击启动图标来运行，不过启动图标不能够方便的调整参数，并不是很方便。

举一个例子：

```
mplayer -shuffle -loop 3 -playlist mymp3.list
```

可能您运行上面命令，系统会提示您 无法找到命令 ，那是因为您没有安装 mplayer 这个程序。mplayer 是我见过的支持格式最多的播放器，几乎任何已知格式的多媒体文件，都可以使用 mplayer 来播放。它包含一个图形界面的前端，您可以在菜单中找到它，鼠标点击运行;也可以通过执行命令来运行它的命令行版本。

如果您的系统中没有 mplayer 播放器，我们建议您安装一个。关于程序的安装，请参阅 软件安装 。

上面命令中， mplayer 调用了 mplayer 播放器程序。参数 -shuffle 表示随机播放， -loop 表示循环播放，后面的 3 为循环的次数，如果为 0 ,则一直播放。 -playlist 表示播放列表中的曲目。我们可以把 mp3 的路径放到 mymp3.list 文件中，让 mplayer 来播放它们。

进程 为运行中的程序，是程序在内存中的镜像。

好了，现在您已经了解了 Shell 、 命令 、 程序 、 进程 的概念，您基本上也就了解了 Linux（Linux 系统真是非常简洁，而且容易理解：）。

但只知道这些，您并不能顺畅使用。接下来的章节中，我们来进一步介绍它的细节。

Linux 系统简介

路径

路径分为绝对路径和相对路径。

绝对路径的起始点为根目录 / ，例如 /usr/local/bin 就是绝对路径，它指向系统中一个绝对的位置，不受其它因素影响。

相对路径的起始点为当前目录，如果您现在位于 /usr 目录，那么相对路径 local/bin 所指示的位置为 /usr/local/bin

也就是说，相对路径所指示的位置，除了相对路径本身，还受到当前位置的影响。例如 Linux 系统中常见的目录 /bin 、 /usr/bin 、 /usr/local/bin ，如果只有一个相对路径 bin，那么它指示的位置可能上面三个目录中的任意一个，也可能是其它目录。

如果我告诉您到 bin 目录寻找一个文件，您可能搞不清楚是哪一个 bin 目录。只有当前位置确定，相对路径指示的位置才能够确定。

现在我说， /usr/local 目录下，它的相对路径 bin 中有某个文件，这样就比较明确了。

在相对路径中 . 表示当前目录， .. 表示当前目录的上一级目录。

假设您安装了一个程序，它的主程序没有被放置到上面三个 bin 目录中的任何一个，或者其它系统能够找到的地方，您就得告诉系统，它的可执行文件在哪里。

可以使用绝对路径，例如： /home/user/bin/可执行文件

或者定位到 /home/user/bin 目录，使用相对目录来定位它 ./可执行文件

如果您定位到了它的子目录，比如 /home/user/bin/gui，您可以使用 .. 来表示它的上级目录 ../可执行文件

路径相关命令

cd (change directory)	更改目录。
pwd (print working directory)	显示当前路径。
ls (list)	显示当前目录中的文件列表。

请尝试以下操作：

cd /etc	进入“/etc”目录，这里使用的是绝对路径
pwd	显示当前路径，这个命令返回结果“/etc”
cd init.d	进入“/etc”目录的子目录“init.d”，这里使用的是相对路径

cd ..　　进入上一级目录“/etc”
cd ../home　“/etc”　　目录的上一级目录为“/”，它的子目录“home”为“/home”
cd -　　回到上一次的目录，我们在“/etc”目录跳转到“/home”目录，所以这次是回到“/etc”目录
cd ~　　“~”代表当前用户的“$HOME”目录，也就是“/home/{用户名}”目录。
ls　　在任何时候，您都可以使用“ls”命令，来了解当前目录下都有哪些文件。

远程路径:

远程路径的表示方法为 **协议://用户名:密码@位置/路径:端口**

大多数的远程路径可以使用默认端口匿名访问，由此用户名、密码、端口通常不需要填写。例如:

http://www.ubuntu.org.cn/.../index.html

要求身份验证的远程路径，您可以使用下面的方式访问:

ftp://user:passwd@ftp.ubuntu.org.cn:21

软件

Linux 中没有 *注册表* 这个概念。安装软件，理论上讲，只要拷贝所有相关文件，并运行它的主程序就可以了。

按照传统，一个软件通常分别拷贝到同级目录下的 **bin**、**etc**、**lib**、**share** 等文件夹。

bin

可执行文件，程序的可执行文件通常在这个目录下。在环境变量中设定搜索路径，就可以直接执行，而不需要定位其路径。

etc

配置文件，大部分系统程序的配置文件保存于 /etc 目录，便于集中修改。

lib

库文件，集中在一起，方便共享给不同程序。相较不同的软件

单独保存库文件，能够节约一些磁盘空间。

share

程序运行所需要的其它资源，例如图标、文本。这部分文件是专有的，不需要共享；而且目录结构相对复杂，混放在一起比较混乱，所以单独存放。

还有一些软件，占用一个单独的目录，所有的资源都在这个目录中。类似于 Windows 下的绿色软件，不推荐在 Linux 系统下这样作。

- 执行时，系统找不到可执行文件（搜索所有路径，资源开销过大，是不现实的），需要定位其位置，像这样 /home/user/bin/可执行文件 ，不够方便。
- 许多系统软件需要协作运行，配置文件分别保存，定位它们非常麻烦
- 如果程序使用的库文件，像图形库文件，都单独存放，那么磁盘空间的浪费会非常严重。

有一些大型软件，或者您布署的重要应用，您可以将它们单独安装在一个文件夹下。（通常源码安装支持这种方式，将在 软件安装 部分介绍）

配置方式

Linux 下没有类似 *注册表* 的系统，系统和软件都可以通过纯文本的配置文件进行设置。

事实上，图形界面的配置工具，通常就是以图形界面的方式修改配置文件，适合设置一些比较简单的程序。如果软件有几千个可以配置的选项，全部作成菜单，想象一下……开始发抖吧……

图形界面的配置工具，可以看作特定配置文件专用编辑器。您一样可以使用通用文本编辑器来编辑配置文件，比如 Nano、Gedit、Knote、Vim 或者 Emacs 等等。不考虑阅读、修改配置文本占用的时间，直接修改配置文件甚至更迅速。[3]

如果只是要修改某一常用选项，而且时常修改，比如主机的 IP 地址。使用文本编辑器，您要找到相应的配置文件，还要在配置文件中找到相应的选项，会浪费掉您的时间和耐性。

图形配置工具经常会受各种因素制约，比如网络服务器中不提供图形服务，图形界面不够稳定……这时，您可以使用命令行的配置工具来完成这些工作。

例如： 修改主机 IP 地址，可以使用 ifconfig 这个程序，执行下面的命令：

```
ifconfig eth0 192.168.0.1
```

[3] 在以后的章节中，如果我们提示您修改某一文件，例如 /etc/fstab ，您可以使用任何顺手的文本编辑器打开它。

隐藏文件

Linux 下，名称中第一个字符为 . 的文件或者文件夹，系统默认情况下将它们隐藏起来，

您可以尝试以下操作：

```
cd ~      进入您的用户目录
ls        查看当前目录下的文件列表
ls -a     查看所有文件的文件列表（包括隐藏文件）。
```

现在，您可以看到许多文件名以 . 起始的文件或者文件夹了吧？使用 ls 命令无法显示它们

- 如果您只想查看隐藏文件，而不包括这两个特殊目录，您可以使用 ls 命令的参数 -A （ls -A）
- 每个目录下都包含两个特殊目录 . 和 .. 。您也许猜到了， . 代表当前目录， .. 代表上一级目录。目录是一种特殊类型的文件！

文件类型

Linux 系统主要根据文件头信息来判断文件类型，扩展名并非决定因素。

现在使用 ls -l 命令，查看详细信息格式的文件列表，您将会看到如下内容：

```
total 5
drwxr-x---  4 user  group   4096 Mar 10 00:37 filename
drwxr-xr-x 21 user  group   4096 Mar 10 20:16 文件名
-rw-------  1 user  group    524 Mar 10 00:40 a
-rw-r--r--  1 user  group     24 Jun 11  2000 b
drwx------  2 user  group   4096 Mar  9 11:06 c
```

共显示了七列信息，从左至右依次为：权限、文件数、归属用户、归属群组、

文件大小、创建日期、文件名称

其中要特别留意的是第一列：

```
drwxr-xr-x
```

一共有 10 个位置，可以分为 4 组：

```
d rwx r-x r-x
```

第一组只有一个字符：

- d 文件夹
- - 普通文件
- l 链接
- b 块设备文件
- c 字符设备文件。

剩下的 3 组分别为归属用户、归属群组、其它用户或群组对于该文件的权限。我们看它的格式

```
rwx rwx rwx
```

- r 可读
- w 可写
- x 可执行

它们的顺便不能颠倒，某一位置为空(-)，则表示不具有相应的权限。

> Tip
>
> Linux 下的可执行文件并不是由扩展名（例如 .exe ）决定的，而是由其可执行权限位决定。

权限

我们已经知道了，文件的权限分为 r （可读）、 w （可写）、 x （可执行）三种类型，而一个文件可以针对归属用户，归属群组，其它用户用户或群组分别设定权限。

这种权限管理的方式灵活、简单、严密、明晰。尽管如此，在最初的阶段，可能会有一点小小的不适。因为它无所不在，而您习惯了的 Windows 的权限管理却不是这样（非常混乱，大多数时间形同虚设，偶尔用到却让人伤透脑筋）。

使用 chmod 命令更改文件的权限，使用 chown 来更改文件的归属。 例如：

```
chmod 755 xxx
chmod a+x xxx
chown user:group xxx  用来更改文件的归属用户，也可以同时更改其归属群组
chgrp   group   xxx    用来更改文件的归属群组
```

上面命令中的 755 和 a+x 是两种类型的表达式

我们将后面章节中详细介绍 权限管理 用户管理

执行命令的权限

有一些命令，普通用户也可以执行，但是只有 root 用户才能执行成功，这是为什么呢?

例如在系统中增加一个新用户 useradd

```
ls -l /usr/sbin/useradd
```

可以看到：

```
-rwxr-xr-x 1 root root 56156 2006-04-03 21:37 /usr/sbin/useradd
```

明明所有的用户都可以执行嘛?

这是因为， useradd 命令是修改 /etc/passwd 文件的一个工具，来看看这个文件：

```
ls -l /etc/passwd
-rw-r--r-- 1 root root 1835 2006-06-24 17:58 /etc/passwd
```

原来只有 root 用户才能写入修改结果，非 root 用户执行 useradd 命令当然不会有结果。

执行命令的身份

默认情况下，您的命令提示符末位为 $ ，这表示您将以普通用户的身份执行命令。

您可以使用 su （switch user）这个命令来切换其它用户。

例如 su root ，切换到 root 用户，如果 su 命令后面没有切换目标，那么这个命令默认切换到 root 用户。

现在您执行 su 这个命令，系统会提示您输入密码，请输入管理员的密码。这个时候，您会发现命令提示符末位变成了 # ，您将以 root 用户的身份执行命令。

Ubuntu 系统默认会随机设定系统的 root 密码，这样会更安全一些，这个时候您可以执行“sudo”命令，输入当前用户密码后，暂时以 root 用户的身份执行命令。（前提是 sudoer 列表中要包含您的 ID。您在安装 Ubuntu 系统时创建的用户，默认具有“sudo”权限） [4]

[4] 如果您能够执行“sudo”命令，那么您也就拥有了 root 权限。在后面的章节中，如果我们提到了“root 权限”，那么您可以通过以上两种方式来实现

命令行

Shell、Console、Terminal

在前面的章节中，我们曾提到，电视机的遥控器，也是一种人机交互的界面，算是一种 Shell。

但是这个概念并不准确，遥控器只是向 Shell 发送指令的工具，Shell 接收到遥控器发出指令后，将指令转换为系统命令，由系统来执行。

例如我们按的遥控器上的 数字键 1 ,遥控器将 切换为 1 频道 的指令发送到 Shell，Shell 将指令转换为系统可以识别的 频道 1 ，系统执行它，您就可以观看 1 频道的电视节目了。

通常每台电视机只有一种 Shell，比如有的电视机系统具有“画中画”的功能，那么 Shell 中便有相应的功能定义，您可以通过遥控器上的“画中 画”功能键来开启它。假设您的电视机没有此功能，Shell 中也就没有相应的功能定义。拥有一个带“画中画”功能控制键的遥控器，即便信号兼容，您还是不 能够使用这一功能：）

不用遥控器也可以控制电视机，假设您的遥控器丢了，您还可以走到电视机前，使用机身上的控制面板来控制它（相当于使用 Linux 的控制台）。但是您一定不喜欢这种方式，除非您想锻炼身体：）

在 Linux 系统中，由于图形界面和控制台的分辨率通常不一致，所以切换时要有一个延时。对于我们中文用户来讲，控制台下中文的显示也比较麻烦。而且控制台显示内容通常不如终端显示的全面。

所以我们推荐您使用终端来执行命令，它使用起来感觉很像遥控器：）

rxvt-unicode

通常情况下，您买一台电视机，只能获得一个遥控器。虽然它为您的电视机量身定作，能够最大限度发挥电视机的能力，但您却不一定喜欢它。说不定这个遥控器体形太大，持握不方便;或者它体形太小，容易失踪;又或者它的按键要么太硬，要么太软;它的键盘要么太大，要么太小……

您一般也可以容忍，毕竟遥控器使用频率并不算高：）

如果您的终端有些地方不讨您喜欢，比如说响应太慢，或者不能正常显示中文……那就难以忍受了，您应该换一个其它的试试。

在前面的章节，我们介绍您使用的终端为 Gnome-Termian1，它是系统默认使用的终端，显示中文不错，不过响应比较慢，您可能已经处于水深火热之中了。。。

我们推荐您使用 urxvt（m1term 也是不错的选择）

您可以使用 sudo apt-get install rxvt-unicode 命令来安装它。

urxvt 启动它 （urxvt 不支持控制台，您得在图形界面下启动它。终端、A1t+F2，建议您在启动栏里新建一个启动图标）

rxvt-unicode 还支持“服务器/客户端”的运行模式：

urxvtd 启动一个守护进程 daemon（支持控制台）
urxvtc 启动客户端 client。多个客户端可以同时连接到一个 urxvtd，以达到节省系统资源的目的。

或许您对 rxvt 的默认设置不满意，您可以修改用户配置文件 ~/.Xresources 来

设定它。修改全局配置文件 /etc/X11/Xresources/Xresources ，则对所有用户生效，只有 root 才可以修改此文件。

这里有一些简单的选项：（以！起始的行是注释，您可以直接拷贝此文件的内容）

```
!!============================================================================
!! RXVT-unicode setting
!!============================================================================
!设置字体分辨率
Xft.dpi:96
!窗口大小
Rxvt.geometry: 80x40+80+80
!颜色
Rxvt.background:#333333
Rxvt.foreground:antiquewhite
Rxvt.inheritPixmap:False
Rxvt.colorBD:yellow
Rxvt.colorUL:antiquewhite
!滚动条
Rxvt.scrollBar:True
Rxvt.scrollBar_left:True
Rxvt.scrollBar_floating:False
Rxvt.scrollstyle:next
Rxvt.scrollColor:#999999
!屏幕缓冲
Rxvt.saveLines:30000
Rxvt.color12:DodgerBlue
Rxvt.font:7x14,xft:AR PL New Sung
!输入法一般设置为 xim
!inputMethod:xim;Scim 除外
!输入法样式可选:Root(置底) OverTheSpot(跟随) OffTheSpot OnTheSpot，后两种不是所有的都支持
Rxvt.preeditType:Root
```

Tip：右键点击启动栏， 添加自启动器 ， 自定义程序 ，便可以在添加自己的启动图标。

在线帮助系统

您可以使用命令 man 或者 info 来阅读 Linux 命令的在线文档。命令的格式非常简单：

```
man xxx
```

大部分命令手册为英文版，如果您的英文不太好，或许有些困难。在后面的 系

统管理 章节中，我们会尽力向您介绍命令的使用方法。

Tip：在使用“man”浏览器的时候，一些快捷键您可能会用到：

Ctrl+f(orward)	向下翻一页	Ctrl+d(own)	向下翻半页
Ctrl+b(ackward)	向上翻一页	Ctrl+u(p)	向上翻半页
/	查找	q(uit)	退出

以上为 VI 风格的键绑定。您也可以使用 Emacs 风格的键绑定

bash

好了，现在我们换了一个遥控器，感觉顺手多了。现在来操练一下，下载一首 mp3：

我们使用 wget 这个程序，它非常可靠，完全值得您信赖。

首先找到一个可以下载的地址，复制链接，在终端窗口内点击鼠标中键，把它粘贴进去。

现在终端中大概是这种情形：

```
http://www.download.net/xxx.mp3
```

按下 Ctrl+a 组合键，我们发现光标移动到了行首。输入 wget 和 空格

```
wget http://www.download.net/xxx.mp3
```

回车后，终端中出现一些信息，不一会儿工夫，mp3 便下载完成。

使用 Ctrl+a 组合键，我们就不需要使用方向键来移动光标，方向键每次只能移动一个字符，没有效率

您还可以使用 Ctrl+f 向前移动光标， Ctrl+b 向后移动光标， Ctrl+e 将光标移动到行末…… （键绑定）

Note

Linux 的图形界面中，鼠标中键通常执行“粘贴”的操作，如果您的鼠标没有中键，您可以左右键同时按下。

中止正在运行的程序

如果一个命令持续时间很长，以致于不能够进行其它操作，可以使用 Ctrl+c 来强行中止它。

Ctrl+s

出于意外，有时您会按下 Ctrl+s 这个组合键，Shell 便被冻结。尝试使用 Ctrl+q 组合键，看能否恢复正常。

键绑定

等等，有必要记这么多快捷键么？都这么复杂！

我们强烈建议您记住，以大幅度的提高操作效率。而且这是 readline 控件的键绑定，在任何使用 readline 控件的程序中，您都可以使用它 们。例如 bash、lftp、gdb 等程序;同时，Linux 下最著名的 Emacs 编辑器，也是这种风格的键绑定（其实是 readline 使用了 Emacs 风格的键绑定才对），甚至 FireFox 中，也可以使用类似风格的快捷键！（Linux 下主要有两种风格的键绑定，一种是 VI 风格，另一种是 Emacs 风格，我们会在 简明 VIM 教程中介绍）

现在列举一些 ReadLine 的键绑定，您可以自行尝试。（运行 man readline 命令，来查看 ReadLine 手册）

先来了解一些约定：

\C-a 表示 Ctrl+a
\M-a 表示 Meta+a Meta 键在 PC 中通常为 ALT 键
A 表示 Shift+a
（下面括号中的\A 代表 Alt，\S 代表 Shift）

移动命令：

\C-a	移动到行首	Aheah
\C-e	移动到行末	End
\C-f	向前移动一个字符	Forward
\C-b	向后移动一个字符	Backward
\M-f	向前移动一个单词	
\M-b	向后移动一个单词	
\C-l	清空屏幕	cLear

这两个命令也可以理解为移动命令

\C-p	上翻，前一条命令	Previous

\C-n　　下翻，后一条命令　　Next

编辑命令：

\C-d	删除光标后的一个字符	\M-d	删除光标后的一个单词	Delete
\BackSpace	删除光标前的一个字符	\M-BackSpace	删除光标前的一个单词	
\C-k	删除光标至行末的部分			Kill
\C-u	删除光标至行首的部分			Unix-line-discard
\C-w	删除光标前的一个单词			Word
\C-y	粘贴（最后删除的对象）			Yank
\C--	撤消			

搜索历史纪录：

\C-r　　连续使用 ``C-r`` 可以查找下一个
\M-p
\M-n

补全：

\Tab　　使用频率最高的功能！
\C-o　　遍历补全 （未定义）
\M-?　　M-=　　列出所有可能选项，相当于按两次 Tab 键（M-?　实际按键为\A+\S+/）
\M-#　　注释掉当前命令，用于将当前命令暂存于历史纪录列表（\A+\S+3）
\M-!　　补全命令，通常用来补全子命令，例如 ``sudo`` 的子命令（\A+\S+1）
\M-~　　补全用户名（\A+\S+`）
\M-@　　补全主机名（\A+\S+2）
\M-$　　补全变量（\A+\S+4）
\M-_　　补全历史纪录中的纪录（\A+\S+-）
\M-*　　将所有可能选项放到命令行中（\A+\S+8）

自定义键绑定

通过修改 /etc/inputrc 文件，可以更改键绑定。建议您使用默认的键绑定，以避免不必要的烦恼。当然了，Emacs 风格的键绑定是通用的，随时都有可能用到。

在文件中添加该行，可以将 ReadLine 的键绑定设为 VI 风格。（Bash、Lftp 等使用 ReadLine 的软件同时生效）

```
set editing-mode vi
```

找到这一行：

```
$if mode=emacs
```

在它的下面添加如下内容

```
"\C-o": menu-complete

###这两行不是必须的，视情况而定###
"\c-p": non-incremental-reverse-search-history
"\c-n": non-incremental-forward-search-history
```

重新登录 Shell，您就可以使用 \C-o （Ctrl+o）来遍历补全。假如您的文件名为中文，或者出现乱码时，您可以使用 \M-* 将所有文件名放入命令行，再删除多余的，这真是麻烦极了！所以您可以使用 \C-o 遍历补全，将所有可能的选项轮流放入命令行。

或者使用 Vim 编辑器编辑 /etc/inputrc 文件，在插入模式下使用 Ctrl+v 组合键。按下 Ctrl+o ，这时编辑区新增一个 ^O 字符，等价于 \C-o

通配符

使用 ? 代表任意单个字符。例如 ???lo ，表示 lo 前有三个字符，它可以匹配 Hello

使用 * 代表随意几个任意字符。例如 *.iso ，代表所有 iso 格式的文件。

Tip：您可以将遍历补全和通配符结合使用，以提高效率。

例如：

```
cd */   则遍历补全只补全文件夹
chmview *.chm 则遍历补全只补全 chm 文件
```

任务管理

&

在命令的末尾加上一个 & 符号，表示背景任务，例如：

```
wget http://www.download.net/xxx/mp3 &
```

;

使用 ；将多个命令连结起来，则表示任务按顺序执行

&&

使用 && 将多个命令连结起来，则表示只有前面的命令执行成功，后面的命令才能得以执行

``

`<命令>` ，如果一个命令中包含以 `` （Esc 键下方的按键）括起来的子命令，那么子命令将被优先执行，执行结果被代入上一级命令继续执行，例如创建一个以当前时间命名的文件：

```
touch `date +%m.%d_%H:%M:%S`
```

touch 命令能够创建一个文件，它的操作对象，为 date +%m%d%H%M%S 命令的输出 06.06_06:06:60

这样，我们创建了一个名为 06.06_06:06:60 的文件（六月六日六时六分刚过六十秒-_-!）

Ctrl+z

将当前 Shell 中的任务挂起

这个时候任务的状态为

```
[1]+  Stopped    xxx
```

bg

将挂起的任务背景运行。这时它的状态为

```
[1]+ xxx &
```

fg

将背景任务调到前台执行

jobs

方括号中的数字为命令的任务编号，您可以使用 jobs 命令来查看所有背景任务

如果后台运行多个任务，您可以在 bg 或者 fg 后跟任务编号，作为操作对象，例如：

```
bg 2
```

管道、重定向

>

重定向符号，它的作用是将命令的输出重定向到一个文件中。比如我们想把命令 ls 的结果保存为 FileList 文件，作一个清单，我们可以使用重定向符号来完成它：

```
ls -l > FileList
```

>>

作用与 > 基本相同，不同点在于， >> 以追加的方式，将命令的输出写入文件的末尾。

<

是从文件到命令的重定向，将文件的内容作为命令的输入。

|

为管道符号，它的作用是将前一个命令的输出，作为下一个命令的输入。假设一个目录下的文件太多，使用 ls 命令不能够在屏幕中完全显示，这个时候您可以将 ls 命令的输出，通过管道符号，作为浏览器 less 的输入。就可以使用浏览器的功能翻页、查找：

```
ls -al | less
```

Tip: less 浏览器的键绑定几乎与 man 相同，请参阅 在线帮助系统

脱字符

Shell 中的一些功能是通过特殊符号作为控制字符来实现的，上面已经介绍了很多了。这产生一个问题，如果一个文件名中，刚好包含了这些字符，比如 ；，就很难对它进行操作。使用 less 浏览这个文件

```
less ;xxx
```

less 会很快返回一个错误信息，因为并没有一个文件名作为操作对象。接着，Shell 会报告，系统中没有 xxx 这个命令。

这是因为 Shell 将文件名中的 ；解析为按顺序执行命令。

或者您的文件名以空白起始，而在 Shell 中，无论多少个空格，都将被解析为一个分隔符。您甚至不是使用命令重命名此文件。

这个时候就要用到脱字符 \ 了，它能够将一个具有特殊涵义的字符转换普通字符。上面的两个任务，可以在文件名中每个特殊字符前加一个 \ ，像这样

```
less \;xxx
less \ \xxx
less \;\ \&\xxx
```

Tip

也可以使用 " 将文件名括起来，例如 less "; &xxx" ，在很多情况下，这样甚至更方便。

脱字符在 Shell 中也可以作为换行符，在一个命令的末尾添加一个 \ ，然后回车，在下一行继续输入命令剩余的部分，将一个命令拆分为多行且不影响它的执行（如果执行一个很长的命令，请将它拆分为多行以便于阅读）

事实上换行符也符合脱字符的定义。回车键有两个涵义，一个是 *执行*（Enter），另一个 *换行* （折线箭头）。在 Shell 中它作为控制字符 *执行* ，使用脱字符后，它便代表排版字符 *换行* 了。

Fish

the friendly interactive shell

正如它的名字，Fish 是一款非常友好的 Shell，大力推荐！使用命令 sudo apt-get install fish 安装它。完成后，运行命令 fish 切换到 fish， exit 返回 bash。

简单介绍一下它的优点：

1. 自动补全、语法高亮

   bash 的自动补全默认只是补全命令、路径，如果想补全变量、参数等，通常需要使用复杂的组合键（见上面 bash 的介绍），即便您能够记住它们，快 速准确的按下这些组合键，也是一种严峻的考验。而 FISH 的自动补全可以自动识别语法，补全正确的内容。并且具有语法高亮的功能，比如用 MPLAYER 放 MP3：

   ```
   mp1<tab>(ayer) -1<tab>(oop) <tab>(0) -sh<tab>(uff1e) -p1<tab>
   (ay1ist) <tab>(mp3_p1ay1ist)
   ```

   一阵猛按 <tab> 键，一个蛮长的命令就完成了。

补全结果不唯一时给出的提示中含有简短的说明，这样通常也不用看帮助了：）比如：

```
mplayer -l
```

<tab> 后，自动将参数补全为 -lo 然后给出提示

```
-{lo}adidx (Load index from file) -{lo}op (Loop playback) {花括号中为青色文字}
```

它的语法高亮功能十分有用，如果你输入的命令是正确的，则用青色显示，正确的参数用白色显示，错误的则一律用红色。

2. 方便的历史纪录搜索

还是上面的那个命令

```
mplayer -loop 0 -shuffle -playlist mp3_playlist
```

用上翻配合下翻浏览命令历史，直到找到这个命令，当然那样太慢了。

还可以输入以上命令中的某一部分，如 uffle 只要翻一次就可以找到了

（还可以 META＋上翻在已输入部分中插入某一历史单词）

3. 文件夹历史纪录

dirh （dir history）就可以显示当前会话中进入的文件夹纪录

使用 prevd 和 nextd 跳转

假如曾进入过 1 2 3 4 5 这几个文件夹， prevd 4 可以让你在 5 中直接跳到 1

4. 其它的功能，fish 基本是兼容 bash 的。键绑定也非常的相似，少数的键绑定不尽一致，例如：

\C-h 删除光标前的一个字符（bash 为退格键，不方便）

修改 /etc/fish_inputrc 这个文件，增加以下行：

```
"\C-n": history-search-forward
"\C-p": history-search-backward
```

现在使用 Ctrl+p 上翻，使用 Ctrl+n 下翻。如果已经在命令行中输入字符，那么 Ctrl+p 就是在历史纪录向上查找您输入的字符，Ctrl+n 为向下查找，非常的方便。

设定您的默认 Shell

如果能够拥有 root 权限，可以直接修改 /etc/passwd 文件。找到您用户 ID 起始的行

```
user:x:1000:112:user,,,:/home/user:/bin/bash
```

最后一个字段为登录后的默认 Shell， /bin/bash 是程序 bash 的主程序路径。 fish 主程序的路径通常为 /usr/bin/fish 。

/etc/shells 中列出系统中所有可用 Shell（ /bin/false 代表禁用 Shell）

也可以使用如下命令更改您的默认 Shell

```
chsh -s /usr/bin/fish
（需要输入您的密码）
```

Tip: 可以使用 whereis xxx 命令，来查找 xxx 程序的安装位置，详见 搜索

设定命令的搜索路径

使用 echo $PATH ，可以显示 $PATH 变量，输出如下：

```
/usr/local/sbin /usr/local/bin /usr/sbin /usr/bin /sbin /bin /usr/bin/X11 /usr/games
/usr/X11R6/bin
```

它是一个环境变量，代表执行命令时，Shell 的搜索路径。

执行一个命令时，Shell 会到 $PATH 变量定义的路径去搜索，并运行与命令同名的可执行文件。如果程序、脚本等可执行文件并不在上面的路径中，就必须使用绝对路径或者相对路径定位可执行文件。

例如：

```
/usr/local/mplayer -menu xxx.rmvb
/etc/init.d/powernowd start
cd /usr/local/   &&  ./mplayer -menu xxx.rmvb
```

可以修改 /etc/environment 文件来设定您的命令搜索路径，找到 PATH 起始的行

```
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin"
```

在双引号中添加您的自定义路径，并以 ： 分隔。

Ubuntu 系统简介

Ubuntu 系统目录结构

以下为 Ubuntu 目录的主要目录结构，您稍微了解它们都包含了哪些文件就可以了，不需要记忆。

```
/    根目录
|
├boot/              启动文件。所有与系统启动有关的文件都保存在这里
|     └grub/        Grub 引导器相关的文件
|
├dev/               设备文件
├proc/              内核与进程镜像
|
├mnt/               临时挂载
├media/             挂载媒体设备
|
├root/              root 用户的$HOME 目录
├home/
|     ├user/        普通用户的$HOME 目录
|     └…/
|
├bin/               系统程序
├sbin/              管理员系统程序
├lib/               系统程序库文件
├etc/               系统程序和大部分应用程序的全局配置文件
|    ├init.d/       SystemV 风格的启动脚本
|    ├rcX.d/        启动脚本的链接，定义运行级别
|    ├network/      网络配置文件
|    ├X11/          图形界面配置文件
|
├usr/
|    ├bin/          应用程序
|    ├sbin/         管理员应用程序
|    ├lib/          应用程序库文件
|    ├share/        应用程序资源文件
```

```
|    ├src/              应用程序源代码
|    ├local/
|    |      ├soft/     用户程序
|    |      └…/        通常使用单独文件夹
|    ├X11R6/           图形界面系统
|
├var/                  动态数据
|
├temp/                 临时文件
├lost+found/           磁盘修复文件
```

启动流程

Linux 系统主要通过以下步骤启动：

1. 读取 MBR 的信息，启动 Boot Manager

   Windows 使用 NTLDR 作为 Boot Manager，如果您的系统中安装多个版本的 Windows，您就需要在 NTLDR 中选择您要进入的系统。
   Linux 通常使用功能强大，配置灵活的 GRUB 作为 Boot Manager，我们将在启动管理章节中向您介绍它的使用方式。

2. 加载系统内核，启动 init 进程

   init 进程是 Linux 的根进程，所有的系统进程都是它的子进程。

3. init 进程读取 /etc/inittab 文件中的信息，并进入预设的运行级别，按顺序运行该运行级别对应文件夹下的脚本。脚本通常以 start 参数启动，并指向一个系统中的程序。

   通常情况下， /etc/rcS.d/ 目录下的启动脚本首先被执行，然后是 /etc/rcN.d/ 目录。例如您设定的运行级别为 3,那么它对应的启动目录为 /etc/rc3.d/ 。

4. 根据 /etc/rcS.d/ 文件夹中对应的脚本启动 Xwindow 服务器 xorg

   Xwindow 为 Linux 下的图形用户界面系统。

5. 启动登录管理器，等待用户登录

   Ubuntu 系统默认使用 GDM 作为登录管理器，您在登录管理器界面中输入用户名和密码后，便可以登录系统。（您可以在 /etc/rc3.d/ 文件夹中找到一个名为 S13gdm 的链接）

更改运行级别

在 /etc/inittab 文件中找到如下内容：

```
# The default runlevel.
id:2:initdefault:
```

这一行中的数字 2 ,为系统的运行级别，默认的运行级别涵义如下：

```
0    关机
1    单用户维护模式
2~5 多用户模式
6    重启
```

服务管理

更改启动服务

在运行级别对应的文件夹中，您可以看到许多文件名以 S## 和 K## 起始的启动脚本链接。 例如：

```
/etc/rcS.d/S35mountall.sh    挂载文件系统
/etc/rcS.d/S40networking     启用网络支持
/etc/rc2.d/S13gdm            启动登录管理器
/etc/rc2.d/S20makedev        创建设备文件
/etc/rc2.d/S23xinetd         启动超级进程
```

init 进程将以 start 为参数，按文件名顺序执行所有以 S## 起始的脚本。脚本名称中的数字越小，它将被越早执行。例如在 /etc/rc2.d/ 文件夹中，S13gdm 文件名中的数字小于 S23xinetd ，S13gdm 将比 S23xinetd 先执行。

如果一个脚本链接，以 K## 起始，表示它将以 stop 参数被执行。如果相应服务没有启动，则不执行该脚本。 例如：

```
/etc/rc2.d/K20powernowd      针对某种硬件的电源管理支持
```

如果您想禁止某一服务在启动时自动运行，您可以将相应运行级别中的脚本由 S##xxx 重命名为 K##xxx 。

手动控制服务

您也可以手动运行带有以下参数的启动脚本，来控制系统服务。 - start 启动 - stop 停止 - restart 重启

例如：

/etc/rc2.d/K20powernowd start

有时您并不清楚当前运行级别，该运行级别下未必有相应脚本；而且此类脚本的前三位字符并不固定，不便于记忆。这时，可以直接使用 /etc/init.d/ 文件夹中的启动脚本（ /etc/rcX.d/ 中的启动脚本链接到 /etc/init.d/ 文件夹下相应脚本），这也是推荐的方式。

例如：

/etc/init.d/powernowd start

Note：以上命令的位置并没有包含在环境变量的搜索路径中，所以要输入完整路径。

常用系统服务

acpi-support 高级电源管理支持
acpid acpi 守护程序.这两个用于电源管理，非常重要
alsa 声音子系统
alsa-utils
anacron cron 的子系统，将系统关闭期间的计划任务，在下一次系统运行时执行。
apmd acpi 的扩展
atd 类似于 cron 的任务调度系统。建议关闭
binfmt-support 核心支持其他二进制的文件格式。建议开启
bluez-utiles 蓝牙设备支持
bootlogd 启动日志。开启它
cron 任务调度系统，建议开启
cupsys 打印机子系统。
dbus 消息总线系统(message bus system)。非常重要
dns-clean 使用拨号连接时，清除 dns 信息。
evms 企业卷管理系统（Enterprise Volumn Management system）
fetchmail 邮件用户代理守护进程，用于收取邮件
gdm gnome 登录和桌面管理器。
gdomap
gpm 终端中的鼠标支持。
halt 别动它。
hdparm 调整硬盘的脚本，配置文件为 /etc/hdparm.conf。
hibernate 系统休眠

hotkey-setup 笔记本功能键支持。支持类型包括： HP，Acer，ASUS，Sony，Dell，和 IBM。
hotplug and hotplug-net 即插即用支持，比较复杂，建议不要动它。
hplip HP 打印机和图形子系统
ifrename 网络接口重命名脚本。如果您有十块网卡，您应该开启它
inetd 在文件 /etc/inetd.conf 中，注释掉所有你不需要的服务。如果该文件不包含任何服务，那关闭它是很安全的。
klogd 重要。
linux-restricted-modules-common 受限模块支持。 /lib/linux-restricted-modules/ 文件夹中的模块为受限模块。例如某些驱动程序，如果您没有使用受限模块，就不需要开启它。
lvm 逻辑卷管理系统支持。
makedev 创建设备文件，非常重要。
mdamd 磁盘阵列
module-init-tools 从/etc/modules 加载扩展模块，建议开启。
networking 网络支持。按 /etc/network/interfaces 文件预设激活网络，非常重要。
ntpdate 时间同步服务，建议关闭。
pcmcia pcmcia 设备支持。
powernowd 移动 CPU 节能支持
ppp and ppp-dns 拨号连接
readahead 预加载库文件。
reboot 别动它。
resolvconf 自动配置 DNS
rmnologin 清除 nologin
rsync rsync 守护程序
sendsigs 在重启和关机期间发送信号
single 激活单用户模式
ssh ssh 守护程序。建议开启
stop-bootlogd 在 2，3，4，5 运行级别中停止 bootlogd 服务
sudo 检查 sudo 状态。重要
sysklogd 系统日志
udev & udev-mab 用户空间 dev 文件系统（userspace dev filesystem）。重要
umountfs 卸载文件系统
urandom 随机数生成器
usplash 开机画面支持
vbesave 显卡 BIOS 配置工具。保存显卡的状态
xorg-common 设置 X 服务 ICE socket。
adjtimex 调整核心时钟的工具
dirmngr 证书列表管理工具,和 gnupg 一起工作。

hwtools irqs优化工具
libpam-devperm 系统崩溃之后，用于修理设备文件许可的守护程序。
lm-sensors 板载传感器支持
mdadm-raid 磁盘陈列管理器
screen-cleanup 清除开机屏幕的脚本
xinetd 管理其他守护进程的一个inetd超级守护程序

重要配置文件

！无论任何情况下，修改配置文件之前，先备份它！

建议使用这个命令： sudo cp xxx xxx_`date +%y%m%d_%H:%M` 。

当然这很麻烦，您可以新建一个名为 bak 的文件，内容如下：

```
#!/bin/bash
sudo cp $1  $1_`date +%y%m%d_%H:%M`
```

把它放在您能够记住的目录下，比如 /home ，执行命令 sh /home/bak xxx ，就可以将当前文件夹下的文件 xxx 另存为 xxx_yymmdd_HH:MM 的格式了

全局配置文件

系统初始化

/etc/inittab 运行级别、控制台数量
/etc/timezone 时区
/etc/inetd.conf 超级进程

文件系统

/etc/fstab 开机时挂载的文件系统
/etc/mtab 当前挂载的文件系统

用户系统

/etc/passwd 用户信息
/etc/shadow 用户密码
/etc/group 群组信息
/etc/gshadow 群组密码
/etc/sudoers Sudoer列表（请使用“visudo”命令修改此文件，而不要直接编辑）

Shell

/etc/shell 可用 Shell 列表
/etc/inputrc ReadLine 控件设定
/etc/profile 用户首选项
/etc/bash.bashrc bash 配置文件

系统环境

/etc/environment 环境变量
/etc/updatedb.conf 文件检索数据库配置信息
/etc/issue 发行信息
/etc/issue.net
/etc/screenrc 屏幕设定

网络

/etc/iftab 网卡 MAC 地址绑定
/etc/hosts 主机列表
/etc/hostname 主机名
/etc/resolv.conf 域名解析服务器地址
/etc/network/interfaces 网卡配置文件

用户配置文件

/etc/ 目录下的文件，只有 root 用户才有权修改。应用软件的全局配置文件，通常普通用户也不能够修改，如果要通过配置软件，来适应特殊需求，您可以修改用户配置文件。

用户配置文件通常为全局配置文件的同名隐藏文件，放在$HOME 目录下，例如：

/etc/inputrc	/home/user/.inputrc
/etc/vim/vimrc	/home/user/.vim/vimrc

也有少数例外，通常是系统程序

/etc/bash.bashrc	/home/user/.bashrc

软件安装

DPKG

Linux 系统中，软件通常以源代码或者预编译包的形式提供。

软件源代码需要编译为二进制的机器代码才能够使用，安装比较耗时，不过您可以自行调节编译选项，决定需要的功能或组件，或者针对硬件平台作一些优化。

预编译的软件包，通常是由软件的发布者进行编译，您只要将软件拷贝到系统中就可以了。考虑到预编译软件包的适用性，预编译软件包通常不会针对某种硬件平台优化。它所包含的功能和组件也是通用的组合。

Ubuntu 系统中，软件通常以 deb 格式的包文件发布，它是一种预编译软件包。deb 包中除了包含已编译的软件，通常还包括软件的拷贝路径、对其它软件包的依赖关系纪录、比较通用的配置文件以及软件的描述、版本、作者、类别、占用空间等信息。

deb 软件包命令遵行如下约定：

soft_ver-rev_arch.deb

soft

软件包名称

ver

软件版本号

rev

Ubuntu 修订版本号

arch

目标架构名称

例如： azureus_2.4.0.2-0ubuntu2_a11.deb

您需要使用 dpkg 命令来管理 deb 软件包：

```
dpkg -i | --install xxx.deb    安装 deb 软件包
dpkg -r | --remove xxx.deb     删除软件包
dpkg -r -P | --purge xxx.deb   连同配置文件一起删除
dpkg -I | -info xxx.deb        查看软件包信息
dpkg -L xxx.deb                查看包内文件
dpkg -l                        查看系统中已安装软件包信息
dpkg-reconfigure xxx           重新配置软件包
```

有些时候，您使用 dpkg 安装一个软件包，系统会提示您该软件包依赖其它软件包。这时，您先安装其它软件包，直到满足依赖关系为止。或者同时安装多个软件包

```
dpkg -i aaa.deb bbb.deb ccc.deb
```

APT

如果一个软件依赖关系过于复杂，使用 *dpkg* 来安装它，并不是一个明智的选择，这个时候您就需要用到 APT 软件包管理系统。APT 可以自动的检查依赖关系，通过您预设的方式来获得相关软件包，并自动安装配置它。事实上，在多数情况下，我们推荐您使用 APT 软件包管理系统。

APT 系统需要一个软件信息数据库和至少一个存放着大量 deb 包的软件仓库，我们称之为 **源** 。 *源* 可以是网络服务器，安装 CD 或者本地软件仓库。您需要修改 /etc/apt/sources.list 文件，使 APT 系统能够连接到 *源*。

从以下页面中获得网络安装源的列表，并且根据您的网络环境，选择速度较快的源。

http://wiki.ubuntu.org.cn/%E5%BF%AB%E9%80%9F%E8%AE%BE%E7%BD%AE%E6%8C%87%E5%8D%97/DapperDrake

APT 系统主要包括 apt-get 和 apt-cache 等命令。通常是复合命令，包含若干个子命令。

```
apt-get install xxx      安装 xxx
        -d 仅下载
        -f 强制安装
apt-get remove xxx       卸载 xxx
apt-get update           更新软件信息数据库
apt-get upgrade          进行系统升级
apt-cache search         搜索软件包
```

Tip: 建议您经常使用 sudo apt-get update 命令来更新您的软件信息数据库

APT 系统修复

由于各种意外，APT 系统可能会出现问题，使用如下命令，尝试进行修复：

```
apt-get -f install
```

源码包

对于绝大多数软件，我们建议您使用 APT 系统来安装它。在少数情况下，例如某软件没有以 deb 包的格式发布，或者需要定制适合自己的软件，您可以通过编译源代码的方式安装它。

首先需要下载软件的源码包，并且将它解包为一些源代码文件。并了便于管理，建议将下载的源码包移动到 /usr/local/src/ 目录下，并在这里解包。

```
sudo mv xxx.tar.gz /usr/local/src        移动源码包
cd /usr/local/src                         进入“/usr/local/src/”目录
sudo tar -xzvf xxx.tar.gz                 解包源码
cd xxx_ver/                               进行解包后的源码目录
```

源码目录中通常有一个 configure 脚本，用来配置即将开始的编译过程。您可以执行它

```
sudo ./configure [--prefix=/usr/loca/xxx ……]
```

它会自动检测软件的编译环境和依赖关系，并且生成 Makefile 文件。

> 使用带参数的命令 ./configure --help ，或者阅读 INSTALL 文件，查看该脚本允许的参数。例如使用 --prefix=/usr/local/xxx 参数，将软件的安装目录设定为 /usr/local/xxx/ 。（如果一定要将软件安装在单独目录下，建议您安装在这里）

现在执行 make 命令，系统会根据 Makefile 文件中的设定，通过 make 工具调用编译器和所需资源文件，将源代码编译成目标文件。

```
sudo make
```

执行 make install 命令， make 工具会自动连接目标文件和库文件，将最终生成的文件拷贝到 Makefile 文件设定的路径中，并且完成更改文件的属性，删除残留文件等活动。

```
sudo make install
```

现在，编译安装已经完成，为了更方便的使用它，需要给程序的可执行文件作

一个符号链接。

```
sudo ln -sf /usr/local/xxx/可执行文件 /usr/local/bin/可执行文件
```

Tip:为了顺利的进行编译，至少需要安装 build-essential 软件包。

```
sudo apt-get install build-essential
```

Xwindow 简介

Xwindow 是工作站图形系统的工业标准，它有多种不同的实现，Ubuntu 系统中使用的为 Xorg。

（比较前卫的图形界面系统 XGL，实际代替 X 服务器的作用，另外还有与之配套的窗口管理器）

历史

当然，Xwindow 有悠久的历史和传统，不过那不在我们讨论的范围。您要注意的有两点：

- Xwindow 和 Xbox 中的“X”本意是不同的， X 只是 W 后的一个字母，差不多应该这样理解，Xwindow 是 Window 的接班人 （注意，Window 不是 Windows）
- 同样，也不要把 Xwindow 说成是 Xwindow**s**，那是一种亵渎！一切伟大的创造，都应得到应有的理解和尊重。

架构及原理

Xwindow 使用服务器－客户端架构。无论本地图形界面，还是远程图形界面，都以同样的流程工作。这样便不需要分别进行设计和维护，极大的提高了网络透明性。

```
本地 X 客户端 ┐                          ┌ 键盘
远程 X 客户端 ┼ X 协议 ― X 服务器 ― 硬件规范 ┼ 鼠标
远程 X 客户端 ┘                          └ 显示器
```

Xserver

Xwindow 系统服务器端，通过驱动程序（硬件规范）来管理硬件资源。

例如：当我们移动鼠标时，通过驱动程序 [5] ，向 Xserver 发送信息：

“向右移动 200 点，向上移动 100 点”（向右上移动）；“按下左键”……

Xserver 作出如下响应：

1、上一次鼠标停止的坐标为 600,500

2、向右 200，向上 100。现在鼠标位于坐标 800,600
3、坐标 800,600 处，为窗口 Firefox 的“关闭”按钮
4、根据预设动作，将 “点击 Firefox 窗口的关闭按钮” 翻译为 “关闭窗口 Firefox”
5、向 X 客户端 Firefox 发送一个“退出”消息
6、Xserver 通过显示子系统（显卡、显示器），全程显示鼠标的位置和移动

- 事实上，向程序发送“退出”信号，通常窗口管理器完成……为了描述方便，这里暂不区分。稍后，我们将向您介绍 窗口管理器 的其它一些细节。

[5] 大多数的鼠标不需要专门的驱动程序，因为它们符合某一硬件规范，例如：有四个移动方向和三个键

Xclient

Xwindow 系统客户端，通过 X 协议，实现与 Xserver 的交互。

例如：

1、Xclient （假设 Firefox） 接收 Xserver 的消息： 输入焦点在地址栏的范围内， “ubuntu.org.cn” ， 回车

2、Firefox 根据预设动作，将这些消息识别为 “打开链接 ubuntu.org.cn”
3、Firefox 向域名服务器请求 链接 “ubuntu.org.cn” 。域名服务器将这个请求转换为 “http://ubuntu.org.cn/” 和 IP 地址 211.148.131.7，发送回 Firefox
4、Firefox 将 “http://ubuntu.org.cn/” 显示在地址栏（向 Xserver 发送请求，在地址栏位置显示这个地址）
5、Firefox 向地址 211.148.131.7 请示显示页面。
6、Firefox 将服务器发送回的页面显示在主窗口中

Xprotocol

Xwindow 系统协议，Xserver 和 Xclient 之间进行通信的规则

窗口管理器

Window Manager，一种特殊的 Xclient。

使用窗口管理器时，Xserver 并不直接与其它 Xclient 通信，而是通过 WM 中转，当一些消息被定义为 WM 指令时，它们会被拦截。例如 A1t+F4 关闭窗口、拖动标题栏……

消息“打开链接 ubuntu.org.cn”，具体内容如下：

输入焦点在地址栏的范围内，“ubuntu.org.cn”，回车

Xserver 并不能直接判断焦点，而是这样：

1、Xserver 向 WM 发送位置和点击的信息，WM 根据当前的“焦点策略”确定激活（最上层）的窗口为 Firefox

2、Xserver 将 Firefox 显示在最上层，高亮显示它的标题栏
3、在窗口 Firefox 内点击地址栏，或者 Ctrl+L，Xserver 将位置信息发送给 WM，WM 再发送给 Firefox
4、Firefox 判断当前焦点后，显示一个闪动的文字输入光标
5、Firefox 将输入光标通过 WM 发送给 Xserver，Xserver 在屏幕相应位置进行显示

那么，“窗口管理器”到底能作些什么呢？其实它所作的一切都是管理窗口。例如：

1. 最上层的窗口会把其它窗口挡住
2. 它通常是一个“已激活窗口”，根据不同的“焦点策略”，窗口管理器确定被激活的窗口。

   激活窗口标题栏高亮显示，接收大部分的键盘消息和窗口内的鼠标点击消息。

3. 为了美观和容易分辨，大多数窗口都要有标题栏和边框。

   为了方便，标题栏上还要有一些按钮，比如：最小化，最大化，关闭（这些按钮是窗口管理器请求的小窗口）

4. 一个窗口可以在另一个窗口旁边显示，而不一定完全被遮挡。为了实现这一点，就要控制窗口显示的位置
5. 为了控制窗口的显示位置，需要将整个屏幕用座标描述，最好的办法是绘制一个填充整个屏幕的窗口，也就是根窗口。

6. 因为根窗口是最大的，所以它可以严严实实的遮挡任何窗口，为了避免这一点，根窗口永远在最底层。

   这很形象的说明了为什么它叫作“根窗口” ……root

7. 根窗口不一定只有一个，大多数的窗口管理器可以使用 “工作区” ，来切换显示多个根窗口
8. 根窗口固定位置上通常放置一些其它 Xclient 的窗口，例如底部面板，顶部面板，侧面板，程序启动图标
9. 面板上又可以放一些其它的 Xclient 窗口，如任务条，启动栏，菜单……

   任务条可以以图标显示正在运行的任务，还可以作其它的杂活，像自动挂载 USB 设备……

启动流程

我们知道 init 是 linux 的根进程，是所有进程的父进程。同样， xinit 是所有 Xwindow 进程的根进程

Startx

startx 命令可以在命令行下启动图形界面。执行 startx 命令时，实际执行这一命令：

```
xinit /etc/X11/xinit/xinitrc -- /etc/X11/xinit/xserverrc
```

根据脚本 /etc/X11/xinit/xserverrc 启动 Xserver，同时根据脚本 /etc/X11/xinit/xinitrc 启动指定 Xclient 进程，例如窗口管理器

脚本 /etc/X11/xinit/xserverrc 以预设的参数运行程序 /usr/bin/X11/X

/etc/X11/xinit/xinitrc 脚本则指向 /etc/X11/Xsession ，依次启动 /etc/X11/Xsession.d 目录中的脚本

- 您可以在用户配置文件 **~/.Xsession** 中定义使用的 WM，它的优先级高于全局配置文件(对于 GDM 会话不起作用)
- startx 启动时，并不会再进行身份认证。因为它启动的是 */etc/X11/Xsession.d/gnome-session* ，而不是 GDM 会话

GDM 会话

Ubuntu 系统启动时自动进入图形界面，不需要运行 *startx* 命令

在某些启动级别中，包含了 gdm 的启动脚本，例如 ：/etc/rc2.d/S13gdm

1. 指向 /etc/gdm/gdm-cdd.conf 文件，加载预设视觉主题，启动 /usr/lib/gdm/gdmgreeter（登录屏幕）

2. 用户身份认证完成后，启动 /etc/X11/default-display-manager 这个文件中设定的默认窗口管理器 /usr/sbin/gdm

gdm 在启动时，会要求用户名和密码，也就是我们看到的登录屏幕（*gdmgreeter*）

- **/usr/share/xsessions** 目录下为所有可用登录会话的脚本

配置文件

X 服务器

X 服务器的主要配置文件为 **/etc/X11/xorg.conf**

布局

```
Section "ServerLayout"
Identifier     "Default Layout"
Screen          "Default Screen" 0 0
InputDevice    "Generic Keyboard"
InputDevice    "Configured Mouse"
EndSection
```

- 定义了 *布局标识* 、 *屏幕标识* 、 *键盘标识* 、 *鼠标标识*

模块

```
Section "Module"
    Load  "i2c"
    Load  "bitmap"
    Load  "ddc"
    Load  "dri"
```

```
    Load  "extmod"
    Load  "freetype"
    Load  "glx"
    Load  "int10"
    Load  "type1"
    Load  "vbe"
EndSection
```

X 核心字体路径

```
Section "Files"
        FontPath     "/usr/share/X11/fonts/75dpi"
        FontPath     "/usr/share/X11/fonts/100dpi"
        FontPath     "/usr/share/X11/fonts/misc"
        FontPath     "/usr/share/X11/fonts/cyrillic"
        FontPath     "/usr/share/X11/fonts/100dpi/:unscaled"
        FontPath     "/usr/share/X11/fonts/75dpi/:unscaled"
        FontPath     "/usr/share/X11/fonts/Type1"
        FontPath     "/usr/share/fonts/Chinese/wqy-bitmapfont"
EndSection
```

屏幕

```
Section "Screen"
    Identifier "Default Screen"
    Device     "ATI Technologies, Inc. RV370 5B62 [Radeon X600 (PCIE)]"
    Monitor    "DELL E176FP"
    DefaultDepth     24
    SubSection "Display"
      Depth     1
      Modes    "1280x1024" "1152x864" "1024x768" "800x600" "720x400" "640x480"
      ................
    EndSubSection
EndSection
```

- DefaultDepth 24 默认色深
- SubSection 可用色深及分辨率

显卡

```
Section "Device"
   Identifier  "ATI Technologies, Inc. RV370 5B62 [Radeon X600 (PCIE)]"
   Driver      "fglrx"
   Option       "KernelModuleParm" "agplock=0"
   VideoRam   131072
EndSection
```

- Identifier 显卡标识
- Driver 显卡驱动（如不同正常启用图形界面，首先尝试"vesa"）
- Option 显卡参数
- VideoRam 显存大小

显示器

```
Section "Device"
   Identifier  "ATI Technologies, Inc. RV370 5B62 [Radeon X600 (PCIE)]"
   Driver      "fglrx"
   Option        "KernelModuleParm" "agplock=0"
   VideoRam    131072
EndSection
```

配置文件内部结构

```
/
├/ "ServerLayout"                布局
| ├ "InputDevice" keyboard       键盘
| ├ "InputDevice" mouse          鼠标
| |
| └/ "Screen"                     显示子系统
|  ├ "Monitor"                   显示器
|  ├ "Device"  videocard         显卡
|
|
├ "Files"                        字体
└ "Module"                       模块
```

X 客户端

在 /etc/X11/Xsession 文件中可以发现下列内容

```
OPTIONFILE=/etc/X11/Xsession.options

SYSRESOURCES=/etc/X11/Xresources
USRRESOURCES=$HOME/.Xresources

SYSSESSIONDIR=/etc/X11/Xsession.d
USERXSESSION=$HOME/.xsession
ALTUSERXSESSION=$HOME/.Xsession
ERRFILE=$HOME/.xsession-errors
```

- OPTIONFILE=/etc/X11/Xsession.options 设定 X 进程的启动参数。例如允许用户进程 allow-user-xsession

- Xresources X 资源文件。许多程序保留了 X 接口，允许 X 服务器管理一些视觉选项，例如窗口内的字体，配色等
- xsession X 进程。可以设置一些启动时自动运行的程序，也可以用来设定自己的窗口管理器（窗口管理器和桌面环境或者登录管理器是无关的）

字体

freetype 渲染引擎

作为 Xorg 服务器的一个模块，freetype 的功能包括读取 Truetype 字体信息，如大小、分辨率、编码等，并以之为依据渲染字体 - freetype2.x 相对于 freetype1.x 增加了抗锯齿等功能 -（ /etc/X11/xorg/conf 的 *Module* 字段中，可以选择字体渲染模块，建议使用默认的 *freetype* ）

freetype 只负责渲染字体。而查找字体，则可以由 X 服务器、X 客户端或者字体服务器来完成。 找到字体后，使用 freetype 引擎就地渲染

X 核心字体

X 服务器根据 X 客户端的请求（字符编码），查找字体并进行渲染，然后显示，我们称之为

Xft 字体

X 客户端自行查找字体并进行渲染，X 服务器只负责显示

由于 Xft 字体的渲染在客户端完成，所以它可以动态的加载，而不需要随同 X 服务器一同启动

字体服务器

另外还有一种字体服务器模式，例如 *XFT 字体*：当客户端请求字体时，X 服务器将请求转发到字体服务器，由字体服务器查找字体，并使用 freetype 引擎渲染，将结果传回 X 服务器，X 服务器进行显示……

X 核心字体

/etc/X11/xorg.conf 中可以配置 X 核心字体的搜索路径

```
Section "Files"
```

```
FontPath "/usr/X11R6/lib/X11/fonts/misc/"
FontPath "/usr/X11R6/lib/X11/fonts/Type1/"
FontPath "/usr/X11R6/lib/X11/fonts/Speedo/"
FontPath "/usr/X11R6/lib/X11/fonts/100dpi/"
FontPath "/usr/X11R6/lib/X11/fonts/75dpi/"
EndSection
```

- 当 X 客户端向 X 服务器请求显示文字的时候，X 服务器会按上面列表的先后顺序查找字体

  例如显示中文时，如果第一个路径中的字体不包含中文，则查找下面的路径，直到发现中文字体

- 请将您偏好的字体放在靠前的位置

要使安装的字体能够作为 X 核心字体使用，将字体的安装路径添加到上面的列表中，使用 *mkfontscale* 、 *mkfontdir* 扫瞄文件夹中的字体，并生成索引，就可以了（建议使用 **ttmkfdir** 生成 *fonts.scale* ，将其复制为 *fonts.dir* ）

字体的选择及显示风格，可以修改 GTK1 的配置文件，或者在 Xresources 文件中对程序单独进行定义

事实上，在我们的日常应用中，X 核心字体环境并不常见，使用 GTK1 图形库的程序、某些类型的终端……

- Emacs 也是这样一个老派的程序……不过 Emacs23 中刚刚加入了 xft 字体的支持

XFT 字体

Xft 字体相关选项在 **/etc/fonts/fonts.conf** 文件中配置

可以使用 *fc-cache* 命令，递归扫瞄以下目录中的字体（包括子文件夹中的字体），建立字体缓存

```
/usr/share/X11/fonts
/usr/share/fonts
/usr/local/share/fonts
~/.fonts
```

- /etc/fonts/fonts.conf 文件的 *<dir>* 字段

多数支持 GTK2 或者 Qt 图形库的 X 客户端能够使用 Xft 字体渲染技术

- GTK2 为 Gnome 使用的图形库，Qt 为 KDE 使用的图形库。相对来说，GTK2

图形库在程序的 GUI 设计中更加通用

安装字体，只要将字体拷贝到以上任意目录， **fc-cache -fv** 刷新字体缓存即可 （参数: -f 强制刷新; -v 显示过程）

使用命令 *fc-list* 列出所有可用字体

字体的选择及显示风格，可以修改 GTK2 或者 Qt 的配置文件，建议使用图形界面配置

- 一般情况下，桌面环境中附带了相关程序，例如 *gnome-font-properties*

系统管理

一些细节

Linux 是大小写敏感的系统，所有的命令、路径、参数、变量……都区分大小写

使用 **TAB** 键补全命令，无论任何时候，多按几次 TAB 总会有所帮助

Shell 的功能键能够协助您更高效的编辑命令，请熟悉其键绑定，尽量使用它

命令由 **命令名** 、 **分隔符** 、 **参数** 、 **操作对象** 构成

命令名

标识命令的功能，例如 cp(copy)、mv(move)、rm(remove)……

有些命令包含一些子命令，您可以认为它的命令名由两个单词构成，例如“apt”软件包管理系统:

```
apt-get install          安装一个软件
apt-get remove           删除一个软件
```

分隔符

通常为空格，多个连续的空格视为一个空格，下面两个命令相

同：

```
cp a b
cp     a     b
```

有一些特殊符号也属于分隔符，例如管道 | 、重定向 > 、 >> 、 < 、后台运行 & 、序列执行 && 、 ; 。使用这些符号时，您不需要再使用空格作为分隔符，例如：

```
ls -al|less
```

写为以下形式，是为了让您更容易的阅读它：

```
ls -al | less
```

参数

精细调节命令的行为，以 - 引导，通常为参数名的首字母。许多软件都可以使用 -h 参数来阅读使用说明，例如：

```
apt-get -h
```

也可以使用参数的全名，一般以 -- 引导，例如：

```
apt-get --help
```

多数命令中，使用 - 引导多个字符，将会被视为多个参数，例如：

```
apt-get -help
```

系统会解读为以下命令

```
apt-get -h -e -l -p
```

少数命令的参数，不需要以 - 引导，或者使用 - 引导参数全名，例如：

```
ps aux
/etc/init.d/gdm start
mplayer -loop xxx
```

需要对多个对象进行操作时，可以使用空格分隔符将它们隔开：

```
touch 1 2 3 4 5 6
```

使用空格分隔的多个对象，视为一个整体，作为命令的一个操作对

象：

```
mv 1 2 3 4 5 6 /home/
```

这个命令把“1 2 3 4 5 6”作为一个操作对象，移动到另一个操作对象，“/home/”目录

递归 表示在子层次中重复相同操作。例如递归复制某目录，不但复制当前目录及其下的所有文件；而且对当前目录的子目录，也进行递归复制的操作。

格式约定

使用 [] 表示可选项，实际输入为方括号中的内容，例如

```
ls [-al]
```

ls 是必须的，参数不需要以方括号括起来。

使用 <> 表示必需项，实际输入为尖括号中的内容

使用 | 表示 **或** ，以 | 分隔的项目不能同时使用，例如

```
tar [-z|j c|x vf] <归档文件> [源文件]
```

参数通常紧跟命令名，除非必要，在命令格式中，我们通常省略它们

系统信息

uptime

联机信息-时间，显示如下

```
11:27pm    up 9 days,  7:12,    3 user,  load average:  0.07,  0.12, 0.14
```

当前系统时间　　系统运行时间　　当前在线用户数　　系统负荷　　1分钟前　　5分钟前　15分钟前

w

联机信息-已登录用户，显示如下

```
01:04:10 up  1:34,  2 users,  load average: 0.25, 0.16, 0.11
```

uptime 信息

```
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
user     ttyl     192.168.0.1       23:30    1:33   0.14s  0.12s -bash
```

用户名　登录方式　来源地址　登录时间　发呆时间　资源占用　当前任务

Tip: w [用户名称] ：显示某一用户相关信息

who

联机信息，常用参数

-r 运行级别

whoami

显示当前用户名

last

最近用户登录信息

-<数字> 使用数字作为参数，控制显示条目。例如

last -10 显示10条纪录

uname

系统信息

-s 内核名称（默认参数）
-a 全部
-p CPU信息
-n 主机名
-r 内核发行信息（版本号）
-v 内核版本信息

date

显示、设定系统时间

-u 显示格林尼洛时间（UTC）

MMDDhhmm[[CC]YY][.ss] 设定时间，需要管理员权限。例如: date 12292359

MM 月份 DD 天数 hh 小时 mm 分钟 CC 年份前两位 YY 年份后两位 ss 秒钟

秒钟、年份为可选，例如： date 122923592006.59

+[%X]设定显示格式，以下为 date 默认输出格式:

date +%Y 年%m 月%d 日%A%H:%M:%S%Z

格式控制

%n 换行
%t 制表符

小时

%H(00~23) %I(01~12) %k(0~23) %1(1~12) %p(AM|PM)

分、秒

%M 分钟(00~59)
%S 秒(00..61)
%T(hh:mm:ss) %r(hh:mm:ss [AM|PM])
%s 从 1970 年 1 月 1 日 00:00:00 UTC 到目前为止的秒数
%X(%H:%M:%S)
%Z 时区

星期

%a(Sun~Sat) %A(Sunday~Saturday) %w ：一周中的第几天 (0..6)

年份

%Y(0000~9999) %y(00~99)

月份

%m(01~12) %b %h(Jan~Dec) %B(January~December)

日期

%d(01~31) %j(001~366)
%x(本地格式 mm/dd/yy) %D(mm/dd/yy) %c

一年中的第几周

%U(00~53)以 Sunday 为一周的第一天 %W(00~53)以 Monday 为一周

的第一天

cal

显示日历

文件管理

一些细节

/ 目录为文件系统根目录，所有目录都是它的子目录

绝对路径以 / 起始，相对路径以当前所在目录起始

目录是一种特殊类型的文件，如果没有特别指明， **文件** 包括文件和目录

.. 表示上一级目录， . 表示当前目录，它们是两个特殊目录

链接

为当前文件建立在其它路径中的访问方法。例如将系统中其它位置的可执行文件，链接到 /usr/local/bin 目录下，使用命令调用。

ls [路径]

显示当前目录文件列表

--color 不同属性以不同颜色显示（默认参数）
-a 全部显示
-i 显示 inode 值
-l 详细信息
-F 显示文件类型后缀 目录/ 链接@ 可执行文件* 端口文件= 管道文件| >
-A 显示隐藏文件
-R 递归显示子目录文件列表
-S 按文件大小排序
-t 按修改时间排序
-u 按访问时间排序
-d 只显示目录，不递归显示目录下的文件

cd [目录路径]|[特殊路径]

切换目录

目录路径可以使用绝对路径或者相对路径 特殊路径:

~ $HOME 目录（默认值）
- 上一次目录
.. 上一级目录
. 当前目录

Tip:您可以通过修改 /etc/environment 文件，来定义 $CDPATH 变量，设定“cd”命令的搜索路径。

pwd

显示当前路径

file <文件名>

显示文件类型

-i 显示 mime 类型

du [路径]

计算文件或目录空间占用

-h 人性化显示。自动以 G、M、K 为单位显示占用空间大小
-l 重复计算硬链接文件大小
-L 计算符号链接文件大小
-a 显示当前目录子目录中的文件
-c 显示文件数

less <文件名>

浏览文件，使用 VI 和 Emacs 两种风格的键绑定。以下为 VI 风格键绑定

Ctrl+f(orward)	向下翻一页	Ctrl+d(own)	向下翻半页
Ctrl+b(ackward)	向上翻一页	Ctrl+u(p)	向上翻半页
/	查找	q(uit)	退出

touch <目标文件>

触碰，在不修改文件的前提下，更改其时间属性。通常用来创建一个空文件

mkdir <文件夹>

创建文件夹

-p <多级目录> 按路径创建多级目录
-m <数字权限值> 设定权限

cp <源文件> <目标目录|文件>

将源文件复制为目录文件，或者将源文件复制到目标目录。多个源文件使用空格分隔

cp <源目录> <目标目录>

将源目录复制到目标目录中，如果复制多个源目录，需要使用 -R 参数

-a 相当于-dpr 参数
-d 保留链接
-f 强制复制，覆盖目标文件
-i 覆盖时询问用户
-p 保留修改时间和访问权限
-r -R 递归复制（目录=>目录）
-l 创建链接
-v 显示过程

rm <目标目录|文件>

删除

-r -R 递归删除
-f 强制删除（无需确认，直接删除。慎用！）
-i 交互式删除（询问用户）

rmdir <目标目录>

删除目录时，建议您使用“rm -r”命令

mv <源文件> <目标目录|文件>

相当于 cp 后删除源文件，也可以作为“重命名”使用。

mv <源目录> <目标目录>

-r -R 递归

ln <源文件> <链接>

链接

-s 符号链接
-f 强制链接，覆盖目标文件
-i 覆盖前询问用户

文件操作

nano

一个简单轻便的文本编辑器，使用 Emacs 风格的键绑定。

split <源文件> [目标文件名前缀]

将源文件按一定规则分割成若干个目标文件。默认文件名前缀为 x

-<行数> 按行数分割文件
-l <行数> 同上
-b <字节> 按大小分割文件。可以使用 b、k、m 作单位，不指定单位的情况下，默认单位为 b
-C <字节> 按大小分割文件，并尽量保持每行的完整

示例：

```
split -C 100k file.split x
```

cat <文件名>

输出文件内容。用空格分隔多个文件名，可以将多个文件内容连接到一起输出。使用重定向合并为一个文件

-n 在输出中添加行号
-b 在输出中添加行号，空行不编号
-s 将两行或以上的空行，合并为一个空行

示例：

```
cat xaa xab xac > file.split
```

sort [-o <输出文件>] [-t <分隔字符>] [+<起始字段> - <结束字段>] [文件]

对文本内容排序

-m 合并文件
-c 检查文件是否已按规则排序
-b 忽略行首空格字符
-u 忽略内容重复行
-f 忽略大小写
-l 忽略非打印字符
-M 作为月份比较
-d 按字典顺序排序，按照字母、数字、空格、制表符排序
-r 逆序输出

more

查看文件内容，我们建议您使用 less

diff <文件名>

比较文件

cksum [文件名]

计算文件的 CRC 值。不指定文件名则从标准输入设备读入数据，例如：

```
echo xxx | cksum
```

md5sum [文件名] 计算文件的 md5 值。同上

权限管理

一些细节

一个文件主要包含下列属性， ls -l

```
- rwx rwx rwx  user  group  date   filename
  lll lol lol
```

其中，第一组为归属用户的权限，第二组为归属群组的权限，第三组为其它用户群组的权限。user 为文件的归属用户，group 为文件的归属群组，date 为日期信息，filename 为文件名。

对于文件夹，必须拥有它的可执行权限，才能够使用 cd 命令进入该文件夹；拥有可读权限，才能够使用 ls 命令查看该文件夹的文件列表。

root 用户拥有最高权限。

可以使用 3 位的二进制数字来描述一组权限，某一权限对应的数字为 1,则表示具有该种权限，为 0,则不具有该种权限。

使用二进制数字来描述一组权限，虽然非常直观，但是 3 组权限需要用 9 位数来表示，使用不够方便。因此我们将三组权限使用 3 位 8 进制数字来表示。它们的对应关系为：

r 100 4
w 010 2
x 001 1

将这三位 8 进制数字相加的结果，就可以表示该组权限的具体内容，例如：

7=4+2+1=rwx
5=4+1=rx
755=4+2+1 4+1 4+1=rwx r-x r-x

还可以使用 **a** 、 **u** 、 **g** 、 **o** 表示归属关系，使用 **=** 、 **+** 、 **-** 表示权限变化，使用 **r** 、 **w** 、 **x** 表示权限内容，

a 所有用户 u 归属用户 g 归属群组 o 其它用户
= 具有权限 + 增加权限 - 去除权限
r 可读权限 w 可写权限 x 可执行权限

例如：

a+x 给所有用户增加可执行权限
go-wx 将归属群组和其它用户的可写、可执行权限去掉
u=rwx 归属用户具有可读、可写、可执行权限

chmod <权限表达式> <文件|目录>

更改文件的权限。权限的表达式可以使用三位 8 进制数字表示，或者使用 augo +-= rxw-s 来表示

-R 递归
-v 显示过程
-c 类似“-v”，仅显示更改部分
--reference=<参考文件或目录> 以指定文件为参考更改权限

示例：

```
chmod -R a+x path
chmod -Rv 755 path
```

chown <归属用户>[:归属群组] <文件|目录>

更改文件的归属用户。可以使用用户名或者 UID

-R 递归

-v 显示过程

-c 类似 -v ，仅显示更改部分

--reference=<参考文件或目录> 以指定文件为参考更改权限

示例:

```
chown user:admin path
chown -R user.admin path
chown user path
```

chgrp <归属群组> <文件|目录>

更改文件的归属群组。可以使用群组名或者 GID

参数同上

SUID、SGID、Sticky bit

某些情况下，需要以可执行文件归属用户的身份执行该文件，可以为该文件设置 SUID。同样，设置 SGID 能够以该文件归属群组的身份执行它。

例如：用户自行设定密码。 出于安全方面的考虑， /etc/shadow 只能由 root 用户直接修改。

```
-rw------- root root /etc/shadow
```

这个时候，可以为程序 /usr/bin/passwd 设置 SUID，当普通用户执行“passwd”命令时，便能够以该程序归属用户 root 的身份修改 /etc/shadow 文件。而“passwd”程序自身带有身份验证机制，不能通过验证时拒绝执行，从而保证了安全。

```
ls -l /usr/bin/passwd
-r-s--x--x root root /usr/bin/passwd
```

我们发现，归属用户的可执行权限位使用 **s** ，表示 SUID。同样，归属群组的可执行权限位使用 **s** ，表示 SGID。 任何用户或群组都拥有 其它用户 的权限，所以不需要以 其它用户 身份执行文件，其它用户的可执行权限位便不会出现

s 。该权限位可能出现的属性为 t ，也就是粘着位 Sticky bit。

```
ls -ld /tmp
drwxrwxrwt root root /tmp
```

粘着位表示任何用户都可能具有写权限，但只有该归属用户或 root 用户才能够删除

SUID、SGID、Sticky bit 也可以像权限一样，使用一个八进制数表示，如下：

```
4       SUID
2       SGID
1       Sticky bit
```

通过在“chmod”命令中使用 4 个八进制数的表达式，如 4755 ，用第一位表示 SUID、SGID、或 Sticky bit，便能够为文件设置这些特殊权限。 示例：

```
chmod -R 4755 path
```

lsattr [路径]

查看文件的特殊属性

-a 全部显示
-d 只显示目录
-R 递归

特殊属性包括：

a：仅供附加用途
b：不更新最后存取时间
c：压缩后存放
d：排除在倾倒操作之外
i：不得任意更动文件或目录
s：保密性删除文件或目录
S：即时更新文件或目录
u：预防以外删除

chattr +|-|=<属性> <路径>

更改文件特殊属性

-R 递归
-V 显示过程

压缩解压

tar -c|x|u|r|t[z|j][v] -f <归档文件> [未打包文件]

将多个文件打包为一个归档文件，可以在打包的同时进行压缩。支持的格式为 tar（归档）、gz（压缩）、bz2（压缩率更高，比较耗时）

-c 创建
-x 解包
-u 更新
-r 添加
-t 查看
-d 比较压缩包内文件和文件
-A 将 tar 文件添加到归档文件中
-z 使用 gz 压缩格式
-j 使用 bz2 压缩格式
-v 显示过程
-f <文件名> 归档文件的文件名
-C <解压路径> 将压缩包中的文件解压到指定目录

[未打包文件] 创建、更新时必须填写

示例：

```
tar -zcvf xxx.tar.gz xxx/ xxx1 xxx2 xxx3 多个待打包文件以空格分隔
tar -zcvf xxx.tar.gz /home/user/xxx/  使用绝对路径打包，解包也使用绝对路径
tar -zxvf xxx.tar.gz 按相对路径解包到当前目录下，或按绝对路径解包
tar -zcvf xxx.tar.gz xxx | split -b 1m 打包后，使用 split 分割为 1m 大小的多个文件
```

其它参数

-P 使用绝对路径压缩时，保留根目录“/”
-W 校验
-p 还原文件权限
-w 询问用户
--totals 统计
-T <表达式> 处理符合条件的文件
-X <表达式> 排除符合条件的文件

zip [参数] <压缩包> <源文件>

使用 zip 格式打包文件

-r 递归，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
-y 直接保存符号连接，而非该连接所指向的文件
-X 不保存额外的文件属性
-m 将文件压缩并加入压缩文件后，删除源文件
-<压缩级别> 1~9，数字越大，压缩率越高
-F 尝试修复已损坏的压缩文件
-T 检查备份文件内的每个文件是否正确无误
-q 不显示指令执行过程
-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
-u 更新压缩包内文件
-f 更新压缩包内文件。如果符合条件的文件没有包含在压缩包中，则压缩后添加
-$ 保存第一个被压缩文件所在磁盘的卷标
-j 只保存文件名称及其内容
-D 压缩文件内不建立目录名称
-i <表达式> 压缩目录时，只压缩符合条件的文件
-x <表达式> 排除符合条件的文件
-n <文件名后缀> 排除指定文件名后缀的文件
-b <缓存路径> 指定临时文件目录
-d <表达式> 从压缩文件内删除指定的文件
-t <日期时间> 把压缩文件的日期设成指定的日期
-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
-A 调整可执行的自动解压缩文件
-c 替每个被压缩的文件加上注释
-z 替压缩文件加上注释
-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。

unzip [参数] <压缩文件> [压缩包中将被释放的文件]

解压 zip 压缩包文件
-P <密码> zip 压缩包的密码
-d <路径> 指定解压路径
-n 解压缩时不覆盖原有文件
-f 覆盖原有文件
-o 不经询问，直接覆盖原有文件
-u 覆盖原有文件，并将压缩文件中的其他文件解压缩到目录中
-l 显示压缩文件内所包含的文件

-t 检查压缩文件是否正确
-z 显示压缩包注释
-Z unzip -Z 等于执行 zipinfo 指令
-j 不处理压缩文件中原有的目录路径
-C 压缩文件中的文件名称区分大小写
-L 将压缩文件中的全部文件名改为小写
-s 将文件名中的空格转换下划线
-X 解压缩时保留文件原来的 UID/GID
-q 执行时不显示任何信息
-v 执行是时显示详细的信息
-c 将解压缩的结果显示到屏幕上，并对字符做适当的转换
-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
-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
-x <表达式> 处理里排除压缩包中的指定文件
-M 将输出结果送到 more 程序处理

7z|7za <子命令> [参数] <压缩包> [文件]

子命令

a 添加

d 删除
e 解压
x 带路径解压
l 列表查看
t 测试
u 更新
参数

-m<压缩方式>

-m0=<压缩算法> 默认使用 lzma
-mx=<1~9> 压缩级别
-mfb=64 number of fast bytes for LZMA = 64
-md=<字典大小> 设置字典大小，例如 -md=32m
-ms=<on|off> 是否固实压缩
-o<输出目录> 设置输出目录
-p[密码] 使用密码

-r[数字] 递归，使用数字定义递归子目录的深度
-sfx[<模块名称>] 使用自解压模块
-si 从标准输入设备读入数据
-so 将数据写入标准输出设备
-y 所有询问均回答 Yes
-w<工作目录>

rar <子命令> [参数] <压缩包> [文件|文件列表|路径]

子命令

x 带路径解压

e 解压到当前目录
a 将文件添加到压缩包内
d 从压缩包中删除文件
u 更新压缩包内文件
f 更新压缩包内文件，并添加压缩包内不存在的文件
m 添加并删除源文件
r 修复
l 列表查看压缩包内文件信息 lt 更详细信息 lb 简短信息
c 添加压缩包注释
cf <文件名> 将文件内容添加为注释
cw <文件名> 将注释保存为文件
t 测试压缩包
rr 添加恢复纪录
rv 恢复到文件
参数

-p<密码> 设置密码

-m<0~5> 设置压缩级别，数字越大，压缩级别越高

搜索

whereis <程序名称>

查找软件的安装路径
-b 只查找二进制文件
-m 只查找帮助文件

-s 只查找源代码
-u 排除指定类型文件
-f 只显示文件名
-B <目录> 在指定目录下查找二进制文件
-M <目录> 在指定目录下查找帮助文件
-S <目录> 在指定目录下查找源代码

locate <文件名称>

在文件索引数据库中搜索文件

-d <数据库路径> 搜索指定数据库

updatedb 更新文件索引数据库

find [路径] <表达式>

查找文件

-name <表达式> 根据文件名查找文件
-iname <表达式> 根据文件名查找文件，忽略大小写
-path <表达式> 根据路径查找文件
-ipath <表达式> 根据路径查找文件，忽略大小写
-amin <分钟> 过去 N 分钟内访问过的文件
-atime <天数> 过去 N 天内访问过的文件
-cmin <分钟> 过去 N 分钟内修改过的文件
-ctime <天数> 过去 N 天内修改过的文件
-anewer <参照文件> 比参照文件更晚被读取过的文件
-cnewer <参照文件> 比参照文件更晚被修改过的文件
-size <大小> 根据文件大小查找文件，单位 b c w k M G
-type <文件类型> 根据文件类型查找文件。b 块设备 c 字符设备 d 目录 p 管道文件 f 普通文件 l 链接 s 端口文件
-user <用户名> 按归属用户查找文件
-uid <uid> 按 UID 查找文件
-group <群组名> 按归属群组查找文件
-gid <gid> 按 GID 查找文件
-empty 查找空文件

grep <字符串>|"<正则表达式>" [文件名]

在文件中搜索内容

其它

echo <字符串>

回显。较复杂的字符串，可以使用 " 括起来。

-n 输出内容不换行
-E 不解析脱字符
-e 解析脱字符

控制字符
\ 反斜线
a 警告
b 退格
n 换行
r 回车
t 水平制表符

clear

消除屏幕

alias <输入内容> <实际内容>

别名，为命令指定一个别名，以简化输入。例如：

```
alias ls='ls --color=auto'
alias ls="l -CF"
```

可以将您的定义保存在 ~/.bashrc 文件中。

export <变量名称>

将变量导出为环境变量，常写变量赋值一同使用，例如：

```
export PATH="$PATH:xxx"
```

其中 $PATH 表示变量 PATH 原值

shutdown

关闭计算机，向根进程 init 发送信号，更改 runlevel 为 0（halt）
-h 关闭电源

-r 重启
-n 强行关机，不向 init 进程 发送信号
-k 模拟关机，向登录者发送关机警告
-t <秒> N 秒后关机
time <时间> 定时关机
-c [说明信息] 取消关机
-f 重启时忽略检测文件系统
-F 重启时强制检测文件系统

halt

关闭计算机。 调用 shutdown -h ，结束系统进程，同步文件系统，停止内核。
-n 不同步文件系统
-w 模拟关机，写 /var/log/wtmp 纪录
-f 不调用 shutdown ,强行关机
-p 默认选项，关机时调用 poweroff
-i 关机前断开网络

reboot

重新启动计算机。参数与 halt 相似

chroot <路径>

Change Root 更改根目录，重新定义会话的运行环境。

用户管理

一些细节

root 用户为根用户，也就是 系统管理员 拥有全部权限

一个用户只能拥有一个 GID ，但是还可以归属于其它附加群组

用户管理的重要配置文件：

/etc/passwd 用户名 密码位 UID 归属 GID 姓名 SHOME 目录 登录 Shell
/etc/shadow 用户名 已加密密码 密码改动信息 密码策略
/etc/group 群组名 密码位 GID 组内用户
/etc/gshadow 群组密码相关文件，不重要

/etc/sudoers　用户名 权限定义 权限

可以使用 pwconv 命令创建影子密码，将 /etc/passwd 文件中的密码转换到 /etc/shadow 文件

su [用户名]

切换到其它用户，默认切换到 root 用户。提示密码为将切换用户密码

-f 快速切换，忽略配置文件
-l 重新登录
-m ，-p 不更改环境变量
-c <命令> 切换后执行命令，并退出切换

sudo [命令]

以其它用户的身份执行命令，默认以 root 的身份执行。提示密码为当前用户密码

-s 切换为 root shell
-i 切换为 root shell，并初始化
-u <用户名|UID> 执行命令的身份
-l 显示自己的权限

passwd [用户名]

设定用户密码

-d 清除密码
-l 锁定用户
-e 使密码过期，在下次登录时更改密码
-S 显示密码认证信息
-x <天数> 密码过期，最大使用时间
-n <天数> 冻结密码，最小使用时间
-s 更改登录 Shell
-f 更改用户信息

示例：

```
$passwd
Changing password for user
(current) UNIX password:      原密码
Enter new UNIX password:      新密码
Retype new UNIX password:     确认新密码
```

chsh [-s <Shell>] [用户名]

更改登录 Shell

usermod <用户名>

修改用户账号

-d <目录> 设定$HOME 目录
-m 设定$HOME 目录时自动建立
-s <Shell> 修改用户登录 Shell
-l <新用户名> 修改为新用户名
-u <UID> 修改用户 UID
-g <群组名> 修改用户归属群组
-G <群组名> 修改用户归属附加群组
-L 锁定帐户
-U 解除锁定
-e <过期时间> 设定用户账号过期时间
-f <缓冲天数> 设定密码过期后多长时间关闭账号
-c <字符串> 修改用户备注

useradd <用户名>

新建用户

-d <目录> 设定$HOME 目录
-m 自动建立$HOME 目录
-M 不自动建立$HOME 目录
-s <Shell> 修改用户登录 Shell
-l <用户名> 修改为新用户名
-u <UID> 修改用户 UID
-g <群组名> 修改用户归属群组
-G <群组名> 修改用户归属附加群组
-n 不建立以用户名为名称的群组
-e <过期时间> 设定用户账号过期时间
-f <缓冲天数> 设定密码过期后多长时间关闭账号
-c <字符串> 修改用户备注
-D [表达式] 更改预设值 （预设值保存于 /etc/default/useradd 文件中）

新建用户规则保存于 /etc/login.defs 文件中

新建用户默认文件保存于 /etc/skel/ 目录中。新建用户时，系统自动拷贝此目录下的文件至新建用户的 $HOME 目录

userdel <用户名>

删除用户

-r 删除用户相关文件和目录

id [用户名]

显示用户 UID GID 归属附加群组

finger [用户名]

显示用户信息

进程管理

一些细节

进程一般分为交互进程、批处理进程和守护进程三类。

守护进程总是活跃，在系统启动时通过脚本自动启动，或由 root 启动，通常在后台运行。

一个进程可以拥有子进程。当父进程终止时，它的子进程也随之终止；而子进程终止时，父进程通常可以继续运行。

init 进程为根进程，所有进程都是它的子进程

ps

显示进程信息，参数可省略 -

aux 以 BSD 风格显示进程 **常用**

-efH 以 System V 风格显示进程

-e ，-A 显示所有进程

a 显示终端上所有用户的进程

x 显示无终端进程

u 显示详细信息

f 树状显示

w 完整显示信息

l 显示长列表

示例：

```
ps a1x                    另一种常用输出格式
ps aux | 1ess             将输出通过管道，使用 1ess 查看
ps aux | grep <关键字>    通过关键字查找进程
```

输出字段

```
USER          进程所有者
PID           进程 ID
PPID          父进程
%CPU          CPU 占用率
%MEM          内存占用率
NI            进程优先级。数值越大，占用 CPU 时间越少
VSZ           进程虚拟大小
RSS           页面文件占用
TTY           终端 ID
STAT          进程状态
    D    不可中断      Uninterruptible sleep (usually IO)
    R    正在运行，或在队列中的进程
    S    处于休眠状态
    T    停止或被追踪
    Z    僵尸进程
    W    进入内存交换（从内核 2.6 开始无效）
    X    死掉的进程


    <    高优先级
    N    低优先级
    L    有些页被锁进内存
    s    包含子进程
    +    位于后台的进程组;
    1    多线程，克隆线程  multi-threaded (using CLONE_THREAD, like NPTL pthreads do)
```

pstree

树状显示进程信息

-a 显示完整命令及参数
-c 重复进程分别显示
-c 显示进程 ID PID
-n 按 PID 排列进程

pgrep <进程名>

显示进程的 PID

-l 显示进程名和进程 PID
-o 进程起始 ID
-n 进程终止 ID

xkill

在图形界面中点杀进程。执行此命令后，鼠标指针变为骷髅图案（一定看过《加勒比海盗》吧）。在窗口中点击左键杀死进程，右键取消

pkill <进程名>

结束进程族。如果结束单个进程，请用 kill

kill [信号代码] <进程 PID>

根据 PID 向进程发送信号，常用来结束进程，默认信号为 -9
-l [信号数字] 显示、翻译信号代码
-9 , -KILL 发送 kill 信号退出
-6 , -ABRT 发送 abort 信号退出
-15 , -TERM 发送 Termination 信号
-1 , -HUP 挂起
-2 , -INT 从键盘中断，相当于 Ctrl+c
-3 , -QUIT 从键盘退出，相当于 Ctrl+d
-4 , -ILL 非法指令
-11 , -SEGV 内存错误
-13 , -PIPE 破坏管道
-14 , -ALRM
-STOP 停止进程，但不结束
-CONT 继续运行已停止的进程
-9 -1 结束当前用户的所有进程

renice <优先级表达式> <进程表达式>

重新设定进程优先级（无此必要）
优先级表达式:
+|-|= <nice 值>
nice 取值范围: -20~19
进程表达式:
-p <PID> 通过进程 ID 进行设定
-g <PGID> 通过进程群组 ID
-u <UID> 通过进程拥有者 UID 设定

top

动态、交互式进程管理器

-c 显示进程启动状态，包括参数、操作对象等；而不只是进程名
-d <秒> 刷新频率。 -d 5，表示5秒刷新一次
-n <次> 刷新次数，然后退出。 -n 5，表示刷新5次后退出；
-b 以批量模式运行，让输出能够使用管道或重定向。但不能进行交互，最好和
-n <次> 参数一同使用
-i 禁止显示空闲进程或僵尸进程；
-p PID 仅监视指定进程的ID；PID是一个数值；
-s 安全模式运行，禁用一些效互指令；
-S 累积模式，输出每个进程的总的CPU时间，包括已死的子进程；

交互命令：
<space> 立即刷新
k 交互式杀死进程，提示输入进程 PID （默认发送信号15）
r 设定renice，提示输入PID和renice值

s 改变两次刷新时间间隔，以秒为单位
n 设定显示进程数， 0 为不作限制
i 隐藏空闲进程和僵尸进程
S 切换到累积时间模式

1 开关，在顶部显示 *uptime* 信息
t 开关，在顶部显示 进程和CPU状态
m 开关，在顶部显示 *free* 信息

c 显示方式切换： 进程名/进程启动状态
A 按进程启动顺序进行排序。由新到旧
M 按内存占用排序。由大到小
N 以进程ID排序。由大到小
P 按CPU占用排序。由大到小
T 按时间／累积时间排序

f ， F 设定显示字段。设定完成后空格退出
o,O 设定显示字段的排序。大写向前移动，小写向后移动，空格退出
h,? 显示有关安全模式和累积模式的帮助信息
W 把当前的配置写到~/.toprc中；

nohup <命令>

将任务提交到后台，输出附加到 ~/nohup.out 文件。即便用户退出登录，提交的命令仍继续执行。

<命令> &

背景执行此命令，如果用户退出登录，则命令停止执行

<命令 1> ; <命令 2> ; ……

命令队列，从左向右，依次执行以 ; 分隔的命令

<命令 1> && <命令 2> && ……

命令队列，从左向右，依次执行以 && 分隔的命令。前一个命令执行成功，后一个命令才能执行

<命令> <Ctrl+z>

<Ctrl+z> 挂起当前 Shell 中的任务

jobs

显示背景任务

bg [任务编号]

将挂起的任务背景执行

fg [任务编号]

将背景任务调到前台执行

计划任务

cron anacron

磁盘和内存管理

一些细节

Linux 中，设备用/dev/目录下的文件表示。例如

/dev/hda1 第一块硬盘的第一主分区
/dev/hdb5 第二块硬盘的第一逻辑分区
/dev/sda4 第一块 SATA 硬盘的第四主分区，或者扩展分区
/dev/null 黑洞设备

关于磁盘设备，详见 分区概念

mount <设备文件> [挂载路径]

挂载文件系统

-t 指定文件系统的类型。通常不必指定，mount 自动检测。下面是常用的格式

reiserfs ReiserFS 3.6 版
jfs IBM 技术
xfs SGI 技术(适合高级服务器，桌面用户慎用)
ext3 Linux 传统文件系统
vfat fat fat32
ext2 不带日志的 ext3
ntfs WINNT
iso9660 光盘
smbfs Windows 文件共享

-o [选项 1] [选项 2] ……

loop 环设备。光盘、ISO 镜像等
ro | rw 只读 readonly；可读写 read-write
sync | async 同步模式|异步模式。决定修改是否立即写入文件系统
atime | noattime 读取时是否修改访问时间。对于写入敏感设备，例如闪存、软盘，建议使用 *noatime*
auto | noauto 自动挂载模式
exec | noexec 是否允许可执行权限
defaults 使用预设的选项 rw, suid, dev, exec, auto, nouser, async

iocharset=UTF-8 指定字符集，可简写为 utf8
codepage=936 指定代码页，可简写为 cp936 西文系统代码页为 437
umask=<权限掩码> 设定权限掩码
uid=<UID> 设定归属用户
gid=<GID> 设定归属群组
remount 以不同选项重新挂载

-L <*卷标* 将带有特殊卷标的分区

>

Tip

权限掩码

权限=777-权限掩码（三位）｜7777-权限掩码（四位）

假如权限掩码为 *022* ，则新建对象权限为 755 rwxr-xr-x

可以使用 **umask** 命令设置权限掩码

mount -a

挂载 /etc/fstab 文件中定义的所有设备

示例：

```
sudo mount -t iso9660 -o loop /dev/cdrom0 /media/cdrom
sudo mount -t vfat -o remount iocharset=utf8,codepage=cp936  /dev/hda5
/media/hda5
```

umount <设备文件>｜<挂载路径>

卸载已挂载文件系统

df

查看已挂载文件系统的磁盘空间占用

-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

-T 显示文件系统类型

-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 <文件系统类型> 显示指定类型的文件系统的磁盘空间使用情况

-x <文件系统类型> 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）。

-l 只显示本地文件系统

free

查看内存、缓冲区、交换空间的占用

-b 以字节为单位显示数值

-k 以千字节为单位显示数值

-m 以兆字节为单位显示数值

-g 以吉字节为单位显示数值
-l 显示内存占用峰值
-o 不显示缓冲区占用
-t 统计结果
-s <秒> 刷新频率

sync

同步文件系统。将缓冲区中的数据写入文件系统

fdisk <磁盘设备文件>

分区表修改工具

交互命令:

m 使用帮助
l 查看已知文件系统类型
p 显示分区信息
n 新建分区 (p:主分区 l:扩展分区 参见 分区概念)
d 删除分区
t 改变分区类型
w 将改动写入分区表
q 放弃改动并退出

- 磁盘设备名称为整块磁盘，而不是磁盘中的分区。例如 /dev/hda ，而不是 /dev/hda1

fdisk -l

查看所有磁盘分区信息

cfdisk

更加友善的分区表修改工具

mkfs.<文件系统类型> <分区设备文件>

将分区格式化为文件系统。 文件系统类型

示例:

```
sudo mkfs.reiserfs /dev/hda1
```

mkfs <分区设备文件>

-t <文件系统类型> 指定文件系统类型

-c 格式化前检查磁盘

mkisofs -o <镜像文件> [源文件目录]

用光盘或者文件制作 iso 镜像

-b 可启动镜像

hdparm <磁盘设备文件>

设置硬盘参数

-d <0|1> DMA 模式开关

-a <0|1> 预计模式开关

-t 性能测试

-T 缓存性能测试

-c <0|1|3> 32 位传输模式开关

-g 显示柱面，扇区等信息

-i -I 显示磁盘信息

网络和硬件管理

ifconfig

配置网络接口

-a 显示所有网络接口

ifconfig <网卡> up|down

激活|禁用网卡

示例： ::

```
sudo ifconfig eth0 up
```

ifconfig <网卡> add <IP 地址> [netmask <子网掩码>]

给网卡指定 IP 地址或子网掩码

route

配置路由及网关

route add -net <路由地址> gw <网关地址> [netmask <子网掩码>] dev <网卡>

指定路由及网关

route del -net <网关地址> gw <网关地址> [netmask <子网掩码>]

删除路由及网关

ip

配置网络

子命令:

link 网卡配置

address 配置地址。相当于 ifconfig
route 配置路由。相当于 route
参数:

show 显示 （默认）

set 设置
add 添加
del 删除

示例:

ip link show	显示网卡配置
ip link set eth0 name xxx	重命名网络接口

ping <IP 地址>

向目标地址发送 ICMP 封包，常用来测试网络

-b <广播地址> ping 整个网段
-c 发送封包次数
-s <封包大小> 默认为 64 字节

netstat

网络连接状态

-r 显示路由表，同 route
-a 所有连接
-t 只显示 TCP 协议
-U 只显示 UDP 协议
-l 只显示正在监听的端口
-p 显示 PID 和进程名
-c <秒> 刷新频率

- http/ftp/ssh…… 为应用层协议
- TCP/UDP 为传输层协议
- IP/ICMP 为网络层协议

lspci

查看 PCI 总线连接的设备

lsusb

查看 USB 接口连接的设备

lsmod

查看已加载模块

- /lib/modules/*uname -r* 目录下为所有可用模块

modprobe <模块名称>

启用模块

简明 VIM 教程

VIM 简介

我们使用的大多数编辑器，都可以直接在编辑区输入字符，并且能够通过一些快捷键来完成一些控制功能，比如使用方向键移动光标，使用 BackSpack 或者 Delete 键删除文字，使用 PgUp 和 PgDn 翻页，使用 Home 和 End 来定位行

首和行末……

而 Vim 是一个带模式的编辑器，同样的按键，在不同模式下，具有不同的功能定义。例如 h j k l 在 **编辑模式** 下输入相应的字符，在 **普通模式** 下却相当于方向键的作用。

由于需要切换模式，Vim 的使用起来略显繁琐。不过优点也显而易见：您只要把手安安稳稳的放在打字区就可以了，而不需要使用诸如方向键、排版键、小键盘等需要挪开双手的键位，从而提高了您的效率和专注程度。事实上，Vim 的前身 Vi 诞生的时候，键盘上还没有方向键、排版键和小键盘：）

命令

使用 Vim 编辑文件:

```
vi [文件名]
vim [文件名]
```

教学模式:

```
vimtutor [语言]
```

vim 教程，相当于使用 Vim 编辑器以只读模式打开教程文件。您无论对这个文件作了什么，都会在退出后恢复原来的样貌。与只读模式的区别在于，它不会没有眼色的提醒您，现在的状态为只读模式。您可以使用它作一些练习

您可以指定教程文件的语言，如果使用本地语言(ZH_cn)出现乱码，您可以尝试使用英语

```
vimtutor en
```

使用 Vim 比较文件区别

```
vimdiff [文件1] [文件2] [其它文件]……
```

配置文件

Vim 的全局配置文件为 /etc/vim/vimrc ，用户配置文件为 ~/.vimrc ， " 起始的行为注释行。我们提供的配置项，您直接加入配置文件就可以了

您可以先对 Vim 进行一些简单的配置:

```
"设定文件编码
```

```
set fileencodings=utf-8,ucs-bom,gb18030,gbk,gb2312,cp936

"开启语法加亮
syntax on
"配色风格
colorscheme pablo

"设定行距 GUI 界面中生效
set linespace=4
"设定 GUI 选项
"set guioptions=gmrLtT  m:菜单 T:工具栏
set guioptions=gmrLt

"设定 Tab 键缩进的空格数
set tabstop=4
"设定编辑器将多少空格视为一个缩进
set shiftwidth=4
"将缩进转换为空格
"set expandtab

"设定折叠方式
"set foldmethod=indent

"以下字符将被视为单词的一部分 (ASCII):
"set iskeyword+=33-47,58-64,91-96,123-128
```

模式介绍

Vim 常见的模式有： 普通模式 、 插入模式 、 命令模式 ，另外我们也会经常用到 可视模式 。

Vim 启动时进入 **普通模式** ；或者在其它模式下，按下 Esc 键，便可以回到普通模式。

使用 vimtutor en 命令进入教程，现在是普通模式。随便按几下 j 、 k 、 1 、 h 键，您会发现光标的位置发生改变。

按下 i 键，编辑器底部出现了 -- 插入 -- 或者 -- insert -- ，您进入了插入模式。随便按几下 j 、 k 、 1 、 h ，您会发现相应的字符出现在编辑区，现在还可以通过方向键来移动光标。 可能您觉得使用方向键移动光标不是什么问题，但是习惯了 Vim 后，您会认为方向键太麻烦了，简直不能容忍！好了，现在按下 Esc 键回到普通模式，我们又可以使用 j 、 k 、 1 、 h 来移动光标了。

在普通模式下，按下 : 键（也就是 Shift+; ），在编辑器底部出现了一个 : ，您进入了命令模式。在 : 后输入一个命令 new ，回车后，编辑器被分割为上下两栏。 为了方便起见，我们在命令前加一个 : 来表示命令模式下输入的命令，像这样

:vnew

一个命令能够以一些规则简化，上面的命令也可以写为这种形式

:vne

现在您的编辑区一定弄的四分五裂，您可以使用命令“:quit”来关闭当前栏，直接用简写就可以了

:q

这个命令是退出编辑器，如果编辑区被分成多栏，则是退出当前栏。

执行完一个命令（按下回车后），编辑器会自动回到普通模式。如果您想不执行当前命令，直接回到普通模式，您可以按下 Esc 键。

按下 v 键，您进入了可视模式，可以使用 j 、 k 、 1 、 h 移动光标，高亮选取文本 。

事实上，可视模式相当于高亮选取文本后的普通模式。

可视模式具有子模式，以行为单位进行选取的可视行模式，使用 V 键进入（也就是 Shift+v ）；和以块为单位进行选取的可视块模式，使用 Ctrl+v 键进入。

模式切换

好了，现在我们总结一下模式间切换的方法

其它模式==>普通模式
Esc

普通模式==>插入模式
i 在光标前插入　　I 在行首插入
a 在光标后插入　　A 在行末插入
o 在当前行之下新建行　　O 在当前行之上新建行
r 替换当前字符　　R 从当前字符开始替换

普通模式==>命令模式
:

普通模式==>可视模式
v 可视模式
V 可视块模式
<Ctrl+v> 可视块模式

移动

在普通模式中，您可以使用以下方式移动光标

j 向下
k 向上　　k
l 向右　　h　　l
h 向左　　j

您可以使用其它更有效率的方式移动光标

w 下一个单词词首　　W 将特殊符号视为单词的一部分
b 上一个单词词首　　B 同上
e 单词末尾　　E 同上

0 行首　　^ 行首文字（行首空格之后）
$ 行末

H 页面顶部
M 页面中部
L 页面底部

在其它模式中，您可以使用方向键移动光标，不过我们不推荐您那样作，您可以在配置文件中绑定插入模式下的功能键

```
noremap! <M-j> <Down>
noremap! <M-k> <Up>
noremap! <M-h> <left>
noremap! <M-l> <Right>
.................
<作用范围> <键位> <功能>
```

其中，map!绑定的键盘映射，作用于所有模式；inoremap!绑定的映射，仅作用于插入模式。

数字参数

您也可以使用数字参数，来重复执行。例如

100j 执行100次j键，向下100行

或者作为跳转的行号、百分比。见下面的 *浏览* 部分

<行号> Ctrl+g 按行号跳转

标记

您可以在当前光标处作一个标记，以便快速返回

m<标记名称> 定义标记。标记名称为一个字符
`<标记名称> 返回标记

mx 将当前光标处定义为标记 x
`x 返回标记 x

浏览

<Ctrl+f> 下翻一页 <Ctrl+d> 下翻半页
<Ctrl+b> 上翻一页 <Ctrl+u> 上翻半页
gg 文件首行
G 文件末行

<行号>G 按行号转到相应行
<1~100>% 按百分比转到相应的行数

zz 将光标所在行调整至页面中间
<Ctrl+e> 下卷一行
<Ctrl+y> 上卷一行

Tip

gg 定位到文件首行， V 进入可视行模式， G 定位到文件末行，实现类似“全选”的功能。依次按下 g g V(Shift+v) G(Shift+g)

编辑

x 剪切当前字符
dd 剪切当前行
y 复制可视模式选取字符
yy 复制当前行
p 在光标后粘贴 P 在光标前粘贴
u 撤消
<tr1+r> 重做
<Ctrl+y> 逐字克隆上一行内容

<Ctrl+e> 逐字克隆下一行内容

寄存器操作

Vim 可以将不同字段剪切或复制到不同寄存器中，您可以从不同寄存器中取出内容后粘贴

"<寄存器名称>　按下“””键和另一个字符键，便可以定义一个寄存器。例如：

"a　"l

定义寄存器后直接进行操作

"ayy　将当前行复制到寄存器 a 中
"ap　将寄存器 a 中的内容粘贴到光标之后

- 通常情况下，寄存器 +（ " + Shift+=)对应 X 下的剪贴板。您在其它程序中复制的内容，可以使用 "+p 粘贴到 Vim 中；您在 Vim 中，可以使用 "+y 将内容复制到剪贴板，再粘贴到其它程序中
- 没有指定寄存器时，Vim 使用“无名寄存器”存储内容

搜索和替换

按下 / 键，编辑器底部会出现 / 符号，接着输入字符串，便可以进行搜索

/ 向下搜索　　? 向上搜索
n 搜索下一个
N 搜索上一个

:s/<源字符串>/<目标字符串>　将源字符串替换为目标字符串
:s/<源字符串>/<目标字符串>/g　替换当前行中所有符合条件的字符串
:<行号 1>,<行号 2>s/<源字符串>/<目标字符串>/g　在指定行中进行替换
:%s/<源字符串>/<目标字符串>/g　全文替换

正则表达式

在搜索和替换时，可以使用正则表达式进行匹配

宏

您可以将一系列的操作录制为一个宏，然后执行它

q<宏名称> 开始录制宏。宏名称为一个字符
q 录制中按下“q”键，结束录制

@<宏名称> 执行宏

插入模式下的快捷键

<Ctrl+r><寄存器名称> 输入指定寄存器内容
<Ctrl+k><2 个字符> 输入二合字符
<Ctrl+v><数字> 通过数字编码输入字符
<Ctrl+v><键位> 输入键位的名称

键绑定、缩写

前面我们已经向您介绍了键绑定，

```
map! <M-j> <Down>
```

尖括号及其中的内容，为 Vim 配置文件的约定，分别描述了按键和功能，表示将功能编写到按键上。如果绑定的只是普通字符，例如：

```
map! xxx XXXXX
```

表示将 fXXXXX 绑定到 xxx 上。当您键入 xxx 时，编辑器会自动替换为 XXXXX 。

如果您只是想将字符串绑定为缩写，方便输入，我们建议您使用 iabbrev 来绑定。例如：

```
iabbrev ubt Ubuntu
```

在插入模式下键入 ubt ，编辑器会自动替换为 Ubuntu 。 您可以将 iabbrev 命令缩写为 iab ，例如：

```
iab ubt Ubuntu
```

以上命令，您可以直接在命令模式下输入，临时启用。也可以写入配置文件，永久启用。

单词补全

<Ctrl+n> 下一个匹配项
<Ctrl+p> 上一个匹配项

您可以在配置文件中定义补全的方式

```
"自动补全方式:(使用逗号分隔)
set complete=k,.

"   .     当前文件
"   b     已被装缓冲区,但是没有在窗口内的文件
"   d     在当前的文件中定义和由#include包含进来的文件
"   i     由#include包含进来的文件
"   k     由dictionary选项定义的文件
"   kfile     名为{file}的文件
"   t     标记(tags)文件
"   u     没有载入的缓冲区
"   w     在其他窗口中的文件

"设定自动补全字典 :
set dictionary=path
```

命令模式

前面介绍了普通模式和插入模式。我们发现，普通模式主要用来浏览和修改文本内容，而插入模式则用来向文本中添加内容。

而命令模式则多用于操作文本文件（而不是操作文本文件的内容），例如保存文件；或者用来更改编辑器本身的状态，例如设定多栏窗口、标签或者退出编辑器……

w(rote) 将更改写入文件

```
:w
```

q(uit) 退出编辑器:

```
:q
```

某些情况下，编辑器会阻止命令的执行。例如您修改了文件，而没有保存，那么您使用 :q 命令退出时，编辑器就不会执行这条命令，而是提醒您保存文件。

这个时候，您可以在命令末尾追加 ！来强制执行命令

```
:<命令>!
```

例如 :q! ，即便您没有保存已修改的文件，使用此命令，编辑器也会放弃修改而强行退出

以 ! 引导一个 Shell 命令，则可以从 Vim 临时切换到 Shell 中，执行一个 Shell

```
:!<命令>
```

例如 :!ls

多栏窗口

您可以使用以下命令，将当前窗口水平分为两栏

```
:new
```

新建一栏空白窗口

```
:split
```

将当前文件分两栏显示

同理，您可以使用下列命令，将当前窗口垂直分为两栏

```
:vnew
:vsplit
```

先按下 <ctrl+w> 键，再按下方向键 j 、 k 、 l 、 h ，您可以切换到其它栏；在当前栏中使用 :q 命令，可以退出当前栏，也可以使用其它命令，对当前栏作出修改

如果您希望当前命令在所有栏中生效，您可以在命令的末尾追加 all

```
:<命令>all
```

例如：

```
:qall
```

如果您希望这条命令强制执行，那么 ! 位于命令的最末

```
:<命令>all!
```

例如：

```
:qall! 强行退出所有栏窗口
```

标签页

Vim 在 7 以后的版本，开始支持标签页的功能

```
:tabnew     新建一个标签
:tabnext    转到下一个标签
:tabprevious  转到上一个标签
```

多数情况下，您可以使用鼠标点击标签进行切换。

引导管理器 Grub

硬件基础

一块硬盘，它起始的一部分扇区为主引导扇区，包括 MBR（主引导纪录）和 DPT（分区表，您可以阅读分区概念章节中相关内容）

每个分区起始的一部分扇区，为分区引导扇区。

在分区引导扇区之后的部分，为文件系统的索引，文件系统通过它定位文件在硬盘上的位置。不同的文件系统采用不同的索引，例如 FAT 文件系统使用文件分配表和目录区。

绝大多数操作系统，对硬盘的读写操作，通过文件系统来完成，因此引导扇区中的内容，我们不能够在文件系统中进行操作，而需要专用软件，比如引导管理器。

我们对文件进行修改后，操作系统会将文件系统索引中的内容同步。

系统引导流程

1. 系统启动时，首先引导至 MBR，将控制权移交安装在 MBR 中的引导管理器

   （Windows 使用 NTLDR，Linux 通常用 Grub）
2. 引导管理器读取分区表
3. 引导管理器读取分区中的配置文件，并按配置文件中预设的参数运行

   例如，Grub 读取“/boot/grub/menu.lst”文件中内容，将可引导系统通过菜单显示

4. 引导管理器根据您的选择，可能会有如下活动

加载内核，启动 Linux 系统 检查活动分区，并引导它（单一 Windows 系统） 读取相应分区的引导扇区，将控制权移交该扇区中的引导管理器，

Ubuntu 系统在安装 Grub 时，会提问您安装在 MBR 或者分区引导扇区中。如果将 Grub 安装在分区引导纪录中，您必须确保 MBR 中的引导管理器能够正确的引导至分区引导扇区。

如果您在 MBR 中使用的是 Windows 的引导管理器 NTLDR，完成这件工作会非常困难，因而我们推荐您使用 Grub。

Grub 介绍

Grub 主要有以下功能:

菜单式选择
命令行模式
支持开机画面
支持大硬盘

其它的功能还有很多，就不一一介绍了。

您可以运行命令 grub 启动它。会显示一些版本信息和使用提示，当然还有命令提示符，如下:

```
GNU GRUB  version 0.97  (640K lower / 3072K upper memory)

   [ Minimal BASH-like line editing is supported.  For
   the  first  word, TAB  lists  possible  command
   completions.  Anywhere else TAB lists the possible
   completions of a device/filename. ]

grub>
```

您可以使用 TAB 键补全命令和路径，这非常重要，因为 Grub 中路径表示方式与操作系统是不同的，您可能比较陌生，所以尽量用 TAB 补全它，既方便，也不容易出错。

您可以在 grub> 提示符后按 TAB 键，会将所有可用的命令显示出来。

呵呵，是不是有点晕，命令可真不少啊！！！

其实我们会用到的命令只有两个，

root
setup

Grub 术语

在分区概念章节里，我们已经介绍了 Linux 系统中表示分区的方法 /dev/hda5

/dev/hdMN　　M 为 a 起始的小写字母，表示硬盘序号；N 为 1 起始的数字，表示分区序号

Grub 中使用的表示方法为 hd0,1

hdX,Y　　X 为 0 起始的数字，表示硬盘序号；Y 为 0 起始的数字，表示分区序号

您得留意它们之间的区别：

N 从 1 开始计数，X 和 Y 从 0 开始计数

N 为 1~4，它是一个主分区；N 为 5 或大于 5，它是第(N-4)个逻辑分区。Y 按分区在硬盘上排列的顺序排列，无论它表示的是主分区还是逻辑分区。

举例来说：

	主	主	逻	逻	主
/dev/hdMN	hda1	hda2	hda5	hda6	hda3
hdX,Y	hd0,0	hd0,1	hd0,2	hd0,3	hd0,4

现在我们来看 root 和 setup 命令的使用：

grub>root (hd0,1)
这个命令将 Grub 的根分区定位为 “(hd0,1)”
grub>setup (hd0)
这个命令表示将 Grub 安装在“(hd0)”，因为没有指定安装的分区，所以安装位置为 MBR

Grub 的根分区 为 Grub 配置文件 /boot/grub/menu.lst 所在分区。假如您单独为 /boot 目录挂载了一个分区，那么 Grub 的根分区通常为您系统中 /boot 目录所在的分区。

搞错了根分区，Grub 就不能正确读取配置文件，自然不能正确引导。

引导分区 ，Windows 等系统的引导分区为它的安装分区，Linux 系统的引导分区为它的 /boot 目录所在的分区

系统根目录所在分区 ，Linux 根目录 / 的挂载分区。Linux 系统的分区挂载信息保存在文件系统分配表 /etc/fstab 文件中

Grub 首先读取根分区中的 /boot/grub/menu.1st 文件，并转到引导分区，如果是 Windows 等系统，则将控制权移动分区引导扇区中的启动管理器。如果是 Linux 系统，则加载内核和设备，并根据 /etc/fstab 文件的内容挂载文件系统。

看这个例子：（假设 Grub 安装在 MBR 中）

```
title       Ubuntu
root        (hd0,0)
kernel      (hd0,1)/boot/vmlinuz-2.6.15-25-686 root=/dev/sda3 ro splash vga=0x31b
initrd      (hd0,1)/boot/initrd.img-2.6.15-25-686
boot
```

Grub 的安装位置为（hd0）

root（hd0,0），这一行表示 Grub 的根分区为第一块硬盘的第一个分区（hd0,0），它读取该分区中的配置文件 grub/menu.1st

kernel 和 initrd 行中的（hd0,1），表示当前系统的 /boot 目录挂载到第一块硬盘的第二个分区（hd0,1）

kernel 行的 root=/dev/sda3 ，表示当前系统的 / 目录挂载到第一块硬盘的第三个分区（hd0,2），内核根据该分区中的 /etc/fstab 文件来挂载文件系统

Grub 配置文件

/boot/grub/menu.1st 文件，主要由一些下面这样的块构成的

```
title       Ubuntu
root        (hd0,2)
kernel      (hd0,2)/boot/vmlinuz-2.6.15-25-686 root=/dev/sda3 ro splash vga=0x31b
initrd      (hd0,2)/boot/initrd.img-2.6.15-25-686
boot


title       Windows xp
root        (hd0,0)
makeactive
chainloader +1
```

每一块代表一个操作系统，包含下面里几个部分

title xxx 标题，title 和分隔符后的内容为 Grub 菜单中显示的条目

root (hdX,Y) 引导分区，可以留空，默认为 grub/menu.lst 所在分区（根分区），可以使用 grub>root (hdX,Y) 命令设置，或者在您安装系统时自动设置

如果您计划引导至分区引导扇区，如 Windows 或者 Unix 系统（Unix 和 Linux 系统，您需要选择将引导管理器安装到分区中），那么需要这样配置

```
makeactive      设置活动分区，系统默认设置，可以删除
chainloader +1 链式引导，不要动它。
```

如果以这种方式引导系统，上面 root (hdX,Y) 这一行通常需要配置，(hdX,Y) 为您的系统所在的分区。假如您的 Windows 在第一块硬盘的第一个分区，则这样写 root (hd0,0)

如果您引导的是 Linux 系统（没有在分区中安装引导管理器，而是安装到 MBR），则需要这样配置

```
kernel      (hd0,2)/boot/vmlinuz-2.6.15-25-686 root=/dev/sda2 ro splash vga=0x31b
initrd      (hd0,2)/boot/initrd.img-2.6.15-25-686  设备镜像文件，与上一行保持一致
boot            不要动它
```

kernel 这一行最关键，它控制系统内核的加载。行末以 ro 起始的部分为参数（ro 只读，splash 显示启动画面，vga 设定启动屏幕分辨率）

之前的部分可以写为这种形式：

```
kernel     (hdX,Y)/boot/vmlinuz   root=/dev/sdMN
```

(hdX,Y) 通常不是必须的，如果您安装了多个 Linux 系统，或者 /boot``目录与根目录 ``/ 不在一个分区，则应把它写为 /boot 目录所在分区。而后面的 root=/dev/sdMN 为系统根目录 / 所在的分区。

initrd 这一行的 (hdX,Y) 与上一行保持一致。

Grub 安装

上面已经向您介绍了 Grub 的安装，不过更多的侧重理论。现在我们来实践一下，假设您的系统不能引导，您可以尝试下面的方法：）

1. 找一张 Ubuntu 的 LiveCD （Knoppix 也是不错的选择）
2. 也可以使用安装光盘，启动后在 boot 提示符后输入 linux rescue ，回车进入救援模式。

2. 如果您拥有 root 权限，命令行提示符为 # ，如果是普通用户，

则为 $ 。

1. 在救援模式下，通常你已经具有了 root 权限

b. 如果是 LiveCD 且没有 root 权限，请在终端中输入 su 命令切换到 root，会提问你 root 的密码。 如果不知道 root 密码，可以使用这个命令 sudo su ，只要你知道自己的密码，并且你在 ID 在 sudoer 列表中就可以了。

3. 在终端中输入 grub ，会进入到 Grub 的提示符界面

a. 输入命令 root（hd 后按 Tab 键，屏幕上就会列出所有可用选项。 比如我的是这样的：

```
Possible disks are: hd0 hd1
```

这表示我装了两块硬盘，如果你只有一块硬盘的话，那么一定是 hd0。

在 root(hd 后输入 0 ,按 Tab ，会自动补上一个 ，，现在你输入的内容成为这样：

```
root(hd0,
```

再按 Tab 键，会列出所有可用选项，我的是这样的。

```
Possible partitions are:
Partition num: 0, Filesystem type unknown, partition type 0x7
Partition num: 1, Filesystem type is fat, partition type 0xc
Partition num: 2, Filesystem type is reiserfs, partition type
0x83
Partition num: 4, Filesystem type is reiserfs, partition type
0x83
Partition num: 5, Filesystem type unknown, partition type 0x82
Partition num: 6, Filesystem type unknown, partition type 0x7
Partition num: 7, Filesystem type is fat, partition type 0xc
```

输入你的选择，比如为 1 ， Tab 一次后，结果是这样的：

```
root (hd1,1)
```

现在回车

2. 输入命令

```
setup (hd0)
```

将 grub 安装在 mbr 中

3. quit 命令退出 GRUB

Grub 使用

命令行

在 Grub 启动菜单中，您可以选择您要的选项，按下 e 键，进入到命令行模式

修改您的启动参数，完成后回车

按 b 键，Grub 将以您修改后的参数引导系统。

其它

在 Grub 启动菜单中，按下 c 进入命令行模式

按下 d 删除当前选中的项

FAQ

我的 D 盘到哪里去了？

在使用 Linux 最初的几天里，我感到有些不安。

Windows 下，我可以把系统装在 C 盘，软件放在 D 盘，音乐放在 E 盘……如果系统出现问题，我只要重装系统就可以了，大部分软件都可以直接使用 （养成使用绿色软件是一种好习惯：），而我辛辛苦苦收集起来的电影和音乐，我总是把它们放在一个安全的地方，以免引发失眠的严重后果。

其实 Linux 下硬盘也具有分区的概念，这一点和 Windows 没有什么不同（分区的概念是由硬盘的物理特性产生的，而不是操作系统）。不同的是，Linux 可以将分区挂载到任意的目录下，而不像 Windows，您的分区只能够在“我的电脑”里面：）

那么这么作有什么好处么？

我们来看看 Linux 和 Windows 下路径的表示方法：

Windows　　C:\Documents and Settings\Users\Documents\MyDocument
Linux　　/home/User/MyDocument

您一定注意到了，在 Windows 下面，路径中含有盘符“C:”，它通常代表您硬盘上的第一个分区。也就是说，在使用这些文件时，您需要知道它们在 硬盘上存储的相关物理细节。而在 Linux 下，您不需要知道这些，或者说，只要您设定好了分区挂载的目录，您就不需要再去理会什么分区。

Windows 下的路径包含有分区、目录和文件三部分内容;而 Linux 下的路径则只有目录和文件，不包含任何分区信息，它的硬件抽象度更高！

Linux 下的目录用“/”表示，这不标准吧？？

完全相反，这才是标准的体现。您只是习惯了 Windows 的目录符号，但是那并不意味着它是标准的。

看看 Winodws 下，各种位置、路径的表示方法:

http://www.ubuntu.org.cn
ftp://192.168.0.1
c:\Windows\
file:///C:/Windows/
\\127.0.0.1\SC
.....

操作系统是一种非常精密的高科技产品，怎么可能如此混乱！！！他们是怎么作到的？？？

Linux 下的病毒少，是因为 Linux 的使用者少，骇客显然不愿意浪费气力去攻击没有人使用的操作系统。

您可能已经知道了，互联网上用作重要用途的服务器，其中很大一部分是 Linux 系统，另外的一部分是 Unix 系统：）如果骇客能够搞掉 Linux 系统的话，那么整个互联网就会陷于瘫痪！效果似乎更好一些。

当然了，您一定会想：骇客也是人，他们也喜欢上网，兔子还不吃窝边草呢……兔子那么笨，连乌龟都跑不过……骇客们可比兔子要聪明的多了！

是的，我承认这一点……不过他们也不一定非得把互联网干掉。很多骇客作梦都想入侵美国军方的服务器，美军服务器中的绝密数据，只要 1kb，应该就可以买一台顶级的个人电脑了：）

如果可以的话，骇客为什么不去入侵美军的服务器，而要入侵您的电脑呢?

这是一个很有意思的观点，与之相映成趣，另一种论调也使人侧目：Windows 服务器占到了服务器操作系统 xx%的份额。

或许这个现象可以用 80：20 法则来解释：）

占服务器总数 80%的 Windows 提供了服务总量的 20%！

请您务必注意，这只是举一个例子，Windows 服务器可能永远也不会占到服务器总数的 80%！它提供的服务，以我个人的角度，我不认为可以达到 20%，而且永远不会有那一天。

软件安装繁琐

或许您已经看过一些关于 Linux 软件安装的文章，但是您也不要忽略，此类文章的数量，是不能够和同类 Windows 文章相比的。

当然，使用源码包安装软件确实有点麻烦，但却不一定比 Windows 下的某些软件复杂。特别要提到，Ubuntu 的包管理系统，为您提供了一种高效 快捷的软件管理方式，您只要知道您需要什么软件就可以了，甚至不需要关心它存放在网络上的哪一台服务器中，而且绝大多数的软件都可以使用这种方式来安装。

详情请参阅 APT

如果您有如下需求，您也可以尝试以源码的形式安装软件：

您需要某些软件的技术预览版本

您想测试您的机器的运算能力
您找不到一种比安装软件更好的方式来消磨时间 | （试图通过编译源码安装来大幅提高系统性能，其结果很可能会让您失望）

源码保密性不强，存在安全隐患

既然 Linux 下软件都开放源代码，那么会不会造成一些安全隐患呢？比如说一名骇客会发现其中的漏洞，并利用它？

事实刚好相反，一个软件，即便它不开放源码，骇客一样可以找到其中的漏洞，雷蒙德的软件巨头就是最好的佐证。就像一把锁，无论如何坚固，它总是能被撬开！它的作用无外乎“聊备一格，以防君子”：）

而这把锁，防住的恰恰是能够改进它的工程师！工程师知道了它的漏洞，却不能够去改进它;骇客知道了它的漏洞，却可以利用它……这把锁正是封闭源码！

这岂不是不妙?

软件功能不够强

虽然您很愿意使用 Linux 系统，但是它的软件并不能使您满意，甚至使您多愁善感的心灵又蒙上了一层阴影，“长太息以掩涕……”

首先您别忘了，《泰坦尼克》的特效就是在 Linux 系统下完成的，连业界巨头 SGI 都在向 Linux 迁移（尽管 SGI 的 IRIX 本来就是一种 Unix 系统）。如果您不知道 SoftImage，那么 Maya 您总听说过吧？它最初就是多平台的。

对于电影特效处理时需要的高吞吐量的数据（以 TB 计）和运算能力，Windows 系统恐怕连崩溃的机会都没有=_=#(最新统计资料显示， Top500 计算机中，使用 Linux 的占到 73.4%，包括最快的前两名。其中 Linux 系统 367 部，Unix 系统 98 部，混合操作系统 24 部， AppleMacOS 系统 5 部,BSD 系统 4 部，Windows 系统，2 部)

类似于大气模拟、基因解码等等真正的科学运算……Windows……前几天我还在 verycd.com 上看到一套欧洲某天文台的天文学软件，只有 Linux 版！

当然了，Linux 下功能强大的软件大多是命令行的，图形界面的程序只能视觉上强大，外强中干！建议您多使用 man 这个命令来查询各类软件的使用方法，它排版美观，格式工整，语法简明，意韵流畅，实在是学习英语难得的教材。

界面不友好

如果您指的是系统的美观程度。Gnome 默认效果我认为与 Windows 处于同一水平线，而 KDE 的效果就要略好一点，很多高手用 FVWM 可以作出让人眼花瞭乱的效果来……而 Novell 的 XGL，更是可以用“惊艳”来形容。

如果您指的是操作，这属于“易于上手难于精通”与“难于上手易于精通”两种理念的冲撞。

当然了，我指的精通主要针对效率而言。如果您经常玩 Blizzard 出品的游戏，您对于“易于上手难于精通”这种理念或许相当了解，甚至非常欣赏。

不过这一理念只适用于竞技游戏！竞技游戏要球能够吸引大量的玩家，所以要易于上手。但是竞技游戏是为竞技而生的，所以不可能人人是高手 ————事实上高手只是一小部分人！

而操作系统是给人们来用的，最好人人都成为高手，所以易于精通是很重要的

……当然最好也能够易于上手。但是考虑到效率的问题，这很难解决……

Linux 怎么占用这么多内存?

Linux 会最大程度的利用物理内存，避免使用交换空间;而不是尽量的回收内存，使用页面文件。

又因为 Linux 系统的内存管理非常优秀，程序退出时可以高效的回收内存，所以更加没有必要在程序运行时就回收内存！

因此，Linux 系统表面看来内存开销很大，实际上系统运行是很稳定的——Linux 不会时而流畅，时而瘫痪。事实上，在正常情况下，它运行是非常流畅的。