# Linux 学习笔记

郑华

2018 年 5 月 30 日

# 目录

# 第一章　Linux 基本知识

## 1.1　基本技能

正如你所见，Linux 发行版并非 Linux，Linux 仅是指操作系统的内核。

### 1.1.1　推荐的发行版

- UBUNTU 适合纯菜鸟，追求稳定的官方支持，对系统稳定性要求较弱，喜欢最新应用，相对来说不太喜欢折腾的开发者。
- Debian，相对 UBUNTU 难很多的发行版，突出特点是稳定与容易使用的包管理系统，缺点是企业支持不足，为社区开发驱动。
- Arch，追逐时尚的开发者的首选，优点是包更新相当快，无缝升级，一次安装基本可以一直运作下去，没有如 UBUNTU 那样的版本概念，说的专业点叫滚动升级，保持你的系统一定是最新的。缺点显然易见，不稳定。同时安装配置相对 Debian 再麻烦点。
- Gentoo，相对 Arch 再难点，考验使用者的综合水平，从系统安装到微调，内核编译都亲历亲为，是高手及黑客显示自己技术手段，按需配置符合自己要求的系统的首选。
- Slackware 与 Gentoo 类似。
- CentOS，社区维护的 RedHat 的复刻版本，完全使用 RedHat 的源码重新编译生成，与 RedHat 的兼容性在理论上来说是最好的。如果你专注于 Linux 服务器，如网络管理，架站，那么 CentOS 是你的选择。
- LFS，终极黑客显摆工具，完全从源代码安装，编译系统。安装前你得到的只有一份文档，你要做的就是照文档你的说明，一步步，一条条命令，一个个软件包的去构建你的 Linux，完全由你自己控制，想要什么就是什么。如果你做出了 LFS，证明你的 Linux 功底已经相当不错，如果你能拿 LFS 文档活学活用，再将 Linux 从源代码开始移植到嵌入式系统，我敢说中国的企业你可以混的很好。

### 1.1.2 Linux 基础

你得挑一个适合你的系统，然后在虚拟机安装它，开始使用它。如果你想快速学会 Linux，我有一个建议就是**忘记图形界面**，不要想图形界面能不能提供你问题的答案，**而是满世界的去找，去问，如何用命令行解决你的问题**。

在这个过程中，你最好能将 Linux 的命令掌握的不错，起码常用的命令得知道，同时建立了自己的知识库，里面是你积累的各项知识。

### 1.1.3 Linux 平台 C++ 开发

你需要学习的是 Linux 平台的 C/C++ 开发，同时还有 Bash 脚本编程，如果你对 Java 兴趣很深还有 Java。同样，建议你抛弃掉图形界面的 IDE，从 VIM 开始，为什么是 VIM，而不是 Emacs，我无意挑起编辑器大战，但我觉得 VIM 适合初学者，适合手比较笨，脑袋比较慢的开发者。Emacs 的键位太多，太复杂，我很畏惧。然后是 GCC，Make，Eclipse（Java，C++ 或者）。

虽然将 C++ 列在了 Eclipse 中，但我并不推荐用 IDE 开发 C++，因为这不是 Linux 的文化，容易让你忽略一些你应该注意的问题。IDE 让你变懒，懒得跟猪一样。如果你对程序调试，测试工作很感兴趣，**GDB 也得学的很好**，如果不是 GDB 也是必修课。这是开发的第一步，注意我并没有提过一句 Linux 系统 API 的内容，这个阶段也不要关心这个。你要做的就是积累经验，在 Linux 平台的开发经验。

我推荐的书如下：C 语言程序设计。C 语言，白皮书当然更好。C++ 推荐 C++ Primer Plus，Java 我不喜欢，就不推荐了，附一个别人的书单：java 入门书籍。工具方面推荐 **VIM 的官方手册**，**GCC 中文文档**，**GDB 中文文档**，**GNU 开源软件开发指导**（电子书），汇编语言程序设计（让你对库，链接，内嵌汇编，编译器优化选项有初步了解，不必深度）。

如果你这个阶段过不了就不必往下做了，这是底线，最基础的基础，否则离开，不要霍霍 Linux 开发。不专业的 Linux 开发者作出的程序是与 Linux 文化或 UNIX 文化相背的，程序是走不远的，不可能像 Bash，VIM 这些神品一样。所以做不好干脆离开。

### 1.1.4 UNIX 环境高级编程

UNIX 环境高级编程堪称神作，经典中的经典。

接下来进入 Linux 系统编程，不二选择，**APUE，UNIX 环境高级编程**，一遍一遍的看，看 10 遍都嫌少，如果你可以在大学将这本书翻烂，里面的内容都实践过，有作品，你口头表达能力够强，你可以在面试时说服所有的考官。

（可能有点夸张，但 APUE 绝对是圣经一般的读物，即使是 Windows 程序员也从其中汲取养分，Google 创始人的案头书籍，扎尔伯克的床头读物。）

这本书看完后你会对 Linux 系统编程有相当的了解，知道 Linux 与 Windows 平台间开发的差异在哪？它们的优缺点在哪？我的总结如下：做 Windows 平台开发，很苦，微软的系统 API 总在扩容，想使用最新潮，最高效的功能，最适合当前流行系统的功能你必须时刻学习。Linux 不是，Linux 系统的核心 API 就 100 来个，记忆力好完全可以背下来。而且经久不变，为什么不变，因为要同 UNIX 兼容，符合 POSIX 标准。所以 Linux 平台的开发大多是专注于底层的或服务器编程。

这是其优点，当然图形是 Linux 的软肋，但我站在一个开发者的角度，我无所谓，因为命令行我也可以适应，如果有更好的图形界面我就当作恩赐吧。另外，Windows 闭源，系统做了什么你更本不知道，永远被微软牵着鼻子跑，想想如果微软说 Win8 不支持 QQ，那腾讯不得哭死。而 Linux 完全开源，你不喜欢，可以自己改，只要你技术够。

另外，Windows 虽然使用的人多，但使用场合单一，专注与桌面。而 Linux 在各个方面都有发展，尤其在云计算，服务器软件，嵌入式领域，企业级应用上有广大前景，而且兼容性一流，由于支持 POSIX 可以无缝的运行在 UNIX 系统之上，不管是苹果的 Mac 还是 IBM 的 AS400 系列，都是完全支持的。另外，Linux 的开发环境支持也绝对是一流的，不管是 C/C++，Java，Bash，Python，PHP，Javascript，就连 C# 也支持。而微软除 Visual Stdio 套件以外，都不怎么友好，不是吗？

如果你看完 APUE 的感触有很多，希望验证你的某些想法或经验，推荐 UNIX 程序设计艺术，世界顶级黑客将同你分享他的看法。

## 1.2 方向选择

### 1.2.1 网络方向

服务器软件编写及**高性能的并发程序**编写

现在是时候做分流了。大体上我分为四个方向：网络，图形，嵌入式，设备驱动。

如果选择网络，再细分，我对其他的不是他熟悉，只说服务器软件编写及高性能的并发程序编写吧。相对来说这是网络编程中技术含量最高的，也是底层的。需要很多的经验，看很多的书，做很多的项目。

我的看法是以下面的顺序来看书：

1. APUE 再深读 – 尤其是进程，线程，IPC，套接字

2. 多核程序设计 - Pthread 一定得吃透了，你很 NB

3. UNIX 网络编程 – 卷一，卷二

4. TCP/IP 网络详解 – 卷一再看上面两本书时就该看了

5. TCP/IP 网络详解 – 卷二我觉得看到卷二就差不多了，当然卷三看了更好，努力，争取看了

6. Lighttpd 源代码 - 这个服务器也很有名了

7. Nginx 源代码 – 相较于 Apache，Nginx 的源码较少，如果能看个大致，很 NB。看源代码主要是要学习里面的套接字编程及并发控制，想想都激动。如果你有这些本事，可以试着往暴雪投简历，为他们写服务器后台，想一想全球的魔兽都运行在你的服务器软件上。

8. Linux 内核 TCP/IP 协议栈 – 深入了解 TCP/IP 的实现

如果你还喜欢驱动程序设计，可以看看更底层的协议，如链路层的，写什么路由器，网卡，网络设备的驱动及嵌入式系统软件应该也不成问题了。

当然一般的网络公司，就算百度级别的也该毫不犹豫的雇用你。只是看后面这些书需要时间与经验，所以 35 岁以前办到吧！跳槽到给你未来的地方！

### 1.2.2 图形方向

我觉得图形方向也是很有前途的，以下几个方面。

1. Opengl 的工业及游戏开发，国外较成熟。

2. 影视动画特效，如皮克斯，也是国外较成熟。

3. GPU 计算技术，可以应用在浏览器网页渲染上，GPU 计算资源利用上，由于开源的原因，有很多的文档程序可以参考。如果能进火狐开发，或 google 做浏览器开发，应该会很好。

### 1.2.3 嵌入式方向

嵌入式方向没说的，Linux 很重要。

掌握多个架构，不仅 X86 的，ARM 的，单片机什么的也必须得懂。硬件不懂我预见你会死在半路上，我也想走嵌入式方向，但我觉得就学校教授嵌入式的方法，我连学电子的那帮学生都竞争不过。奉劝大家，一定得懂硬件再去做，如果走到嵌入式应用开发，只能祝你好运，不要碰上像 Nokia，Hp 这样的公司，否则你会很惨的。

### 1.2.4 驱动程序设计

软件开发周期是很长的，硬件不同，很快。每个月诞生那么多的新硬件，如何让他们在 Linux 上工作起来，这是你的工作。由于 Linux 的兼容性很好，如果不是太低层的驱动，基本 C 语言就可以搞定，系统架构的影响不大，因为有系统支持，你可能做些许更改就可以在 ARM 上使用 PC 的硬件了，所以做硬件驱动开发不像嵌入式，对硬件知识的要求很高。

可以从事的方向也很多，如家电啊，特别是如索尼，日立，希捷，富士康这样的厂子，很稀缺的。

## 1.3 硬件设备

在 linux 系统中，每个设备都被当成一个文件来对待, 如图1.1。

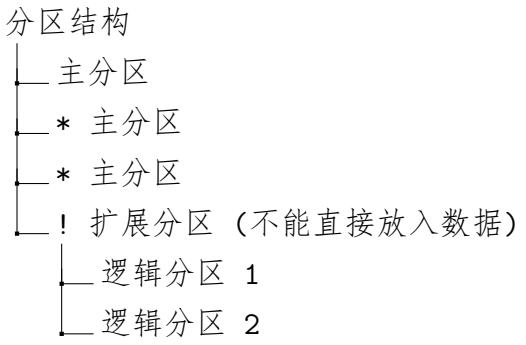
| 设备 | 设备在 linux 中的文件名 |
|---|---|
| SATA/USB 硬盘/U 盘 | /dev/sd[a-p] |
| 当前鼠标 | /dev/mouse |

表 1.1: 设备文件名

## 1.4 磁盘分区知识

磁盘分区是使用分区编辑器 (partition editor) **在磁盘上**划分几个**逻辑部分**。碟片一旦划分数个分区 (Partition), 不同类的目录与文件可以存储进不同的分区。

### 1.4.1 分区类型




**主分区** 最多有 4 个，主分区中不能再划分其他类型的分区，因此每个**主分区**都相当于**一个逻辑磁盘**（在这一点上主分区和逻辑分区很相似，但主分区是直接在硬盘上划分的，逻辑分区则

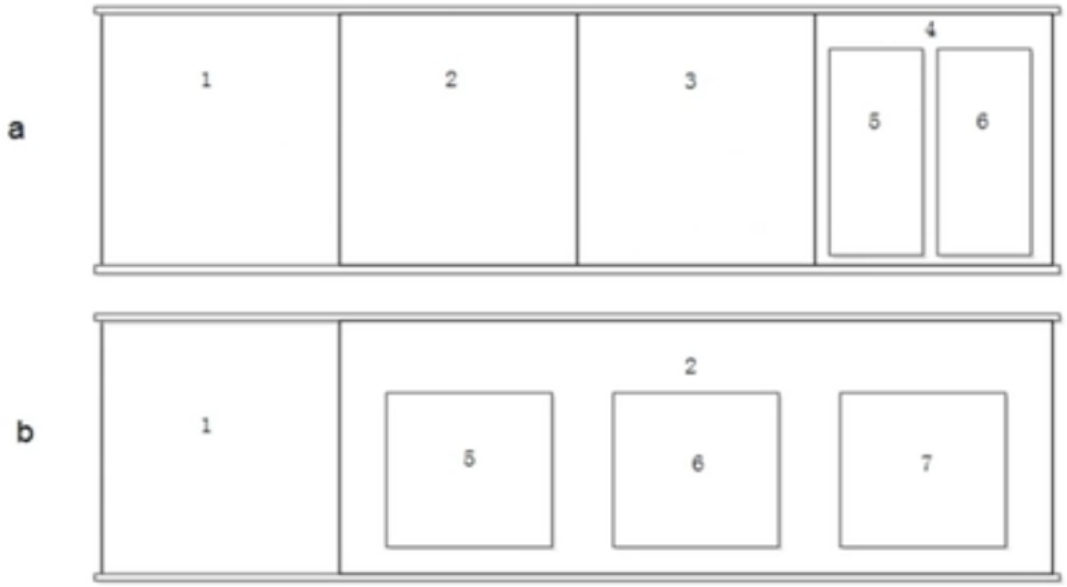



Fig 1.1: 分区设备名表示

必须建立于扩展分区中）

**扩展分区**　最多只有 1 个，在使用 MBR(主引导分区记录) 方式下 **主分区与扩展分区最多只能有 4 个**，而且扩展分区不能写入数据，只能包含逻辑分区

**分区设备文件名**　/dev/sda+数字 -> 磁盘号 + 逻辑分区编号

- 扩展分区编号只能从 4 开始

### 1.4.2　挂载

有了分区，还需要为其分配盘符，这个分配盘符的过程叫做挂载，即给分区盘符 (C D E)，**使得系统能够识别不同分区**。

**推荐系统分区结构**　->

磁盘
- /boot
- /home
- /
- sawp

**必须分区**

- / 根分区
- swap swap 分区（交换分区，内存的两倍，不超过 2G）

**推荐分区** /boot (启动分区，200MB)

### 1.4.3 GUID

## 1.5 裸机装机必备软件及技巧

**vim** 必备，要么你修改配置文档都是个问题。通过 wifi 先将基本的 vim 搞定。

```
sudo apt-get install vim
```

编程相关的插件到后续再安装。

**网络配置** 有网才能工作

1.ln -s 创建快捷命令

2.1 配置 IP(一般不动)

- vim /etc/network/interfaces
- 根据情况确定用动态 IP 还是静态 IP

```
#动态IP
    auto eth0
    iface eth0 inet dhcp

#静态IP
    auto eth0
    iface eth0 inet static
    address xx.xx.xx.xx
    netmask xx.xx.xx.xx
    gateway xx.xx.xx.xx
```

2.2 配置 DNS

- sudo vi /etc/resolv.conf
- 添加nameserver 8.8.8.8 或 8.8.4.4

- 重启网络：

```
service networking restart
```

```
sudo /etc/init.d/networking restart
```

**更新源**　使用国内阿里云镜像，更快速度。

**输入法**

1. `sudo apt-get update`
2. `sudo apt-get install fcitx-googlepinyin`
3. 注销或重启
4. 找标题栏中的小企鹅，点开找配置 fcitx
5. 点+, 取消选中 only. 查找 Google-pinyin

**文本工具 Texlive 与 Texstudio**

**git**

**网易云音乐**　有音乐工作才有动力

1. 官网下载 deb
2. `sudo dpkg -i ~/Downloads/xx.deb`
3. 出现错误时`sudo apt install -f`
4. 再次运行安装命令

**vim 插件安装**　http://blog.csdn.net/lhf_tiger/article/details/7216500

1. Ctags: `sudo apt-get install ctags`

   进入目录中 `ctags -R`

2. Cscope: http://blog.csdn.net/av_geek/article/details/47976981

```
sudo apt-get install cscope
```

- 进入项目主目录 `find . -name "*.h" -o -name "*.cc">cscope.files`
- 执行 `cscope -bkq -i cscope.files`
- 执行 `ctags -R`
- 进入要编辑的文件中:`cs add cscope.out`
- 然后执行相应的命令 `:cs find c|d|e|f|g|i|s|t name`

# 第二章　命令相关

命令格式：

命令　[-选项]　[参数]

1. 个别命令不遵循此格式
2. 当有多个选项时，可以写在一起 `ls -la` 等价于 `ls -l -a`
3. 简化选项与完整选项 `-a` 等价于 `--all`

## 2.1　Linux 系统

**X windows 与 Console**

- `ctrl + alt + F1~F6`：Console 终端模式
- `ctrl + alt + F7`：图形模式界面

**目录结构**　linux 目录如表2.1所示。

**PATH**　当执行一个指令 (ls) 的时候，系统会依照`$PATH` 的设置去每个`$PATH` 定义的目录下搜寻文件名为 (ls) 的可执行文件，如果在`$PATH` 定义的目录中含有多个文件名为 (ls) 的可执行文件，那么先搜寻到的同名指令先被执行。

如果命令不在`$PATH` 指定路径下时，单纯敲击指令名称是不会执行的，有以下两种方法可以执行该指令

1. 使用绝对路径或相对路径指定命令 `./ls` 或 `/root/ls`
2. 将目标目录添加到`PATH` 中 `PATH = "${PATH}:/root"`

表 2.1: Linux 目录详解

| 目录 | 存放的内容类别 |
| --- | --- |
| / | 文件系统的入口，**最高一级目录** |
| /bin | 存放**系统命令**的目录，**普通用户和超级用户都可以执行**。不过放在/bin 下的命令**在单用户模式 (安全模式) 下也可以执行** |
| /sbin | 保存和**系统环境设置**相关的命令，**只有超级用户**可以使用这些命令进行系统环境设置，部分命令可以被普通用户查看 |
| /usr/bin | 存放**系统命令**的目录，**普通用户和超级用户都可以执行**，这些命运与启动无关，不能在单用户模式下运行。 |
| /usr/sbin | 存放**根文件系统不必要的系统管理**命令，例如多数服务程序，**只有超级用户**可以使用。/sbin 目录只用超级用户可以使用，/bin 目录存放的命令都可以使用。 |
| /dev | **设备文件存储目录**，是 Linux 文件系统的一个闪亮的特性 - 所有对象都是文件或目录。仔细观察这个目录你会发现 hda1, hda2 等, 它们代表系统主硬盘的不同分区。 |
| /etc | 存放系统程序或者一般工具的**配置文件** |
| /lib | 库文件存放目录这里包含了**系统程序所需要的所有共享库文件**，类似于 Windows 的共享库 DLL 文件。 |
| /lost+found | 在 ext2 或 ext3 文件系统中，当系统意外崩溃或机器意外关机，而**产生一些文件碎片放在这里** |
| /mnt | 这个目录一般是用于**存放挂载储存设备的挂载目录**的，比如有 cdrom 等目录。有时我们可以把让系统开机自动挂载文件系统，把挂载点放在这里也是可以的。比如光驱可以挂载到/mnt/cdrom |
| /opt | 表示的是可选择的意思，**有些软件包也会被安装在这里，也就是自定义软件包** |
| /proc | /proc 目录是伪装的文件系统 proc 的挂载目录，proc 并不是真正的文件系统。这个目录本身是一个“虚拟文件系统（virtual filesystem）”喔！他放置的数据都是在内存当中，例如系统核心、行程信息（process）、周边设备的状态及网络状态等等。因为这个目录下的数据都是在内存当中，所以本身不占任何硬盘空间啊！ |
| /tmp | 临时文件目录，每次开机会清空该目录。 |
| /usr | (Unix Software Resources) 这个是**系统相关第三方程序存放目录**，比如命令、帮助文件等。这个目录下有很多的文件和目录。 |
| /var | 这个目录的内容是经常变动的，如/var/log 系统日志存放，分析日志要看这个目录的东西 |

用户配置

- ~/.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. 此文件类似于/etc/profile，也是需要需要重启才会生效，~/.bash_profile **只对当前用户生效**。

- ~/.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, 该文件被读取.（每个用户都有一个.bashrc 文件，在用户目录下）此文件类似于/etc/bashrc，**不需要重启生效，重新打开一个 bash 即可生效**，但~/.bashrc 只对当前用户新打开的 bash 生效。

- /etc/profile

  /etc/profile 对所有用户生效

- /etc/bashrc

  /etc/bashrc 对所有用户新打开的 bash 都生效

**交换区** 在内存小于 2G 的情况下，交换分区**应为内存的 2 倍**，超过 2G 的话，交换分区为**物理内存加上 2G**

**符号链接** 符号链接又叫软链接, 是一类特殊的文件，这个文件包含了**另一个文件的路径名**

**开关机** 先关服务，再关机

服务器不能关机，只能重启，因为一般都在远程。

- shutdown -r 20:30: 在 20:30 重启

  1. -c: 取消前一个关机命令
  2. -h: 关机
  3. -r: 重启

- poweroff
- init 0

**重启**

- `shutdown -r now`
- `reboot`
- `init 6`

**系统运行级别** `cat /etc/inittab` 可以查看运行级别的含义。

- 0：**关机**
- 1：单用户-> 安全模式，附加程序构件不运行
- 2:【命令行】不完全多用户，不含 NFS 服务
- 3:【命令行】完全多用户
- 4：未分配
- 5:【图形界面】
- 6：**重启**

查询当前运行级别：`runlevel`

运行某级别：`init x`

## 2.2 导航命令

**pwd** Print Working Directory 打印当前工作目录，即当前所处位置的绝对路径

`-P`：显示出确实的路径，而非使用链接`link` 路径。

```
范例：显示出实际的工作目录，而非链接文件本身的目录名而已
[root@study ~]# cd /var/mail   <==注意，/var/mail是一个链接文件
[root@study mail]# pwd
/var/mail          <==列出目前的工作目录
[root@study mail]# pwd -P
/var/spool/mail    <==怎么回事？有没有加 -P 差很多～
[root@study mail]# ls -ld /var/mail
lrwxrwxrwx. 1 root root 10 May  4 17:51 /var/mail -> spool/mail
#  看到这里应该知道为啥了吧？因为 /var/mail 是链接文件，链接到 /var/spool/mail
#  所以，加上 pwd -P 的选项后，会不以链接文件的数据显示，而是显示正确的完整路径啊！
```

Fig 2.1: pwd -P 选项示例

**cd** 改变当前工作目录

- 绝对路径：从根目录 / 开始
- 相对路径：从当前位置开始，. 表示当前目录，.. 表示上级目录
- `cd ~ || cd` ：进入当前用户家目录
- `cd -` ：进入上次目录
- `cd ..` ：进入上一级目录
- `cd .` ：进入当前目录

**ls** 列出目录内容

- 所在路径：`/bin/ls`
- 执行权限：所有用户

表 2.3: **ls** 参数说明

| 简化选项 | 完整选项 | 意义 |
|---|---|---|
| `-a` | `--all` | 列出所有文件，包括以点号开头的文件，这些文件通常是不列出来的 (比如隐藏的文件) |
| `-d` | `--directory` | 查看目录的详细信息，而不是目录里的内容 |
| `-F` | `--classify` | 如果名字是目录，则会加上一个斜杠 |
| `-l` | | 使用长格式显示结果 |
| `-i` | | `-inode` 印出每个文件的`inode` 号 |
| `-r` | `--reverse` | 以相反的顺序显示结果 |
| `-S` | | 按照文件的大小对结果进行排序 |
| `-t` | | 按照文件的修改时间排序 |
| `--color` | `={never, auto, always}` | 结果颜色选项 |
| `--full-time` | | 完整呈现结果的最后修改时间 |

## 2.3 操作文件和目录

### 2.3.1 文件系统

**EXT2**

- `superBlock`: 记录文件系统相关属性
  - dataBlock 与 Inode 大小熟悉
  - dataBlock 与 Inode 数量
- `bitMap`: 用于判断有哪些数据块可用
- `inodeMap`: 用于判断有哪些 Inode 块可用
- `inode(s)`: 存放文件的相关属性，和相关数据块指针 (涉及 2 级 3 级指针)。*每个文件仅会占用 1 个 Inode*
- `dataBlock(s)`: 存放数据的实际地方，*每个数据块只能存放一个文件的内容*, 所以目录文件一般独自占用一个 dataBlock。

-> 例子：当我们在 Linux 下的 ext2 **创建一个一般文件时，ext2 会分配一个 inode 与相对于该文件大小的 block 数量给该文件**。例如：假设我的一个 block 为 4 KBytes，而我要创建一个 100 KBytes 的文件，那么 linux 将分配一个 inode 与 25 个 block 来储存该文件！但同时请注意，由于 inode 仅有 12 个直接指向，因此还要多一个 block 来作为区块号码的记录喔！

**dumpe2fs** 列出相关设备的 superBlock 的相关信息

```
[root@study ~]# dumpe2fs /dev/vda5
```

**df** (disk free 可用磁盘) 列出文件系统的**整体磁盘使用量**;

- `-h` 选项，以人类易读的格式输出（例如，5K，500M 及 5G）
- `-a` 选项，显示所有文件系统的磁盘使用情况
- `-i` 选项，用于显示文件系统的 inode 信息

参考 https://linux.cn/article-6466-1.html

**du** du 可以显示当前目录及子目录的磁盘占用情况

- -d 选项可以指明递归目录的深度
- -s 等价于
- -h 表示以可读的形式显示，比如B，KB，GB 等

### 2.3.2 相关命令

通配符 ->

表 2.5: 通配

| 形式 | 含义 |
|---|---|
| * | 匹配任意多字符-包括 0 个 1 个 |
| ? | 匹配任一单个字符-不包括 0 个 |
| [characters] | 匹配任意一个属于字符集中的字符 |
| [!characters] | 匹配任意一个不属于字符集的字符 |
| [[:class:]] | 匹配任意一个属于指定类的字符，如[[:digit:]] |
| [[:alnum:]] | 匹配任意一个字母或数字 |
| [[:alpha:]] | 匹配任意一个字母 |
| [[:digit:]] | 匹配任意一个数字 |
| [[:lower:]] | 匹配任意一个小写字母 |
| [[:upper:]] | 匹配任意一个大写字母 |

**cp** 复制文件和目录，默认会改变时间戳

- -r: 复制目录
- -p: 连带文件属性复制
- -d: 若源文件是链接文件，则复制链接属性
- -a: 相当于-pdr
- -i: 若目标文件已经存在时，在覆盖时会询问动作是否进行。
- -u: 只有目标文件与原文件有差异时，才会复制

`cp fileName1  /dir/fileName2`: 改名复制

`cp fileName1  /dir/`: 原名复制

-> 复制总是希望复制到的数据最后是我们自己的，所以，在默认的条件中，cp 的**来源文件**与**目的文件**的**权限是不同的**，**目的文件**的**拥有者**通常会是**指令操作者本身**。

**mv**　移动或重命名文件和目录

当源文件和目标文件在同一目录就是改名

当源文件和目标文件不在同一目录就是剪切

**dd**　用指定大小的块拷贝一个文件，并在拷贝的同时进行指定的转换

- if= 文件名：输入文件名，缺省为标准输入。即指定源文件。`< if=input file >`
- of= 文件名：输出文件名，缺省为标准输出。即指定目的文件。`< of=output file >`
- http://www.cnblogs.com/ginvip/p/6370836.html

**mkdir**　创建目录

- 可以连续创建多个目录 `mkdir /temp/Japan/cangjing  /temp/Japan/longze ...`
- `-m` ： 设置文件的权限喔！直接设置，不需要看默认权限umask 的脸色
- `-p` ：递归创建 `mkdir -p /Hello/Practice/First`

**rmdir**　Remove Empty Directory: 即只能删除空目录

**rm**　移除文件和目录，默认删除时会确认

- `rm -r`: 删除目录
- `rm -f`: 强制删除，不用确认是否删除

**ln**　创建硬链接和符号链接

- **硬链接**:`ln file file_hard`

1. **拥有相同的 i 节点和存储块**，可以看作是带有引用计数的`shared_ptr`，因为本质只有一个资源，而建立新的链接相当于添加新的代理类，但是访问资源都是一个资源。
2. 可以通过 i 节点识别: 通过命令`ls -i` 查看
3. 不能针对目录使用
4. 不能跨分区
5. 创建后，`ls -1` 后，会发现第 2 个参数，引用增加，其他不变（权限、大小）

- **软链接**:`ln -s file file_soft` 在软连接中，文件实际上是一个文本文件，其中包含的有另一文件的位置信息。

1. 类似于 windows 的快捷方式
2. 依赖于硬链接
3. 有自己的存储块，存储对应的链接命令文件位置。
4. 访问时，先访问自己的存储块读取要读写的文件，然后再打开该文件，如果硬链接将该文件删除了，那么执行软链接就会失败。
5. 改变软链接文件同样对源文件进行操作。
6. 删除其对文件是否再存在没影响
7. Linux 使用时尽量使用**绝对路径**，两者都，相对可能会出错
8. 创建后，`ls -1` 后，会发现第 1 个参数为`1rwxrwxrwx`，其他也与源文件不同，因为只是快捷方式。

-> **区别**：https://www.ibm.com/developerworks/cn/linux/l-cn-hardandsymb-links/

我们知道文件都有文件名与数据，这在 Linux 上被分成两个部分：**用户数据** (user data) 与**元数据** (metadata)。用户数据，即文件数据块 (data block)，数据块是记录文件真实内容的地方；而元数据则是文件的附加属性，如文件大小、创建时间、所有者等信息。

在 Linux 中，**元数据中的 inode 号**（inode 是文件元数据的一部分但其并不包含文件名，inode 号即索引节点号）**才是文件的唯一标识而非文件名**。文件名仅是为了方便人们的记忆和使用，系统或程序通过 inode 号寻找正确的文件数据块。图2.2展示了程序通过文件名获取文件内容的过程。

**查看** inode 号可使用命令 `stat` 或 `ls -i`;

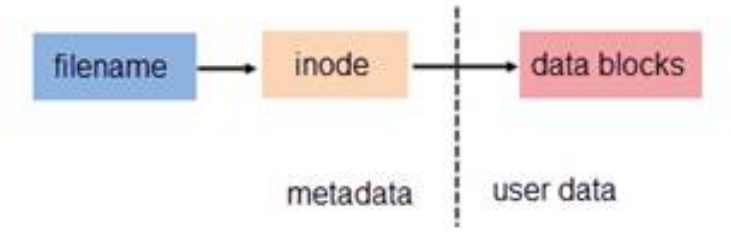



Fig 2.2: Linux Link Data Access Roud

**查找**有相同 inode 号的文件 `find / -inum 1114`

**查看**路径 /home 有相同 inode 的所有硬链接 `find /home -samefile /home/old.file`

1. 若一个 `inode` 号对应多个文件名，则称这些文件为硬链接。换言之，硬链接就是同一个文件使用了多个别名
2. 软链接与硬链接不同，若文件用户数据块中存放的内容是另一文件的路径名的指向，则该文件就是软连接。软链接就是一个普通文件，只是数据块内容有点特殊。

## 2.4 查看文件

**cat** 由第一行开始显示文件内容

- `-b`：列出行号，仅针对非空白行，空白行不标行号。
- `-n`：打印出行号，包括空白行
- `-v`：列出一些看不出的特殊字符
- `-E`：将结尾的断行字符`$` 显示出来
- `-T`：将`tab` 键以`^I` 的形式显示出来
- `-A`：`-vET` 可以列出特殊字符，而不是以空白代替

**tac** 由最后一行开始显示文件内容，反向`cat` 输出

**nl** 显示的时候，顺道输出行号

**more** 一页一页的显示文件内容

- 空格 Or f -> 翻页
- 回车 -> 换行
- q -> 退出

**less** 与more 类似，但是比more 更好的是，它**可以往前翻页、可以查找**

- pageUp -> 向上翻页
- Up Down -> 上下换行
- /key -> 查找，n 向下查找

**head** 只看头几行，head -n 5 ls-text.txt，不指定多少行时，默认显式 10 行。

-n：后面接数字，代表显示几行的意思

**tail** 只看尾部几行，tail -n 5 ls-text.txt

- -f 当文件增长时, 输出后续添加的数据
- -s 与-f 合用, 表示在每次反复的间隔休眠 S 秒

为了查看不断更新的日志文件，可以使用的指令tail -f

**od** 以二进制的方式读取文件内容

**touch** 修改文件时间或创建新文件

文件时间概念

- modification Time(mtime)：**当该文件的“内容数据”变更时，就会更新这个时间！**内容数据指的是**文件的内容**，而不是文件的属性或权限喔！
- status Time(ctime)：**当该文件的“状态（status）”改变时，就会更新这个时间**，举例来说，像是**权限与属性**被更改了，都会更新这个时间啊。
- access Time(atime)：当“该文件的内容被取用”时，就会更新这个读取时间（access）。举例来说，我们使用 cat 去读取 /etc/man_db.conf ，就会更新该文件的 atime 了。

**touch 选项**

- -a: 修改文件访问时间 atime
- -c: 修改文件的时间 ctime
- -m: 修改文件的 mtime

## 2.5 ’命令’解释帮助命令

**man** 查看帮助，man 级别 命令，级别含义如下

- 1 可执行程序或 shell 命令【默认】
- 2 系统调用 (内核提供的函数)
- 3 库调用（程序库中的函数）
- 4 特殊文件（通常位于/dev）
- 5 配置文件或规范（/etc/passwd）
- 6 游戏
- 7 其他

**type** 说明如何解释命令名：type command

**which** 说明会执行在哪块的可执行程序：which ls –> /bin/ls

**whereis** 说明会可执行程序、配置文件的位置：which ls –> /bin/ls

**whatis** 显示命令的简要描述:whatis ls

**man** 显示程序的手册页:man command，查看配置文件时切记不要使用绝对路径，当有多个级别的命令时，可以使用-f 指定级别。

优先查找命令的帮助、其次是配置文件

当同时存在命令与配置文件时，使用 man 时，查看命令添加 1->man 1 passwd, 如果是配置文件则使用 5man 5 passwd

**–help** 显示程序的使用信息:`command --help`

**help** 获得`Shell` **内置命令** (找不到路径的命令) 的帮助信息

```
->help umask
->help if
```

**apropos** 显示适合的命令:`apropos keyWords`

**info** 显示程序的 info 条目:`info command`

**alias** 使用别名创建自己的命令:`alias name='command[s]'`

**参考** http://man.linuxde.net

## 2.6 重定向

这个功能可以把命令行的输入重定向为从文件中获取内容，也可以把命令行的输出结果重定向到文件中。使用重定向符‘>’，后接文件名，就可以把标准输出重定向到另一个文件中，而不是显示在屏幕上。

当使用重定向符'>' 来重定向标准输出时，目的文件通常会从文件开头部分重新改写。如果需要删除一个文件内容（或者创建一个新的空文件），可以采用这样的方式。`> fileName`

如果不需要从文件的首位置开始覆盖文件，而是从文件的**尾部开始添加内容**，我们可以使用重定向符‘>>’来实现。

可以使用重定向符`&>` 来把标准输出和标准错误都重定向到同一文件中。

**cat** 除了查看文件，还可以将不同的文件合并到一个文件里，如 `cat movie*.mpeg > movie` movie.mpeg

**管道** 命令从标准输入得到数据，并将数据处理后发送到标准输出。使用管道操作符可以把一个命令的输出传送到另一个命令的标准输入中。 ‘|’

**sort** 按照顺序排列输入表

**uniq** 可以接受来自于标准输入或者一个单个文件名对应的已经排好序的数据列表，默认情况下，该命令删除列表中的所有重复行，因此在管道中常与 sort 结合使用–> `ls /bin /usr/bin sort` | uniq | less|

**wc** 用来显示文件包含的-行数，字数，字节数`-l`，`-w`，`-m`，`-c`。

**tee** 从 stdin 读取数据，并同时输出到 stdout 和文件

**/dev/null 文件** /dev/null 是一个特殊的文件，写入到它的内容都会被丢弃；如果尝试从该文件读取内容，那么什么也读不到。但是 /dev/null 文件非常有用，将命令的输出重定向到它，会起到”**禁止输出**” 的效果。

## 2.7 shell 扩展

**echo** 把文本参数内容打印到标准输出:`echo *` –> 打印当前工作目录的所有文件名

**波浪线扩展** 如果没有指定用户名,则扩展为当前用户的主目录:`echo ~ --> /home/me echo ~foo --> /`

**算术扩展** 调用方式：`echo $(( expression ))`

**花括号扩展** `echo F-{A,B,C}-B` –> F-A-B F-B-B F-C-B; `echo n{1..4}` –> n1 n2 n3; `echo n{z..a}` –> nz ny nx . na;

**命令替换** 可以把一个命令的输出作为一个扩展模式使用：`echo $(ls)` –>Desktop Documnet Music Pictures.. , 即在需要命令参数处调用‘`$(Command)`’

**双引号扩展** 如果把文本放到双引号里，那么所有特殊字符都将失去特殊含义除算数扩展和命令替换

**单引号扩展** 如果把文本放到单引号里，那么所有特殊字符都将失去特殊含义，没有除

## 2.8 用户管理

### 2.8.1 配置

**用户信息文件/etc/passwd** 可以通过man 5 passwd 查看。

每行代表一个用户-> 用户名: 密码标志: UID(用户id): GID(用户初始组): 用户说明: 家目录: 登录之后的shell。

**UID** -> 用户标识

- 0 超级用户
- 1~499 系统用户 (伪用户-> 系统运行时用到的)
- 500~65535 普通用户

即想把一个普通用户转化成超级用户，**修改** passwd 配置文件的 **UID 段**为 0 即可。

**影子文件/etc/shadow** 每行一个用户-> 用户名:加密密码:密码最后一次修改日期:两次密码修改间隔: 密码生效时间: 提醒时间: 宽限时间

**组信息文件/etc/group、组密码/etc/gshadow**

### 2.8.2 命令

**useradd** 添加新用户 : useradd 用户名

- -u ->UID: 指定用户的 UID 号
- -d ->家目录: 指定用户的家目录
- -c ->用户说明: 指定用户的说明
- -G ->附加组名: 指定用户的附加组
- -s ->shell: 指定用户的登陆 Shell

**用户默认值文件** ~

/etc/default/useradd ->

- GROUP = 100 : 用户默认组
- HOME = /home: 用户家目录
- INACTIVE = -1: 密码过期宽限天数
- EXPIRE=: 密码失效天数
- SHELL=/bin/bash: 默认 shell
- SKEL=/etc/skel: 模版目录
- CREATE_MAIL_SPOOL=yes: 是否创建目录

/etc/login.defs ->

- PASS_MAX_DAYS 9999: 密码有效期
- PASS_MIN_DAYS 0: 密码修改时间间隔
- PASS_MIN_LEN 5: 密码最小 5 位
- PASS_WARN_AGE 7: 密码到其警告
- UID_MIN 500: 最小和最大 UID 范围
- GID_MAX 60000
- ENCRYPT_METHOD SHA512: 加密模式

间接影响文件 ->

```
#useradd sc
#grep sc /etc/passwd
#grep sc /etc/shadow
#grep sc /etc/group
#grep sc /etc/gshadow
#ll -d /home/sc/
#ll /var/spool/mail/sc
#ll /etc/skel
```

**passwd** 更改密码 -> passwd 用户名

root 可以更改任何人的密码

普通用户 只能更改自己的密码

- -S ：查询用户密码的密码状态。仅 root 用户可用
- -l ：暂时锁定用户。仅 root 用户可用
- -u ：解锁用户。仅 root 用户可用
- --stdin ：可以通过管道符输出的数据作为用户的密码。echo '123'|passwd --stdin user1

**usermod** 修改用户信息，选项同 useradd

- -c ：修改用户帐号的备注文字
- -d ：修改用户登入时的目录
- -e ：修改帐号的有效期限
- -f ：修改在密码过期后多少天关闭该帐号
- -g ：修改用户所属的群组
- -G ：修改用户所属的附加群组
- -l ：修改用户帐号名称
- -u ：修改用户 uid

**chage** 修改用户密码状态

- -l ：列出用户的详细密码状态
- -d ：修改密码最后一次更改日期（shadow 3 字段）
- -m ：两次密码修改间隔（4 字段）
- -M ：密码有效期（5 字段）
- -W ：密码过期前警告天数（6 字段）
- -I ：密码过后宽限天数（7 字段）
- -E ：帐号失效时间（8 字段）

要求用户初次登录就修改密码：chage -d 0 user2

**id** 查看用户 UID 和 GID

**userdel** 删除用户

-r ：删除用户的同时删除用户家目录-> userdel -r User3，一般都需要加-r 选项。

**su** 切换用户

- - ：选项只使用‘-’代表用户的环境变量一起切换
- -c ：仅执行一次命令，而不切换用户身份。

默认情况下su user1 切换只是部分切换，而对环境变量不进行改变，可以使用env 查看。

而‘-c’命令经常如下使用：su - root -c "useradd user3", 使用 root 身份执行一次 root 命令但不切换用户。

**groupadd** 添加用户组groupadd 选项 组名

-g GID: 添加指定 ID 组

**groupmod** 修改用户组

- -g GID: 修改组 ID
- -n 新组名: 修改组名

把组名 group1 修改为 test1：groupmod -n test1 group1

**groupdel** 删除用户组

- 用户的主组为删除组不能删除
- 其他用户附加组为删除组不影响删除

**gpasswd** 把用户加入组或从组中删除。gpasswd 选项 组名

- -a 用户名: 把用户添加到组
- -d 用户名: 把用户从组中删除

**who** 查看登陆用户信息

显式内容-> 用户名 登录终端（tty 本地终端，pts 远程终端） 登陆时间 IP地址

**whoami** 显式当前登陆用户信息

**w** 获取更多的登陆用户信息

显式内容line1-> uptime，登陆用户个数，load average

显式内容line2-> 用户名 登陆终端 IP地址 登入时间 空闲时间 累计占用时间 当前操作占用时间 具体操作

## 2.9 权限-基础

### 2.9.1 基础

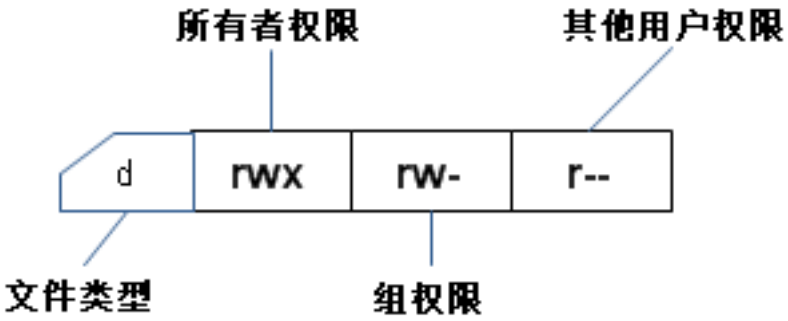



Fig 2.3: ls -l 文件权限

```
Type Owner Group Others
[- rwx rwx ---]
[- 111 111 000]
[- 421 421 000]
[- 7 7 0 ]
```

表 2.6: 权限说明

| 权限 | 含义 | 对文件 | 对目录 |
|---|---|---|---|
| r | 读 | 可以查看文件内容->cat等 | 可以列出目录中的内容->ls |
| w | 写 | 可以修改文件内容->vim | 可以在目录中创建、删除文件->touch/mkdir/rmdir/rm |
| x | 执行 | 可以执行文件->script | 可以进入目录->cd |

因此在目录中，有x执行权限肯定有r读权限

### 2.9.2 命令

**chmod** 更改文件的模式-> chmod 权限 文件

chmod 命令符号表示如下：

表 2.7: chmod 作用对象表示法

| 符号 | 含义 |
|---|---|
| u | user 缩写，表示文件或者目录的所有者 |
| g | 文件所属群组 |
| o | others 的缩写，表示其他用户 |
| a | all 的缩写，是 u g o 三者的组合 |
| + | 添加权利 |
| - | 删除权利 |
| = | 赋予权利 |

如下示例：

```
$chmod u+x test.txt
--> 给test.txt 的user 添加可执行的权限
```

各种权限可以使用数字来表示，设置权限时可以直接使用数字进行设置:

```
chmod Owner+Group+Others file--> chmod 771 file
```

**umask** 设置文件的默认权限, 该值是需要减去的权限. 将给出的掩码与原文件的权限进行**异或运算**，即**如果掩码为 1，那么将该位置的权限擦除**

```
$umask 0002
--> 设置默认权限 0 000 000 010 : 即如果其他用户拥有写权限的话，将擦除该权限
```

如果 umask 为 022，所以 user 的权限并没有被拿掉什么权限，但是 group 和 others 的权限都被拿掉了 2，也就是写 (w) 权限。

```
$umask 022
 --> 创建文件时: （-rw-rw-rw-） - （-----w--w-） ==> -rw-r--r--
 --> 创建目录时: （drwxrwxrwx） - （d----w--w-） ==> drwxr-xr-x
```

**chown** 只有root 用户才可以使用该命令，更改文件所有者, 所有者必须是已经存**在系统中的帐号，**也就是在/etc/passwd 这个文件中有纪录的使用者名称才能改变。

**chgrp** 更改文件所属群组, 不过，请记得，要被改变的群组名称必须要在/etc/group 文件内存在才行，否则就会显示错误！

**passwd** 更改用户密码

## 2.10 权限-ACL

直接针对具体用户添加具体权限

**查看 ACL 命令** getfacl 选项 文件名

**设定 ACL 命令** setfacl 选项 文件名

**给特定用户添加对文件（or 目录）特殊权限** setfacl -m u:用户名:权限 文件或目录

```
// 示例，给用户李四 添加对目录 的读和进入权限
# useradd zhangsan
# useradd lisi
# useradd wangwu
# groupadd tgroup
# mkdir /project
# chown root:tgroup /project
# chmod 770 /project/
# setfacl -m u:lisi:rx /project
```

**给特定组添加对文件特殊权限** `setfacl -m g:组名:权限 文件或目录`

**ack 最大有效权限** 与 mask 相与得到的权限，使得给个别用户的权限不会太过。

**chattr** 改变文件属性

- `+` ：在原有参数设定基础上，追加参数。
- `-` ：在原有参数设定基础上，移除参数。
- `i` ：禁止用户修改，删除，改名
- `a` ：只能向文件中添加数据，而不能删除，多用于服务器日志文件安全，只有 root 才能设定这个属性。

**lsattr** 查看文件属性

## 2.11 权限-特殊

**Set UID** `- ..s ... ...`

- 只有**可执行文件**才能设定**SUID 权限**
- 命令执行者 要对该程序 拥有执行权限（x）
- 命令执行者 在执行改程序文件时 获得程序文件属主身份,`A->binary== binaryOwner->binary`
- SetUID 权限**只在程序执行过程中有效**，即*身份改变只在程序执行过程中有效*

普通用户执行特定用户的命令时，短暂时间内转化为特定用户，从而拥有一些只有特定用户拥有的权限，从而使用一些特定用户可以的动作。

只针对可执行权限 x. `x->s`, 即只有文件的 owner 位的权限如下形式才可以完成此操作。

```
-rws r-x r-x
```

**添加该权限：**`chmod u=rwxs file` 或 `chmod u+s file` 或 `chmod 4755 文件名`

其中数字 4 表示针对用户添加 SUID，后面的权限 755 意思不变，如果是组的话就是 2，是其他用户的话就是 1。

**Set GID** `- ... ..s ...`

- 只有**可执行文件或目录**才能设定**SGID 权限**
- 命令执行者　要对该程序　拥有执行权限（x）
- 命令执行者　在执行改程序文件时　组身份升级为程序文件的属组, 即原 A 组的 a 用户执行 B 组的程序，在执行期间身份自动提升为 B 组身份
- SetGID 权限**只在程序执行过程中有效**，即*身份改变只在程序执行过程中有效*

**设置 SGID**　`chmod g+s` 文件名 或 `chmod 2755` 文件名

**Sticky BIT** `- ... ... ..s`

- 粘着位只能**对目录生效**
- 普通用户对该目录拥有 w 和 x 权限，即可以进入并写入。
- 如果没有粘着位，因为普通用户拥有 w 权限，那么可以删除其他用户创建的文件。

  而如果赋予粘着位，除了 root 可以删除所有文件，普通用户就算拥有 w 权限，也只能删除自己建立的文件，并且不能删除其他用户建立的文件。

**设置粘着位**　`chmod o+t` 目录名 或 `chmod 1755` 目录名

**UID EUID**　http://keren.blog.51cto.com/720558/144908

**sudo 权限**

- root 把本来只能超级用户执行的命令赋予普通用户执行
- sudo 操作对象是系统命令

首先需要管理员在 `/etc/sudoers` 文件中利用`visudo` 命令完成对用户可以执行的命令赋予权限。

```
root ALL=(ALL) ALL
// 用户名 被赋予管理的地址=(可以使用的身份) 可执行的命令-绝对路径
sc ALL=/sbin/shutdown -r now
sc 192.168.0.156=/usr/bin/vim
```

```
// 不再输密码验证
sc localhost=NOPASSWD: /usr/bin/vim

%root ALL=(ALL) ALL
// 组名 被赋予管理的地址=(可使用的身份) 可执行命令的绝对路径
```

## 2.12 环境/etc

### 2.12.1 基本概念

login **shell**[需要输入用户名和密码的] 启动时会读取一个或多个启动文件以配置系统环境：

- `/etc/profile` 适用于所有用户的全局配置脚本
- `~/.bash_profile` 用户个人启动文件，可扩展或重写
- `~/.profile`

non-login **shell** 则会读取以下文件

- `/etc/bashrc` 适用于所有用户的全局配置脚本
- `~/.bashrc` 用户个人启动文件，可扩展或重写

### 2.12.2 相关命令

**printenv** 打印部分或全部的环境信息

表 2.8: 一些必要的环境变量

| 符号 | 含义 |
|---|---|
| DISPLAY | 运行图形界面环境时界面的名称 |
| EDITOR | 用于文本编辑的程序名称 |
| SHELL | 本机 shell 名称 |
| HOME | 本机主目录的路径名 |
| PWD | 当前工作目录 |
| USER | 用户名 |
| PATH | 以冒号分割的一个目录列表，**当用户输入一个可执行程序的名称时，会查找该目录列表** |

PATH 变量 通常时由启动文件 /etc/profile 中的一段代码设定，但并不如此，取决于系统的发行版本

```
// 这段代码将$HOME/bin 添加到PATH值的尾部
   PATH = $PATH:$HOME/bin
```

还有一个常见的命令 export, 该命令会告诉 shell, 将 shell 的子进程使用PATH 变量的内容

```
   export PATH
```

修改环境变量时应当把修改放入以下文件

- .bash_profile 或者 其他等效文件(如 Ubuntu中的 .profile)

其他的改变则应录入.bashrc, 如

```
// .bashrc 文件增加如下代码
   umask 0002
   export HISTSIZE = 1000
   alias ll = 'ls␣-l␣--color=auto'
```

但是只要启动 shell 时才会读取.bashrc, 所以对.bashrc 做出的修改只有在关闭 shell 终端回话并重启的时候才会生效。当然我们也可以使用以下命令强制命令 bash 重新读取.bashrc 文件

```
   $source .bashrc
```

**set** 设置 shell 选项

**export** 将环境导出到随后要运行的程序中

## 2.13 VI 编辑器

### 2.13.1 基本模式

- 普通模式
- 插入模式
- 可视模式
- 选择模式

- 命令行模式
- ex 模式


Fig 2.4: vi 插入示例

### 2.13.2 模式转换

模式如下，所有在**使用命令的时候一般需要按 Esc 按键返回到命令模式下才可以.**. 而并非在插入模式与 ex 转义方式下。通过输入 vi 的插入命令（i)、附加命令（a)、打开命令（o)、替换命令（s)、修改命令 (c）或取代命令（r）可以从命令方式进入输入方式。

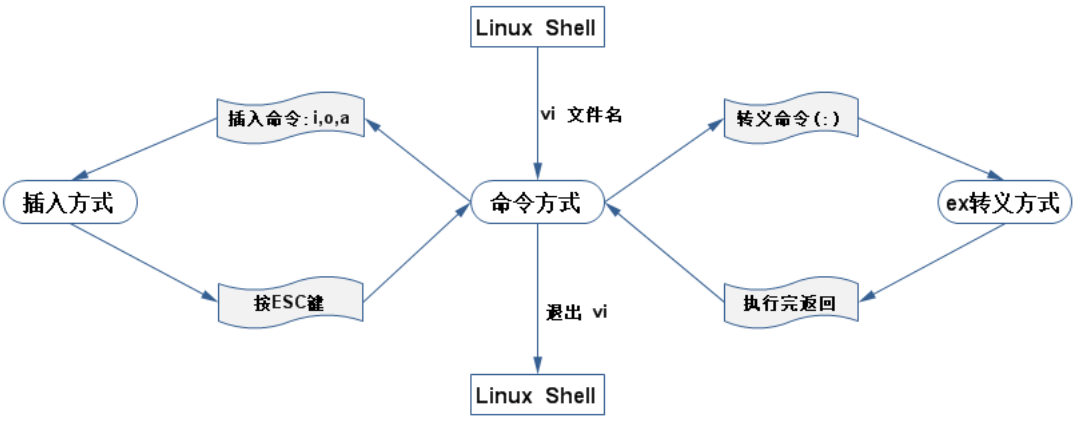



Fig 2.5: vi 模式转换示例

### 2.13.3 常用命令

http://blog.csdn.net/weixin_40910753/article/details/79077074

数字奥义(单，范围) + 命令 + 正则光标

**选中一块区域并缩进缩出 ->**

**可视模式** 按ESC 进入正常模式

1. 按v 进入视觉模式
2. 按shift，移动上下键选择文本
3. > 缩进，< 缩出

**正常模式** 按ESC 进入正常模式

数字的奥义 + >>|<<

**移动光标 ->**

按ESC 进入正常模式

**数字 0** 移动光标至本行开头，Home 也有此功能

**$** 移动光标至本行末尾，End 也有此功能

**gg** 移动光标到文件第一行

**G** 移动光标到文件最后一行

**b** 移动光标到该单词的第一个字母

**e** 移动光标到该单词的最后一个字母

**w** 移动光标到下一个单词的起始处

**num+G** 移动光标到第 num 行

删除操作 ->

按ESC 进入正常模式

**x** 删除当前字符

**d0** 删除当前光标到行开头字符

**d$** 删除当前光标到结尾字符

**dW** 删除整个单词

**dd** 删除当前行

**d+G** 删除当前行到文件末尾

**num+dd** 删除与本行在内的往后 num 行

数字 + motion = 重复多个 motion

**d+num+G** 删除当前行到第num 行

**:n1,n2d** 删除 n1 到 n2 行

撤销恢复 ->

**u** undo 操作

**ctrl+r** redo 操作

复制粘贴 ->

按ESC 进入正常模式

**y0** 复制当前光标到行开头字符

**y$** 复制当前光标到结尾字符

**yW** 复制整个单词

**yy** 复制当前行

**num+yy** 复制连同当前行往后的 num 行

**y+num+G** 复制当前行到文件第 num 行

**y+G** 复制当前行到文件末尾

**p** 粘贴复制文本到当前光标**前**

**P** 粘贴复制文本到当前光标**后**

**J** 合并后一行至当前行 https://stackoverflow.com/questions/2770739/vim-error-e492-not-ar

查找 ->

按ESC 进入正常模式

**'/keyword'** 在当前文本向后查找 keyword

**'?keyword'** 在当前文本向前查找 keyword

**n** 寻找下一个

**N** 寻找上一个

**%** 快速匹配括号

**全局搜索和替换 ->**

esc : 进入ex 模式

表 2.9: ex 搜索替换使用规范

| 符号 | 含义 |
|---|---|
| : | 分号用于启动一条 ex 命令 |
| % | 确定作用范围，% 代表从文件的第一行到最后一行，1,5 代表第一行到第 5 行，2,$ 代表从第二行到最后一行. 如果不指定范围的话，只会在当前行生效 |
| s | 指定了具体的操作- 本次操作为替换操作 (搜索和替换) |
| s/find/replace | 搜索 find 关机字，并将其替换为 replace |
| r | 取代光标所在处字符 |
| R | 从光标所在处开始替换字符，按 Esc 结束 |
| g | 替换正行 |
| c | 确认是否替换 |

:%s/find/replace/g 全文将 find 替换为 replace

:1,4s/find/replace/gc 将第一行到第 4 行的 find 全部替换为 replace, 并询问确认意见

http://blog.sina.com.cn/s/blog_736f1c59010136ry.html

**查找删除注释 ->**

**'s/old/new/ '** 用 old 替换 new，替换当前行的第一个匹配

**'s/old/new/g '** 用 old 替换 new，替换当前行的所有匹配

**'%s/old/new/'** 用 old 替换 new，替换所有行的第一个匹配

**'%s/old/new/g'** 用 old 替换 new，替换整个文件的所有匹配

**',5 s//#/g'** 注释当前行到第 5 行

**',+5 s//#/g'** 注释当前行与之后的 5 行

**'3,5 s/#//g'** 解除 3-5 行的注释

**查看目录下有什么文件 ->**

`:! command`

`:! ls /` 列出根目录文件

**文件保存 ->**

`esc : 进入ex 模式`

**:w** 保存修改

**:wq** 保存修改并退出

**:q!** 不保存修改退出

**:wq!** 强行保存并退出（文件所有者和 root 可以使用）

**局部另存为 ->**

- `esc v` 进程可视模式
- `shift` 选中文本
- `:w newFileName` 将选中的另存为 newFileName

**文件覆盖** `:w! oldFile` 将 oldfile 用现有内容覆盖

**文件合并** `:r file` 将 file 文件的内容合并到当前文件光标的下方。

技巧：结合重定向技巧将 bash 命令执行结果写入文件中如`:r !date`

编辑多文本 -> https://www.cnblogs.com/end/archive/2012/06/12/2546757.html

**vi file1 file2**

- -o 垂直并排
- -O 水平并排
- ctrl + w + w 切换到下一个文件
- ctrl + w + 方向键 切换到相对位置的文件
- a 作用于所有，在 ex 模式下使用:wa :qa

**在不同窗口中打开多个文件** 如果已经打开一个了一个文件，

- esc : 进入命令模式
- :sp newFile 水平分割继续打开第二个文件
- :vsp newFile 纵向分割打开第二个文件

或者用 vi -o file1 file2 file3.... 用**分割屏幕窗口方式同时打开多个文件**。

**分屏操作**

- ctrl+w s 对当前文档内容分屏显示
- ctrl+w q 关闭所处分屏
- ctrl+w o 仅显示当前分屏内容

**n 与 N** 切换下一个文件

**:buffers** 显示当前打开的文件们

**:buffer num** 切换到文件 num

**:e newfile** [* 常用] 载入更多的文件，然后使用ctrl + ^ 在文件之间切换

- :ls 展示全部的打开文档
- :xn 切换到向下的第 x 个文档
- n ctrl+6 切换到第 n 个 buffer
- ctrl+6 或 :e# 回到前一个编辑文件

**:tabe** http://blog.csdn.net/achang21/article/details/44562253

**文件间复制** 跟普通复制粘贴一样，只是需要切换目标文件先

**插入整个文件** :r insertFile 将 insertFile 的内容插入到光标位置之前

### 2.13.4 配置

https://segmentfault.com/a/1190000011466454

**.vimrc**

.vimrc 是 Vim 的配置文件，需要我们自己创建

```
cd Home // 进入 Home 目录
touch .vimrc // 配置文件

# 安装插件管理工具 vim-plug
# Unix
# Vim
curl -fLo ~/.vim/autoload/plug.vim --create-dirs \
https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim
# Neovim
curl -fLo ~/.local/share/nvim/site/autoload/plug.vim --create-dirs \
https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim
```

**基本配置**

- 文件编码 set encoding=utf-8
- 显示行号 set number

- 显示光标当前位置 `set ruler`
- 设置缩进

```
set cindent
set tabstop=2
set shiftwidth=2
```

- 突出显示当前行 `set cursorline`

**插件配置**

**树形目录** Plug 'scrooloose/nerdtree'，使用`shit+r` 动态刷新目录。

**代码补全** Plug 'Valloric/YouCompleteMe'

**git 显示** Plug 'airblade/vim-gitgutter'

### 2.13.5 常用插件

### 2.13.6 vundle 插件管理器

http://blog.csdn.net/myloveqingmu/article/details/52518563

http://blog.csdn.net/mergerly/article/details/51671890

http://blog.csdn.net/jack__cj/article/details/52671312

**ctags**

安装：http://blog.csdn.net/u013445530/article/details/46726109

tags 下载：https://wiki.geany.org/tags/start

1. 安装软件：`sudo apt-get install ctags`
2. 下载补全资源：`.h`
3. 生成 tags 文件：ctag 生成 `tags` 文件
4. 使用：`ctrl+p` 与 `ctrl+n` 使用

### 2.13.7 Gtags

### 2.13.8 NERDTree

### 2.13.9 Tagbar

**YouCompleteMe**

## 2.14 存储介质挂载

**挂载点的意义** 每个 filesystem 都有独立的 inode、 block 、superblock 等信息，**这个文件系统要能够链接到目录树才能被我们使用。将文件系统与目录树结合的动作我们称为“挂载”。**

重点是：**挂载点一定是目录**，该目录为进入该文件系统的入口。因此并不是你有任何文件系统都能使用，必须要“挂载”到目录树的某个目录后，才能够使用该文件系统的。

挂载

- windows -> 硬件设备映射为一个盘符
- linux -> 将硬件设备挂载成 (映射到) 一个目录

linux 的每个挂载了分区的目录就相当于 windows 系统中的盘符，比如/home/ftp 和 /oracle 目录我们就可以把她看做一个盘符和一个分区关联

http://blog.csdn.net/gongweijiao/article/details/8425629

**mount** 查询系统中已经挂载的设备

mount [-t 文件系统] [-o 特殊选项] 设备文件名 挂载目录

- -a 依据配置文件/etc/fstab 的内容，开机自动挂载
- -t 文件系统类型可以有 ext3, ext4, fat16, fat32 等
- -o 可以指定挂载的额外选项

-> 挂载U盘: 此盘在 windows 下格式化为 fat32 文件系统

1. 挂 u 盘之前，运行命令cat /proc/partitions, **看看现在系统中有哪些分区**。插上 u 盘以

后，再次运行上述命令，看看多出来什么分区（通常是 sda1 或者 sdb1)

2. 输入 `fdisk -l /dev/sda` 查看输出结果, **决定以什么文件系统挂载**。

```
#fdisk -l /dev/sda
 Disk /dev/sda: 131 MB, 131104768 bytes
 3 heads, 32 sectors/track, 2667 cylinders
 Units = cylinders of 96 * 512 = 49152 bytes
 Device Boot      Start         End      Blocks   Id System
/dev/sdb1   *           1        2668      128016    6 FAT16
```

3. 根据文件系统类型挂载，如`mount -t msdos /dev/sdb1 /mnt/usb`

   可以使用`cat /proc/filesystems` 查看当前系统支持的文件系统格式

   - 如果是`fat16` 格式，就用命令： `mount -t msdos /dev/sdb1 /mnt/usb`
   - 如果是`fat32` 格式,就用命令：`mount -t vfat /dev/sdb1 /mnt/usb -o iocharset=utf8`
   - 如果是`ext2` 格式，就用命令： `mount -t ext2 /dev/sda1 /mnt/usb`
   - 如果是`NTFS` 格式，就用命令： `mount -t ntfs /dev/sda1 /mnt/usb`

挂载实例：https://wenku.baidu.com/view/ef8aa226dd36a32d73758139.html

**umount** 如果需要挂载其他盘，则需要卸载`umount` 设备名

## 2.15 网络

**write** 给指定**在线用户**发信息，以`Ctrl+D` 保存结束

`write` 用户名 消息 `Ctrl-D`结束输入

**wall** 给所用在线用户发送消息，广播

`wall` 消息

**ping** **向网络主机发送特殊数据包** (`ICMP ECHO_REQUEST`). 多数网络设备收到该数据包后会做出回应，通过此法可判断网络链接 (两者间) 是否正常. `ping baidu.com`

指定 ping 的**次数**：`ping -c 3 baidu.com` ping 3 次百度。

**ifconfig** 查看和设置网卡信息

**查看** `ifconfig`

**设置** `ifconfig eth0 192.168.1.12`

`service network restart`

**mail** 查看发送电子邮件

发送：`mail 用户 消息 Ctrl-D结束`

接受：`mail`

- 查看，输入序列号即可
- 删除，d 序列号

**last** 列出目前与过去登入系统的用户信息->`last`

**traceroute** **跟踪网络数据包的传输路径**，会显示文件通过网络从本地系统到指定主机过程中所有停靠点的列表。`traceroute slashdot.org`

**netstat** 检查网络设置以及相关统计数据，`-ie` 可以检查系统的网络接口信息，`-r` 可以显示内核的网络路由表 http://www.cnblogs.com/ggjucheng/archive/2012/01/08/2316661.html

| netstat | options |
|---|---|
| netstat | -a (all) 显示所有选项，默认不显示LISTEN 相关 |
| | -t (tcp) 仅显示tcp 相关选项 |
| | -u (udp) 仅显示udp 相关选项 |
| | -n 拒绝显示别名，能显示数字的全部转化成数字 |
| | -l 仅列出有在 Listen (监听) 的服務状态 |
| | -p 显示建立相关链接的程序名 |
| | -r 显示路由信息，路由表 |
| | -e 显示扩展信息，例如uid 等 |
| | -s 按各个协议进行统计 |
| | -c 每隔一个固定时间，执行该netstat 命令 |

LISTEN和LISTENING 的状态只有用-a 或者-l 才能看到

- 列出所有 tcp 端口: netstat -at
- 列出所有 udp 端口: netstat -au
- 只列出所有处于监听状态 tcp 端口：netstat -lt
- 显示tcp 和进程名称 netstat -pt

**ftp:lftp:sftp** File Transfer Protocol,ftp 服务器就是那些包含供网络上传、下载文件的机器

表 2.10: FTP 使用说明

| 命令 | 含义 |
| --- | --- |
| ftp server | 建立 FTP 连接，在命令上中先输入ftp 然后空格跟上 FTP 服务器的域名 'domain.com' 或者 IP 地址 |
| anonymous | 使用用户名密码登录, 绝大多数的 **FTP 服务器是使用密码保护的**，因此这些 FTP 服务器会询问'username' 和'password'. 如果你连接到被称作匿名 FTP 服务器，可以尝试anonymous 作为用户名以及使用空密码：Name: anonymous, Password: |
| 目录操作 | FTP 命令可以列出、移动和创建文件夹，如同我们在本地使用我们的电脑一样。ls 可以打印目录列表，cd 可以改变目录，mkdir 可以创建文件夹 |
| 使用 FTP 下载文件 | 在下载一个文件之前，我们首先需要使用lcd 命令**设定本地接受目录位置**。lcd /home/user/yourdirectoryname 如果你不指定下载目录，文件将会下载到你登录 FTP 时候的工作目录。现在，我们可以使用命令 get **来下载文件**，比如：get file **文件会保存在**使用lcd 命令设置的目录位置 |
| 使用 FTP 上传文件 | 使用 put 命令上传文件：put file |
| 关闭 FTP 链接 | quit exit bye3 个命令都可以 |

**wget** 非交互式网络下载工具 wget http://www.linuxde.net/testfile.zip

*** 远程操作 ssh** 安全登录远程计算机:SSH 解决了与远程主机进行安全通信的两个基本问题：1、该协议可以验证远程主机的身份是否真实;2、该协议将本机与远程主机之间的通信内容全部加密

SSH 协议包括 2 部分：1、运行在远程主机上的 SSH 服务端，用来监听端口 **22** 上可能过来的连接请求;2、是本地系统上的 SSH 客户端，用来与远程服务器进行通信

关于 SSH 的相关命令有 scp，sftp, 分别表示远程拷贝和更简单高效快速的 ftp

例如：执行ssh happycast.net 命令，然后输入密码，接着就可以直接访问该服务器.. 但是每次连接都得输入密码我们可以使用以下步骤进行简化：

```
$ssh-keygen
$cd ~/.ssh
$ls
id_rsa id_rsa.pub

// 然后我们把公钥id_rsa.pub 上传至服务器/home/Usr1/.ssh/authorized_keys即可
$ssh-copy-id 服务器名(Usrl@happycasts.net)
```

*** 数据传输 rsync** 见归档备份

**nc** netcat 所做的就是在两台电脑之间建立链接并返回两个数据流，在这之后所能做的事就看你的想像力了。你能建立一个服务器，传输文件，与朋友聊天，传输流媒体或者用它作为其它协议的独立客户端。

http://blog.csdn.net/zhangxiao93/article/details/52705642

- 端口扫描
- ChatServer
- 文件传输

**pv** 是 Pipe Viewer 的简称，意思是通过管道显示数据处理进度的信息。这些信息包括已经耗费的时间，完成的百分比（通过进度条显示），当前的速度，全部传输的数据，以及估计剩余的时间。

http://www.jb51.net/LINUXjishu/409870.html

**tcpdump** http://www.cnblogs.com/ggjucheng/archive/2012/01/14/2322659.html

- 监视指定主机的数据包
  - 打印所有**进入或离开**sundown 的数据包:

    tcpdump host sundown 或 tcpdump host 210.27.48.1
  - 打印 helios **与** hot **或者与** ace 之间通信的数据包

tcpdump host helios and \( hot or ace \) 或

tcpdump host 210.27.48.1 and \ (210.27.48.2 or 210.27.48.3 \)

– 打印 ace 与任何不是helios 主机通信的 IP 数据包

tcpdump ip host ace and not helios 或

tcpdump ip host 210.27.48.1 and ! 210.27.48.2

– 截获主机hostname 发送的所有数据

tcpdump -i eth0 src host hostname

– 监视所有送到主机hostname 的数据包

tcpdump -i eth0 dst host hostname

- 获取主机10.2.4.1 在端口 23 上接收或发出的包:tcpdump tcp port 23 and host 10.2.4.1
- () 得使用转义字符进行，要么不识别

tcpdump \(src 172.17.14.98 and dst 172.17.15.112\) or \(src 172.17.15.112 and dst 172.17

**iperf** 可以测试TCP和UDP 带宽质量。iperf 可以报告带宽，延迟抖动和数据包丢失。

**ifconfig**

**iptables**

**vmstate** vmstat 命令是最常见的 **Linux/Unix 监控工具**，可以展现给定时间间隔的服务器的状态值, 包括服务器的 **CPU 使用率，内存使用，虚拟内存交换情况,IO 读写情况**

-> 使用:

vmstat [-a] [-n] [-S unit] [delay [ count]]

vmstat [-s] [-n] [-S unit]

vmstat [-m] [-n] [delay [ count]]

vmstat [-d] [-n] [delay [ count]]

vmstat [-p disk partition] [-n] [delay [ count]]

vmstat [-f]

| 参数 | 含义 |
| --- | --- |
| delay | 刷新时间间隔。如果不指定，只显示一条结果。 |
| count | 刷新次数。如果不指定刷新次数，但指定了刷新时间间隔，这时刷新次数为无穷。 |
| -a | 开启显示 active/inactive memory |
| -f | 显示此系统启动以来的 forks 的总数，包括 fork、vfork 和 clone system calls |
| -m | 显示 slabinfo 信息 |
| -n | 只显示头信息，不周期性显示. 也就是说开启这个参数，只显示头部信息一次。 |
| -s | 显示各种事件计数器表和内存统计信息，这显示不重复 |
| -d | 显示磁盘统计数据 |
| -w | 可以扩大字段长度，当内存较大时，默认长度不够完全展示内存。 |
| -p | 显示磁盘分区数据 |
| -S | 参数 S 控制输出性能指标的单位，k(1000) K(1024) 或 M(1048576) 默认单位为 K（1024 bytes） |

http://www.cnblogs.com/tommyli/p/3746187.html

http://www.cnblogs.com/ggjucheng/archive/2012/01/05/2312625.html

## 2.16 通配符

通配符是用于匹配符合条件的**文件名**,通配符是**完全匹配**。包括任意个* 一个？括号中的一个[x]

ls、find、cp 这些命令不支持正则表达式，所以只能使用通配符来进行匹配。

## 2.17 正则表达式

正则表达式用来在**文件中**匹配符合条件的**字符串**，正则是**包含匹配**。

grep、awk、sed 等命令支持正则表达式

$$\text{正则表达式} = \begin{cases} \text{字符类} \\ \text{数量限定类} \\ \text{位置限定类} \end{cases}$$

http://www.cnblogs.com/hanxiaoyu/p/5759477.html

### 2.17.1 字符类

| 字符 | 含义 | 例子 |
| --- | --- | --- |
| . | 匹配任意一个字符 | abc. 可以匹配 abcd、 abc9 |
| [] | 匹配括号中的任意一个字符 | [abc]d 可以匹配 ad、 bd、cd |
| - | 在[] 中使用，表示字符范围 | [0-9 a-f A-F] 可以匹配一位 16 进制数字 |
| ^ | 位于括号内的开头，匹配括号中的字符外的任意一个字符 | [^xy] 可以匹配除x、y 之外的任一字符 |
| [[:xx:]] | 预定义的一些命名字符 | [[:digit:]] 匹配一个数字 |

[\d] 等价于 [0-9]

/ 只是在某些语言中作为正则的**边界符**, 如 sed 在匹配正则时，不仅要指定 r, 正则还要以'/' 开头

### 2.17.2 数量限定类

| 字符 | 含义 | 例子 |
| --- | --- | --- |
| ? | 紧跟在他前面的单元应匹配 0 次或 1 次 | [0-9]?\.[0-9] 匹配 0.0 2.3 .5 等，特殊字符. 需要加转义字符 |
| + | 紧跟在他前面的单元应匹配 1 次或多次 | [a-zA-Z0-9_-]+ @ [a-zA-Z0-9_.-]+ \.[a-zA-Z0-9_.-]+ 匹配 email 地址 |
| * | 紧跟在他前面的单元应匹配 0 次或多次 | [0-9]* 匹配至少 1 位数字 |
| {N} | 紧跟在他前面的单元应精确匹配 N 次 | [1-9][0-9]{2} 匹配 100 到 999 的整数 |
| {N,} | 紧跟在他前面的单元应匹配至少 N 次 | [1-9][0-9]{2,} 匹配 3 位及 3 位以上的整数 |
| {,M} | 紧跟在他前面的单元应匹配最多 M 次 | [0-9]{,1} 最多匹配一次数字 |
| {N,M} | 紧跟在他前面的单元应匹配至少 N 次, 最多 M 次 | [0-9]{1,3} 表示 0-9 数字至少匹配 1 次，最多匹配 3 次 |

### 2.17.3 位置限定类

| 字符 | 含义 | 例子 |
| --- | --- | --- |
| ^ | 匹配行首的位置 | ^content 匹配行首为 content 的行 |
| $ | 匹配行末的位置 | ;$ 匹配行末尾为 ; 的行，^$ 匹配空行 |
| \< | 匹配单词开头的位置 | \<th 匹配.. this.., 但不匹配 tenth 等 |
| \> | 匹配单词结尾的位置 | p\> 匹配 leap, 但不匹配 parent |
| \b | 匹配单词的开头\bx 或结尾x\b 的位置 | \b at 匹配 at, 但不匹配 batch |
| \B | 匹配非单词开头\Bx 或结尾的位置 | \B at 匹配 battery, 但不匹配 attend、hat |

### 2.17.4 特殊字符

| 字符 | 含义 | 例子 |
| --- | --- | --- |
| \ | 转义字符 | ([0-9]{1,3}\.){3}[0-9]{1,3} 匹配 IP 地址 |
| () | 将正则表达式的一部分括起来组成一个单元，可以对整个单元使用数量限定符 | |
| \| | 链接两个子表达式，表示或的关系 | |

## 2.18 搜索命令

*** 文件搜索命令 locate** 通过 [部分]文件名 查找**文件**的具体位置，搜索速度比find 快。在后台数据库中按文件名搜索，搜索数据速度更快，但是后台数据库更新一般是 1 天更新一次，但是可以使用updatedb 强制更新后台数据库。

但是使用updatedb 时遵循/etc/updatedb.conf 配置文件的规则

- 开启搜索限制 PRUNE_BIND_MOUTS = "yes"
- 搜索时，不搜索的文件系统 PRUNEFS =
- 搜索时，不搜索的文件类型 PRUNENAMES =
- 搜索时，不搜索的路径 PRUNEPATHS =，默认包含\tmp

当配置文件开启搜索规则时，update 命令执行后，locate 会在搜索时遵照 updatedb.conf 规则进行剔除搜索。

选项->

- -i：忽略大小写
- -c：不输出文件名，仅计算找到的文件数量
- -l：仅输出结果的几行，例如输出 5 行则是 -l 5

*** 文件搜索命令 find** 在文件系统目录框架中查找文件 (**完全匹配**)

```
find path -option [-print -delete -ls -quit] [[-exec -ok] command{}[; +]]
```

| find | options | 含义 |
|---|---|---|
| find | -name filename | 查找名为filename 的文件 |
| | -perm | 按执行权限来查找 |
| | -user username | 按文件属主来查找 |
| | -group groupname | 按组来查找 |
| | -mmin -n +n | 按文件内容修改时间来查找文件 (分钟) -n 多少分钟以内 |
| | -mtime -n +n | 按文件更改时间来查找文件，-n 指n 天以内，+n 指n 天以前 |
| | -atime -n +n | 按文件访问时间来查 |
| | -ctime -n +n | 按文件属性修改时间来查找文件，-n 指n 天以内，+n 指n 天以前 |
| | -newer f1 !f2 | 查更改时间比f1 新但比f2 旧的文件 |
| | -type b/d/c/p/l/f | 查是块设备 b、目录 d、字符设备 c、管道 p、符号链接 l、普通文件 f |
| | -size n[c] | 查长度为n 块 [或n 字节] 的文件,-小于size的，+大于size的, find /etc -size +20k -a -size -50k |
| | -inum fileNum | 查找文件节点号为 fileNum 的文件 |
| | -mount | 查文件时不跨越文件系统mount 点 |
| | -and -or | -and 条件与 -or 条件或 |

- -print 将查找到的文件输出到标准输出
- -exec command {} \; —–将查到的文件执行 command 操作,{} 为查找的结果，也是新命令的参数。
- -ok 和-exec 的作用相同，只不过以一种更为安全的模式来执行该参数所给出的 shell 命令，在执行每一个命令之前，都会给出提示，让用户来确定是否执行。

**通配符***任意内容　？任意一个字符　[]任意一个括号内的字符

http://www.cnblogs.com/wanqieddy/archive/2011/06/09/2076785.html

*** 字符串搜索命令 grep**　把匹配的**行**打印出来。使用正则的都是**包含匹配**，grep 也不例外

-> grep "正则"　文本

-> |grep 选项　字符串Or正则

| grep | options | regex pattern |
|---|---|---|
| | -c 只输出匹配行的计数<br>-i 不区分大小写 (只适用于单字符)<br>-e 并列使用多个 -e 参数可以实现**或**条件<br>-h 查询多文件时不显示文件名<br>-l 查询多文件时只输出包含匹配字符的文件名<br>-n 显示匹配行及行号<br>-s 不显示不存在或无匹配文本的错误信息<br>-v 不显示包含匹配文本的所有行<br>-r 递归的对目录下的所有文件（包括子目录）进行 grep | \　忽略正则表达式中特殊字符的原有含义<br>^　匹配正则表达式的开始行<br>netstat -an \|grep -e EST -e WAIT<br>$　匹配正则表达式的结束行<br>\<　从匹配正则表达式的行开始<br>\>　到匹配正则表达式的行结束<br>[ ] 单个字符，如 [A] 即 A 符合要求<br>*　有字符，长度可以为 0<br>[ - ] 范围，如 [A-Z]，即 A、B、C 一直到 Z 都符合要求 |

cat /etc/passwd | grep /bin/bash | grep -v root: 在 passwd 文件中截取默认登录 shell 为/bin/bash 即普通用户，然后并包包含 root 用户.

**命令搜索命令 whereis 与 which**　which 指令会在环境变量$PATH 设置的目录里查找符合条件的文件。

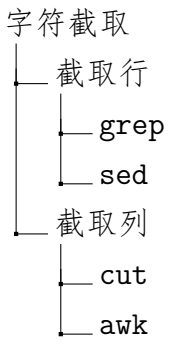
whereis 指令会在特定目录中查找符合条件的文件。这些文件的属性应属于原始代码，二进制文件，或是帮助文件。

**xargs**　从标准输入中建立、执行命令行, 一般结合管道 | 在结果中使用。

```
find ~ -type f -name 'foo*' -print |xargs ls -l
```

**stat** 显示文件或文件系统的状态

## 2.19 文本相关




**sed** sed 是一个很好的文件处理工具，本身是一个管道命令，主要是**以行为单位进行处理**，可以将数据行进行替换、删除、新增、选取等特定工作

```
sed [options] 'command' 输入文本
```

**选项**

- -n□ 使用安静 (silent) 模式。在一般 sed 的用法中，所有来自 STDIN 的资料一般都会被列出到萤幕上。但如果加上 -n 参数后，则只有经过 sed 特殊处理的那一行 (或者动作) 才会被列出来。
- -e□ 直接在指令列模式上进行 sed 的动作编辑；
- -f□ 直接将 sed 的动作写在一个档案内，-f filename 则可以执行 filename 内的 sed 动作；
- -r□sed 的动作支援的是延伸型正规表示法的语法。(预设是基础正规表示法语法)
- -i□ 直接修改读取的档案内容，而不是由萤幕输出。

**命令**

- a □ **新增**，a 的后面可以接字串，而这些字串会在新的一行出现 (目前的下一行)
- c □ **取代**，c 的后面可以接字串，这些字串可以取代 n1,n2 之间的行！
- d □ **删除**，因为是删除啊，所以 d 后面通常不接任何咚咚；

- i □ **插入**，i 的后面可以接字串，而这些字串会在新的一行出现 (目前的上一行)；
- p □ **列印**，亦即将某个选择的资料印出。通常 p 会与参数 sed -n 一起运作
- s □ **取代**，可以直接进行取代的工作哩！通常这个 s 的动作可以搭配正规表示法！例如 1,20s/old/new/g

http://www.cnblogs.com/dong008259/archive/2011/12/07/2279897.html

**awk** awk '条件1{动作1} 条件2{动作2}...' 文件名，先读入一行，然后逐行执行操作。

**条件 Pattern** 一般使用关系表达式作为条件

- x > 10
- x >= 10
- x <= 10

**动作 Action**

- 格式化输出
- 流程控制语句

**例子**

- awk ' {printf $2 "\t" $6 "\n"}' student.txt :patten 省略，表示对每行都执行，显式文件中的第 2 列和第 6 列。
- df -h|awk '{print $1 "\t" $3}' : 截取 df 输出的第 1 列和第三列，与 cut 不同的是，awk 无须指定分割符，并且需要注意的是，print 会自动分行，但是格式控制与 printf 相同。
-

**BEGIN** 在其他所有命令逐行执行前，执行一次 BEGIN 后的动作

```
awk 'BEGIN{print "Test awk BEGIN"} {print $2 "\t" $5}' student.txt
```

这条命令首先执行 BEGIN 后的 print Anction, 然后逐行读取数据执行之后的条件与动作。

**FS 内置变量**　指定分隔符

```
awk '{FS=":"}{print $1"\t"$3}'  /etc/passwd
```

这条命令意思是指定分割符为:，但是在读入第一行后，再执行后面 Action 时，指定分隔符已经来不及了，所以这种情况第一行时不会正确使用分割符分割的，而是从第 2 行开始的，为了解决这个问题，可以利用前面提到的 BEGIN 解决。

```
awk 'BEGIN{FS=":"}{xx}'  xx
```

**END**　在所有命令执行完后，执行一次 END 后的动作

```
awk '{print $1"\t"$3}  END{print"The End"}'  student.txt
```

**关系运算**　`cat fileName.txt | grep -v Name | awk '$6 >= 87 {print $2}'`

**cat**　连接文件并打印到标准输出

- -A, –show-all 等价于 -vET
- -b, –number-nonblank 对非空输出行编号
- -e 等价于 -vE
- -E, –show-ends 在每行结束处显示 `$`
- -n, –number 对输出的所有行编号
- -s, –squeeze-blank 不输出多行空行
- -t 与 -vT 等价
- -T, –show-tabs 将跳字符显示为 `^I`

**sort**　对**文本行**排序..

- `-r` 默认升序，使用-r 选项使用降序
- `-t` 设定分隔符，如存在按列分割，然后按照第几列排序时，需要指定分割符
- `-k` 按照第几列排序
- `-u` 去除重复行

- -n 按照数值排序，解决按照字符串排序的尴尬（1 > 10 > 2）, 加了-n 后 (1,2,10)

**wc** 对**文本**进行统计 wc 选项 文件

- -1 只统计行数
- -w 只统计单词数
- -m 只统计字符数

**uniq** 报告并省略重复行

**cut** cut 选项 文件名，有局限，仅支持制表符、和常用分隔符（,:。-），不支持空格。

- -f列号 提取第几列
- -d分隔符 按照指定分隔符分割列

cut -f 2 student.txt 提取文件第 2 列

cut -f 2,3 student.txt 提取文件第 2 列和第三列

cut -d ":" -f 1,3 /etc/passwd 截取第一列和第三列，并且分隔符号为：

**paste** 合并文件文本行

**join** 基于某个共享字段来联合两个文件的文本行

**comm** 逐行比较两个已经排序好的文件

**diff** 逐行比较文件

**patch** 对原文件打补丁

**tr** 转换或删除字符

**aspelx** 交互式拼写检查器

## 2.20 归档备份

### 2.20.1 压缩与解压缩

常见格式有.zip .gz .bz2 .tar.gz .tar.bz2

- .zip
  - **压缩**
    1. zip 压缩文件名 源文件：压缩文件
    2. zip -r 压缩文件名 源目录：压缩目录
  - **解压缩** unzip 压缩文件
- .gz
  - **压缩**
    1. gzip 源文件：压缩为.gz 格式的压缩文件，源文件会消失
    2. gzip -c 源文件 > 压缩文件：压缩为.gz 格式，源文件保留
    3. gzip -r 目录：压缩目录下所有子文件，但是不能压缩目录
  - **解压缩**
    1. gzip -d 压缩文件：解压缩文件
    2. gunzip 压缩文件：解压缩文件
    3. 解压目录加-r 即可
- .bz2
  - **压缩**，不能压缩目录
    1. bzip2 源文件：压缩为.bz2 格式，不保留源文件
    2. bzip2 -k 源文件：压缩后保留原文件
  - **解压缩**，加关键字-k 保留压缩文件
    1. bzip2 -d 压缩文件
    2. bunzip2 压缩文件
- **打包** tar -cvf 打包文件名 源文件

1. -c 打包
2. -v 显示过程
3. -f 指定打包后的文件名

`tar -cvf long.tar long`

将文件直接打包后压缩为.tar.gz 格式，-z 前缀

1. -z **压缩**为.tar.gz:tar -zcvf 压缩文件名　原文件
2. -x **解压缩**.tar.gz:tar -zxvf 压缩包名

将文件直接打包后压缩为.tar.bz2 格式，-j 前缀

1. -z **压缩**为.tar.bz2:tar -jcvf 压缩文件名　原文件
2. -x **解压缩**.tar.bz2:tar -jxvf 压缩包名
3. -C 解压到指定路径下tar -jxvf test.tat.bz2 -C /tmp/

### 2.20.2 同步文件和目录

**rsync** 在对 rsync 服务器配置结束以后，下一步就需要在客户端发出 rsync 命令来实现将服务器端的文件备份到客户端来，但是也可以同步本地的文件夹等，命令格式如下：

```
rsync [OPTION]... SRC DEST
rsync [OPTION]... SRC [USER@]HOST:DEST
rsync [OPTION]... [USER@]HOST:SRC DEST
rsync [OPTION]... [USER@]HOST::SRC DEST
rsync [OPTION]... SRC [USER@]HOST::DEST
rsync [OPTION]... rsync://[USER@]HOST[:PORT]/SRC [DEST]
```

对应于以上六种命令格式，rsync 有六种不同的工作模式：

- 拷贝本地文件。当SRC 和DES 路径信息都不包含有单个冒号”:”分隔符时就启动这种工作模式。如：rsync -a /data /backup
- 使用一个远程 shell 程序 (如 rsh、ssh) 来实现将本地机器的内容拷贝到远程机器。当DST 路径地址包含单个冒号”:”分隔符时启动该模式。如：rsync -avz *.c foo:src
- 使用一个远程 shell 程序 (如 rsh、ssh) 来实现将远程机器的内容拷贝到本地机器。当SRC 地址路径包含单个冒号”:”分隔符时启动该模式。如：rsync -avz foo:src/bar /data

- 从远程 rsync 服务器中拷贝文件到本地机。当SRC 路径信息包含”::”分隔符时启动该模式。如：`rsync -av root@172.16.78.192::www /databack`
- 从本地机器拷贝文件到远程rsync 服务器中。当DST 路径信息包含”::”分隔符时启动该模式。如：`rsync -av /databack root@172.16.78.192::www`
- 列远程机的文件列表。这类似于rsync 传输，不过只要在命令中省略掉本地机信息即可。如：`rsync -v rsync://172.16.78.192/www`

http://www.cnblogs.com/subsir/articles/2565373.html

### 2.20.3 备份

**备份策略** 完全备份、**增量备份**、差异备份（与完全备份相比）

**dump** 增量备份：dump 选项 备份之后的文件名 原文件或目录

- -level :0-9 备份级别,0 表示基本备份，以后增加 level，表示第多少次增量备份，并且**目录不支持增量备份**
- -f : 指定备份之后的文件名
- -u : 备份成功后，把备份时间记录在/etc/dumpdates 文件
- -v : 显式备份过程中输出的信息
- -j : 调用 bzlib 库压缩备份文件.bz2
- -W : 显式运行被 dump 的分区的备份等级及备份时间

**restore** restore [模式选项] [选项]

- -C : 比较备份数据和实际数据的变化
- -i : 进入交互模式，手工选择需要恢复的文件
- -t : 查看模式，用于查看备份文件中拥有哪些数据
- -r : 还原模式，数据还原
- -f : 指定备份文件的文件名

## 2.21 进程管理

**ps** 显示当前所有进程的运行情况，显式格式为USER PID %CPU %MEM VSZ TSS TTY ...，具体表示为

- USER ： 该进程由哪个用户产生的
- PID ： 该进程的 ID 号
- %CPU ： 该进程占用 CPU 资源百分比
- %MEM ： 该进程占用物理内存百分比
- VSZ ： 该进程占用虚拟内存百分比
- RSS ： 该进程占用实际物理内存的大小
- TTY ： 该进程在哪个终端中运行的
- STAT ： 进程状态。R 运行、S 睡眠、T 停止、s 包含子进程、+ 位于后台
- START ： 该进程的启动时间
- TIME ： 该进程占用 CPU 的运算时间，注意不是系统时间
- COMMAND ： 产生此进程的命令名

将程序运行至后台..xxCommand &

表 2.11: ps 参数含义

| 符号 | 含义 |
|---|---|
| l | 长格式输出 |
| u | 按用户名和启动时间的顺序来显示进程 |
| j | 用任务格式来显示进程 |
| f | 用树形格式来显示进程 |
| a | 显示所有用户的所有进程 (包括其它用户) 如：ps a 显示现行终端机下的所有程序 |
| x | 显示无控制终端的进程 |
| r | 显示运行中的进程 |

参考：http://www.cnblogs.com/wangkangluo1/archive/2011/09/23/2185938.html

**top** 查看系统健康状态，**实时动态显示**当前所有任务的资源占用情况

- 系统信息
- 进程信息
- CPU 信息
- 物理内存信息
- 交换区信息
- ps 信息

**pstree** 进程树，显式进程的父子关系

**kill** 发送信号给某个进程，kill 选项 进程号

**kill -l** 查看可用的进程**信号**。常用信号如下：

- SIGHUP 1：该信号让进程立即关闭，然后重新读取配置文件之后重启
- SIGINT 2：程序终止信号，用于终止前台进程。相当于 <Ctr-c>
- SIGFPE 8：算术致命运算错误
- SIGKILL 9：用于立即结束程序的运行，本信号不能被阻塞、处理。一般用于强制终止进程
- SIGALRM 14：时钟定时信号，alarm 函数使用该信号
- SIGTERM 15：正常结束进程信号
- SIGCONT 18：该信号让暂停的进程恢复执行，本信号不能被阻断
- SIGSTOP 19：该信号暂停前台进程，相当于 <Ctr-z>

**killall** 杀死指定名字的进程。killall 选项 进程名：杀死**进程名**的所有进程，注意不是**进程号**

**pkill** -t 按照终端号踢出进程。

**后台执行** Command &（后台运行命令）

**特点:** 是执行后**命令行不会一直占用**，可以执行其他命令. 例如

```
$firefox &
...

$Available..
```

**ctrl+z** 暂停当前运行进程，并放入后台，（后台暂停命令）

**ctrl+c** 结束当前运行进程

**jobs** 列出所有活动作业的状态信息工作号 工作状态 工作进程

**bg** 设置在后台中运行作业bg %工作号, 如果不想添加工作号，默认执行工作号后面有 [+] 的，接下来是 [-], 如bg \n。

注意：在把后台暂停的工作恢复到后台执行的过程中，后台恢复的执行的命令是不能和前台有交互的，否则不能恢复到后台执行。如top Vi

**fg** 设置在前台中运行作业fg %工作号, 第二种方式如 bg, 直接运行fg 即可。

**lsof** 列出进程打开或使用的文件信息

- -c 字符串：只列出以字符串开头的进程打开的文件
- -u 用户名：列出某个用户的进程打开的文件
- -p pid：列出某个 pid 进程打开的文件

## 2.22 服务管理

- 包安装服务
  - 独立的服务
  - 基于xinetd 服务
- 源码包服务

**查询已安装的服务-包安装与源码安装服务** 查看包安装的服务chkconfig --list, 查看服务的自启动状态，service 只能管理包安装的服务，不能管理源码安装的服务。主要原因是**安装位置的区别**。

查看源码服务安装位置，一般是/usr/local 下

**包安装-独立服务的启动与自启**

- /etc/init.d/ ：启动脚本位置
- /etc/sysconfig/ ：初始化环境配置文件位置
- /etc/ ：配置文件位置
- /etc/xinetd.conf :xinetd 配置文件
- /etc/xinetd.d/ ：基于 xinetd 服务的启动脚本
- /var/lib/ ：服务产生数据放在这里
- /var/log/ ：日志

**启动**

- 标准方式：/etc/init.d/独立服务名 start|stop|status|restart|-> 服务绝对路径 + 操作
- service 独立服务名 start|stop|status|restart|-> 在默认目录中搜索服务名 + 操作

**查询安装服务的状态** service --status -all

**服务自启动**

- 命令方式：chkconfig –level 运行级别服务名 on(off)
- 标准方式：修改/etc/rc.d/rc.local 文件，输入用户民和密码之前，会执行该文件里保存的命令
- 其他：使用ntsysv 命令管理自启动

**包安装- xinetd 服务的启动与自启** 几乎不使用了，被淘汰了，现在只有 telnet 在使用。

**源码安装- 服务的启动与自启**

**启动**

- 使用绝对路径
- 调用**启动脚本**来启动。不同的源码包的启动脚本不同，可以查看源码包安装说明，查看启动脚本的方法

```
usr/local/apache2/bin/apachetcl    start|stop
```

**自启动**

- 修改/etc/rc.d/rc.local,加入脚本启动命令usr/local/apache2/bin/apachetcl  start
- 创建快捷方式至 service 执行时的搜索目录/etc/init.d/,然后执行service 服务名  start 即可

让源码包的服务 可以被chkconfig 与 ntsysv 识别，需要在/etc/init.d/链接文件 中添加如下内容，并执行chkconfig --add 服务链接文件

```
#chkconfig: 35 86 76
#chkconfig: 运行级别 启动顺序 关闭顺序
#description: source package apache
#说明，内容随意
```

## 2.23 系统管理

### 2.23.1 资源管理

**缓冲与缓存** 缓存（cache）用于加速数据从硬盘**读取**的。缓冲（buffer）是用于加速数据**写入**硬盘的

**vmstat** top 精简版-监控系统资源（memory swap  io  system  cpu）:vmstat  刷新延时  刷新次数

**dmesg** 开机时内核检测信息

**free** 查看内存使用状态：`free [-b|-k|-m|-g]`

- `-b`: 以字节显式
- `-k`: 以 KB 显式
- `-m`: 以 MB 显式
- `-g`: 以 GB 显式

**/proc/cpuinfo** 保存 cpu 信息，查看 cpu 可以通过此文件、也可以通过 vmstat, 也可以通过 top。

**uptime** 显式系统的启动时间和平均负载，也就是 top 命令的第一行。w 命令也可以看到这个数据。

**uname** 查看系统与内核相关信息

- `-a` ：查看系统所有相关信息
- `-r` ：查看内核版本
- `-s` ：查看内核名称

**file /bin/外部命令** 判断当前系统位数`file /bin/ls  file /bin/find`

**lsb_release -a** 查看当前 Linux 系统发行版本

### 2.23.2 定时任务

**crond 服务：crontab** `crontab [选项]`

- `-e`: 编辑 crontab 定时任务
- `-l`: 查询 crontab 任务
- `-r`: 删除当前用户所有的 crontab 任务

**任务脚本格式** `* * * * *` 执行的任务

表 2.12: crontab 参数含义

| 项 | 含义 | 范围 |
| --- | --- | --- |
| 第1个* | 一小时当中的第几分钟 | 0-59 |
| 第2个* | 一天当中的第几个小时 | 0-23 |
| 第3个* | 一月当中的第几天 | 1-31 |
| 第4个* | 一年当中的第几月 | 1-12 |
| 第5个* | 一周当中的星期几 | 0-7（0 和 7 都代表星期日） |
| * | 代表任何时间 | 比如第一个 * 就代表一小时的每一分钟都执行一次的意思 |
| , | 代表不联系时间 | 比如第一个 * 替换为‘0,8,14’则表示为每小时的第 0,8,14 分钟执行命令 |
| - | 代表连续的时间范围 | 比如第一个 * 替换为 0-13，则表示为每小时的第 0 分钟到第 13 分钟执行命令 |
| */n | 代表每隔多久执行一次 | 比如第一个替换为 */10，则表示为每隔 10 分钟执行一次命令 |

示例 ->

- 45 22 * * * 命令 ： 在 22 点 45 分执行命令
- 0 17 * * 1 命令 ： 每周 1 的 17 点 0 分执行命令
- 0 5 1,15 * * 命令 ： 每月 1 号和 15 号的 5 点 0 分执行命令
- 40 4 * * 1-5 命令 ： 每周 1 到每周 5，即工作日内每天 4 点 40 分执行命令
- */10 4 * * * 命令 ： 每天 4 点，每隔 10 分钟执行一次命令
- 0 0 1,15 * 1 命令 ： 每月 1 号和 15 号，每周 1 的 0 点 0 分都会执行命令。
- 0 5 1,10,14 * * /root/sh/autobak.sh ：每月 1，10,14 号的 5 点执行脚本 autobak.sh

## 2.24 日志管理

### 2.24.1 常见日志

**/var/log/cron** 记录了系统定时任务相关的日志

**/var/log/cups/** 记录打印信息的日志

**/var/log/dmesg** 记录了系统在开机时内核自检的信息。也可以使用 dmesg 命令直接查看内核自检信息。

**/var/log/btmp** 记录错误登陆的日志。这个文件时 2 进制文件，不能直接使用 vi 查看，需要使用 lastb 命令查看。

**/var/log/lastlog** 记录系统中所有用户最后一次的登录时间的日志。也同样是二进制文件，需要使用 lastlog 命令查看。

### 2.24.2 rsyslogd 日志服务

#### 日志文件格式

- 事件产生的时间
- 发生事件的服务器的主机名
- 产生事件的服务名或程序名
- 事件的具体信息

#### 日志配置文件 `/etc/rsyslog.conf`

配置格式:服务名称[连接符号]日志等级 日志记录位置 -> `authpriv.*  /var/log/secure`

##### 常见服务名称

- auth：安全和认证相关信息

- authpriv：安全和认证相关信息
- cron：系统定时任务 cront 和 at 产生的日志
- daemon：各个守护进程产生的日志
- ftp：ftp 守护进程产生的日志
- kern：内核产生的日志

### 连接符号

- * 代表所有的日志等级
- . 代表只有比后面的级别高的日志都记录下来
- = 代表只记录所需等级的日志
- .! 代表不等于，也就是除了该等级的日志外，其他等级都记录

### 日志等级

- debug : 一般调试信息说明
- info : 基本的通知信息
- notice : 普通信息，但是有一定的重要性
- warning : 警告信息，但是不会影响到服务和系统的运行
- err : 错误信息，达到 err 级别后会影响系统的运行
- crit : 理解状况信息，比 err 还严重
- alert : 比 crit 还严重
- emerg : 比 alert 还严重，系统无法使用了

‘

### 日志记录位置

- 日志文件的绝对路径：`/var/log/secure`
- 系统设备文件：`/dev/lp0`

- 转发给远程主机：@192.168.0.210:514
- 用户名：root
- 忽略或丢弃日志：~

### 2.24.3 日志轮替

日志处理：切割、删除过期日志

#### 日志文件的命名规则

- 【常用】如果日志拥有 dateext 参数，每天都有新的文件，如 secure-20130612.
- 如果日志不拥有 dateext 参数，每天也会有新的文件，那么文件命名会如下：secure.1, 并且当新建文件后，secure.1 会自动更名为 secure.2。

#### 日志记录时间配置 /etc/logrotate.conf

- daily : 日志的轮替周期是每天
- weekly : 日志的轮替是每周
- monthly : 日志的轮替周期是每月
- rotate 数字: 保留的日志文件的个数。0 指没有备份
- compress : 日志轮替时，旧的日志进行压缩
- create mode owner group : 建立新日志，同时指定新日志的权限与所有者和所属组
- size 大小：日志只要大约指定大小才进行日志轮替，而不是按照时间。
- dateext：使用日期作为日志轮替文件的后缀。如 secure-20140123

#### 添加源码服务的日志轮替 vi /etc/logrotate.conf

添加：

```
/usr/local/apache2/logs/access_log{
    daily
    create
    rotate 30
```

```
}
```

**logrotate** logrotate 选项 配置文件名

- 没有选项：按照配置文件中的条件进行日志轮替
- -v：显式日志轮替过程。
- -f：强制进行日志轮替。

## 2.25 启动管理

运行级别

**runlevel** 查看运行级别

**init** 改变运行级别

**/etc/inittab** 定义开机时运行级别

## 2.26 软件包管理

### 2.26.1 已编译文件

linux 的文件安装换句话说就是把相应的执行文件拷贝到系统执行命令的搜索目录中,即$PATH 下，或者在PATH 下创建可执行软件的符号链接

- 将文件拷贝到PATH 下任意目录

```
INSTALL_DIR=/usr/local/bin

all:
-@cd src && make

.PHONY: clean

clean:
```

```
-@cd src && make clean

install:
-cp -f bin/* $(INSTALL_DIR)
```

- ln -s 可执行软件位置 $PATH下任意目录/别名:ln -s ~/.sublime/sublime_text ~/bin/subl

### 2.26.2 源码安装

固定 3 步:

1. ./configure
2. make
3. sudo make install

### 2.26.3 deb 包安装

```
sudo dpkg -i xx.deb
```

**Ubuntu**

```
sudo apt-get install xx
```

## 2.27 参考

http://www.linuxcommand.org/index.php

linux 高级编程:http://guojing.me/linux-kernel-architecture/tags/#do_group_exit

# 第三章　其他

## 3.1　Shell

参考 Scripts/Shell 学习笔记

## 3.2　Linux 系统编程

参考 linux System Coding 笔记

## 3.3　Linux 网络编程

参考 linux Network 笔记