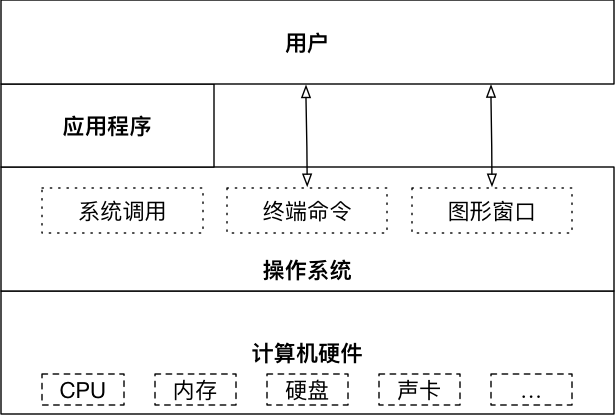
**一、操作系统（科普章节）**

**目标**

* 了解**操作系统**及作用

**1.1. 操作系统（Operation System，OS）**

操作系统作为接口的示意图



没有安装操作系统的计算机，通常被称为 **裸机**

* 如果想在 **裸机** 上运行自己所编写的程序，就必须用机器语言书写程序
* 如果计算机上安装了操作系统，就可以在操作系统上安装支持的高级语言环境，用高级语言开发程序

**1.1.1 操作系统的作用**

* 是现代计算机系统中 最基本和最重要 的系统软件
* 是 配置在计算机硬件上的第一层软件，是对硬件系统的首次扩展
* 主要作用是管理好硬件设备，并为用户和应用程序提供一个简单的接口，以便于使用
* 而其他的诸如编译程序、数据库管理系统，以及大量的应用软件，都直接依赖于操作系统的支持

**1.1.2 不同应用领域的主流操作系统**

* 桌面操作系统
* 服务器操作系统
* 嵌入式操作系统
* 移动设备操作系统

**1> 桌面操作系统**

* Windows 系列
  + 用户群体大
* macOS
  + 适合于开发人员
* Linux
  + 应用软件少

**2> 服务器操作系统**

* Linux
  + 安全、稳定、免费
  + 占有率高
* Windows Server
  + 付费
  + 占有率低

**3> 嵌入式操作系统**

* Linux

**4> 移动设备操作系统**

* iOS
* Android（基于 Linux）

**1.1.3 虚拟机**

虚拟机（Virtual Machine）指通过软件模拟的具有完整硬件系统功能的、运行在一个完全隔离环境中的完整计算机系统

* 虚拟系统通过生成现有操作系统的全新虚拟镜像，具有真实操作系统完全一样的功能
* 进入虚拟系统后，所有操作都是在这个全新的独立的虚拟系统里面进行，可以独立安装运行软件，保存数据，拥有自己的独立桌面，不会对真正的系统产生任何影响
* 而且能够在现有系统与虚拟镜像之间灵活切换的一类操作系统

**二、操作系统的发展史（科普章节）**

**目标**

* 了解操作系统的发展历史
* 知道 Linux 内核及发行版的区别
* 知道 Linux 的应用领域

**2.1. 操作系统的发展历史**

**2.1.1 Unix**

1965 年之前的时候，电脑并不像现在一样普遍，它可不是一般人能碰的起的，除非是军事或者学院的研究机构，而且当时大型主机至多能提供30台终端（30个键盘、显示器)，连接一台电脑

**为了解决数量不够用的问题**

* 1965 年左后由 **贝尔实验室** 加入了 **麻省理工学院** 以及 **通用电气** 合作的计划 —— 该计划要建立一套 **多使用者(multi－user)**、**多任务(multi－processor)**、**多层次(multi－level)** 的 **MULTICS** 操作系统，想让大型主机支持 300 台终端
* 1969 年前后这个项目进度缓慢，资金短缺，贝尔实验室退出了研究
* 1969 年从这个项目中退出的 **Ken Thompson** 当时在实验室无聊时，为了让一台空闲的电脑上能够运行 "星际旅行（Space Travel）" 游行，在 8 月份左右趁着其妻子探亲的时间，**用了 1 个月的时间**，使用汇编写出了 Unix 操作系统的原型
* 1970 年，美国贝尔实验室的 **Ken Thompson**，以 **BCPL** 语言为基础，设计出很简单且很接近硬件的 **B 语言**（取BCPL的首字母），并且他用 **B 语言** 写了第一个 UNIX 操作系统
* 1971 年，同样酷爱 "星际旅行（Space Travel）" 的 **Dennis M.Ritchie** 为了能早点儿玩上游戏，加入了 **Thompson** 的开发项目，合作开发 UNIX，他的主要工作是改造 **B 语言**，因为**B 语言** 的跨平台性较差
* 1972 年，**Dennis M.Ritchie** 在 **B 语言** 的基础上最终设计出了一种新的语言，他取了 **BCPL** 的第二个字母作为这种语言的名字，这就是 **C 语言**
* 1973 年初，**C 语言**的主体完成，**Thompson** 和 **Ritchie** 迫不及待地开始用它完全重写了现在大名鼎鼎的 **Unix 操作系统**

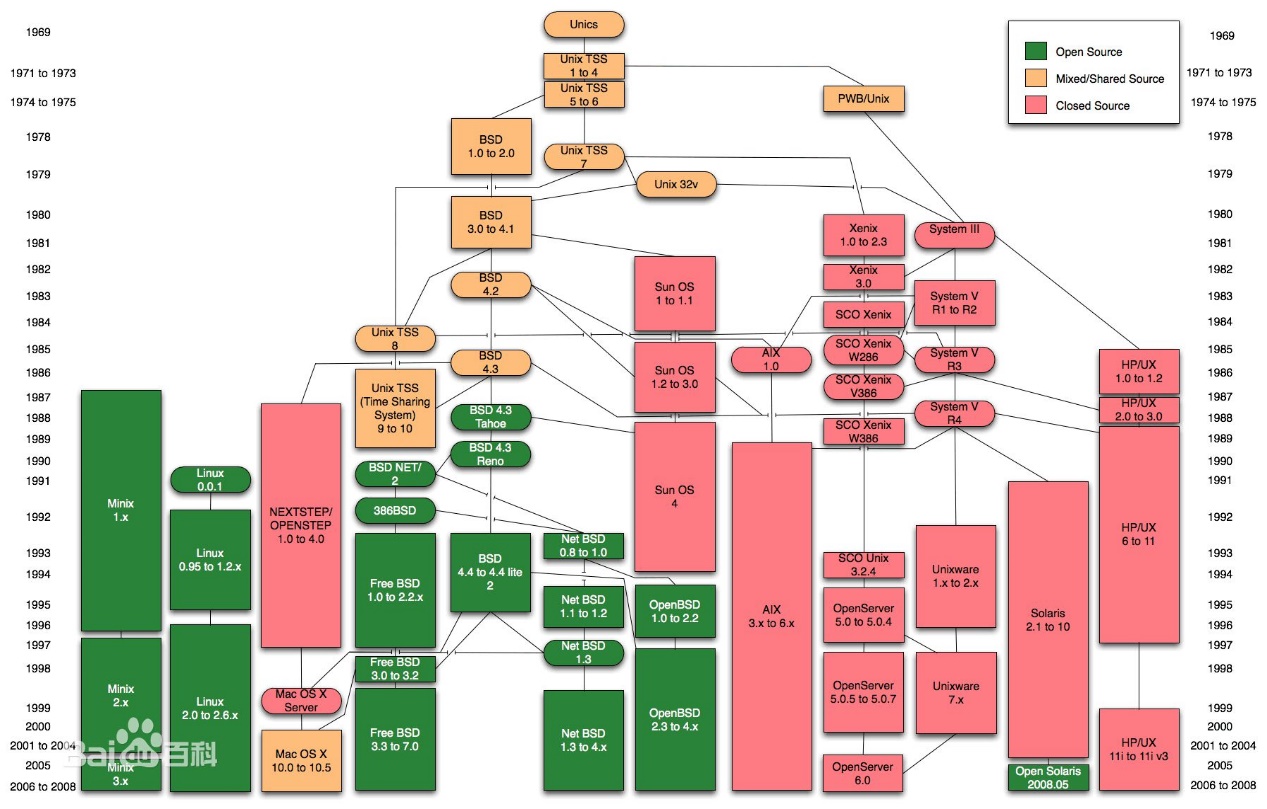


肯·汤普逊（左）和丹尼斯·里奇（右）

**C 语言**

* 在把 **UNIX** 移植到其他类型的计算机上使用时，**C 语言**强大的移植性（Portability）在此显现
* 机器语言和汇编语言都不具有移植性，为 x86 开发的程序，不可能在 Alpha，SPARC 和 ARM 等机器上运行
* 而 **C 语言**程序则可以使用在任意架构的处理器上，只要那种架构的处理器**具有对应的 C 语言编译器和库**，然后将 C 源代码编译、连接成目标二进制文件之后即可运行

**Unix 家谱**

****

### 2.1.2 Minix

* 因为 **AT&T（通用电气）** 的政策改变，在 Version 7 Unix 推出之后，发布新的使用条款，将 UNIX 源代码私有化，在大学中不再能使用 UNIX 源代码
* **Andrew S. Tanenbaum（塔能鲍姆）** 教授为了能 **在课堂上教授学生操作系统运作的细节**，决定在不使用任何 AT&T 的源代码前提下，自行开发与 UNIX 兼容的操作系统，以避免版权上的争议
* 以 **小型 UNIX（mini-UNIX）**之意，将它称为 **MINIX**

### 2.1.3 Linux

* 1991 年 **林纳斯（Linus）** 就读于赫尔辛基大学期间，对 Unix 产生浓厚兴趣，尝试着在Minix 上做一些开发工作
* 因为 **Minix** 只是教学使用，因此功能并不强，**林纳斯** 经常要用他的终端 **仿真器（Terminal Emulator）** 去访问大学主机上的新闻组和邮件，为了方便读写和下载文件，他自己编写了磁盘驱动程序和文件系统，这些在后来成为了 Linux 第一个内核的雏形，当时，他年仅 21 岁！
* **林纳斯** 利用 GNU 的 bash 当做开发环境，gcc 当做编译工具，编写了 Linux 内核，一开始 Linux 并不能兼容 Unix
* 即 Unix 上跑的应用程序不能在 Linux 上跑，即应用程序与内核之间的接口不一致
* 一开始 Linux 只适用于 386，后来经过全世界的网友的帮助，最终能够兼容多种硬件



**2.2. Linux 内核及发行版**

### 2.2.1 Linux 内核版本

* **内核（kernel）**是系统的心脏，是运行程序和管理像磁盘和打印机等硬件设备的核心程序，它提供了一个在裸设备与应用程序间的抽象层
* Linux 内核版本又分为 **稳定版** 和 **开发版**，两种版本是相互关联，相互循环
* **稳定版**：具有工业级强度，可以广泛地应用和部署。新的稳定版相对于较旧的只是修正一些 bug 或加入一些新的驱动程序
* **开发版**：由于要试验各种解决方案，所以变化很快
* 内核源码网址：<http://www.kernel.org>

*所有来自全世界的对 Linux 源码的修改最终都会汇总到这个网站，由 Linus 领导的开源社区对其进行甄别和修改最终决定是否进入到 Linux 主线内核源码中*

### 2.2.2 Linux 发行版本

* **Linux 发行版（也被叫做 GNU/Linux 发行版）**通常包含了包括桌面环境、办公套件、媒体播放器、数据库等应用软件
* 常见的发行版本如下：
  + Ubuntu
  + Redhat
  + Fedora
  + openSUSE
  + Linux Mint
  + Debian
  + Manjaro
  + Mageia
  + CentOS
  + Arch
* 十大 Linux 服务器发行版排行榜：<http://os.51cto.com/art/201612/526126.htm>

*在几乎每一份与 Linux 有关的榜单上，基于 Debian 的 Ubuntu 都占有一席之位。Canonical 的Ubuntu 胜过其他所有的 Linux 服务器发行版 ―― 从简单安装、出色的硬件发现，到世界级的商业支持，Ubuntu确立了难以企及的严格标准*

**2.3. Linux 的应用领域**

### 2.3.1 服务器领域

* Linux 在服务器领域的应用是其重要分支
* Linux 免费、稳定、高效等特点在这里得到了很好的体现
  + 早期因为维护、运行等原因同样受到了很大的限制
  + 近些年来 Linux 服务器市场得到了飞速的提升，尤其在一些高端领域尤为广泛

### 2.3.2 嵌入式领域

* 近些年来 Linux 在嵌入式领域的应用得到了飞速的提高
* Linux 运行稳定、对网络的良好支持性、低成本，且可以根据需要进行软件裁剪，内核最小可以达到几百 KB 等特点，使其近些年来在嵌入式领域的应用得到非常大的提高

*主要应用：****机顶盒****、****数字电视****、****网络电话****、****程控交换机****、****手机****、****PDA****、等都是其应用领域，得到了 Google、三星、摩托罗拉、NEC 等公司的大力推广*

### 2.3.3 个人桌面领域

* 此领域是传统 Linux 应用最薄弱的环节
* 传统 Linux 由于界面简单、操作复杂、应用软件少的缺点，一直被 Windows 所压制
* 近些年来随着 **Ubuntu**、**Fedora** 等优秀桌面环境的兴起，同时各大硬件厂商对其支持的加大，Linux 在个人桌面领域的占有率在逐渐的提高

**三、文件和目录（理解）**

**目标**

* 理解 Linux 文件目录的结构

**3.1. 单用户操作系统和多用户操作系统（科普）**

* **单用户操作系统**：指一台计算机在同一时间 **只能由一个用户** 使用，一个用户独自享用系统的全部硬件和软件资源
  + **Windows XP** 之前的版本都是单用户操作系统
* **多用户操作系统**：指一台计算机在同一时间可以由 **多个用户** 使用，多个用户共同享用系统的全部硬件和软件资源
  + **Unix** 和 **Linux** 的设计初衷就是多用户操作系统

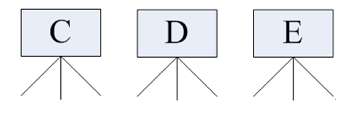
**3.2. Windows 和 Linux 文件系统区别**

### 3.2.1 Windows 下的文件系统

* 在 Windows 下，打开 “计算机”，我们看到的是一个个的驱动器盘符：

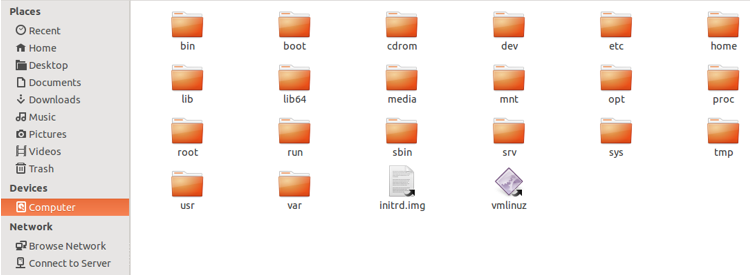


* 每个驱动器都有自己的根目录结构，这样形成了多个树并列的情形，如图所示：

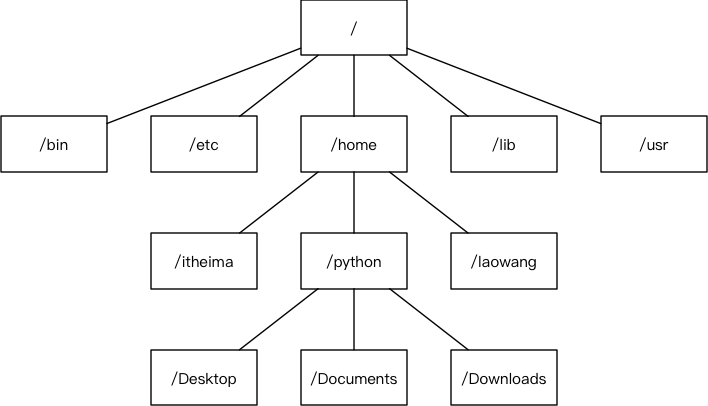


### 3.2.2 Linux 下的文件系统

* 在 Linux 下，我们是看不到这些驱动器盘符，我们看到的是文件夹（目录）：



* Ubuntu 没有盘符这个概念，只有一个根目录 /，所有文件都在它下面



### 3.2.3 用户目录

* **位于 /home/user，称之为用户工作目录或家目录**，表示方式：

/home/user

~

**3.3 Linux 主要目录速查表**

* /：根目录，**一般根目录下只存放目录**，在 linux 下有且只有一个根目录，所有的东西都是从这里开始
  + 当在终端里输入 /home，其实是在告诉电脑，先从 /（根目录）开始，再进入到 home 目录
* /bin、/usr/bin：可执行二进制文件的目录，如常用的命令 ls、tar、mv、cat 等
* /boot：放置 linux 系统启动时用到的一些文件，如 linux 的内核文件：/boot/vmlinuz，系统引导管理器：/boot/grub
* /dev：存放linux系统下的设备文件，访问该目录下某个文件，相当于访问某个设备，常用的是挂载光驱mount /dev/cdrom /mnt
* /etc：系统配置文件存放的目录，不建议在此目录下存放可执行文件，重要的配置文件有
  + /etc/inittab
  + /etc/fstab
  + /etc/init.d
  + /etc/X11
  + /etc/sysconfig
  + /etc/xinetd.d
* /home：系统默认的用户家目录，新增用户账号时，用户的家目录都存放在此目录下
  + ~ 表示当前用户的家目录
  + ~edu 表示用户 edu 的家目录
* /lib、/usr/lib、/usr/local/lib：系统使用的函数库的目录，程序在执行过程中，需要调用一些额外的参数时需要函数库的协助
* /lost+fount：系统异常产生错误时，会将一些遗失的片段放置于此目录下
* /mnt: /media：光盘默认挂载点，通常光盘挂载于 /mnt/cdrom 下，也不一定，可以选择任意位置进行挂载
* /opt：给主机额外安装软件所摆放的目录
* /proc：此目录的数据都在内存中，如系统核心，外部设备，网络状态，由于数据都存放于内存中，所以不占用磁盘空间，比较重要的文件有：/proc/cpuinfo、/proc/interrupts、/proc/dma、/proc/ioports、/proc/net/\* 等
* /root：系统管理员root的家目录
* /sbin、/usr/sbin、/usr/local/sbin：放置系统管理员使用的可执行命令，如 fdisk、shutdown、mount 等。与 /bin 不同的是，这几个目录是给系统管理员 root 使用的命令，一般用户只能"查看"而不能设置和使用
* /tmp：一般用户或正在执行的程序临时存放文件的目录，任何人都可以访问，重要数据不可放置在此目录下
* /srv：服务启动之后需要访问的数据目录，如 www 服务需要访问的网页数据存放在 /srv/www 内
* /usr：应用程序存放目录
  + /usr/bin：存放应用程序
  + /usr/share：存放共享数据
  + /usr/lib：存放不能直接运行的，却是许多程序运行所必需的一些函数库文件
  + /usr/local：存放软件升级包
  + /usr/share/doc：系统说明文件存放目录
  + /usr/share/man：程序说明文件存放目录
* /var：放置系统执行过程中经常变化的文件
  + /var/log：随时更改的日志文件
  + /var/spool/mail：邮件存放的目录
  + /var/run：程序或服务启动后，其 PID 存放在该目录下

**四、常用 Linux 命令的基本使用**

**目标**

* 理解学习 Linux 终端命令的原因
* 常用 Linux 命令体验

**4.1. 学习 Linux 终端命令的原因**

* Linux 刚面世时并没有图形界面，所有的操作全靠命令完成，如 **磁盘操作**、**文件存取**、**目录操作**、**进程管理**、**文件权限** 设定等
* 在职场中，大量的 **服务器维护工作** 都是在 **远程** 通过 **SSH 客户端** 来完成的，并没有图形界面，所有的维护工作都需要通过命令来完成
* 在职场中，作为后端程序员，必须要或多或少的掌握一些 Linux 常用的终端命令
* Linux 发行版本的命令大概有 200 多个，但是常用的命令只有 10 多个而已

*学习终端命令的技巧：*

* *不需要死记硬背，对于常用命令，用的多了，自然就记住了*
* *不要尝试一次学会所有的命令，有些命令是非常不常用的，临时遇到，临时百度就可以*

**4.2. 常用 Linux 命令的基本使用**

| **序号** | **命令** | **对应英文** | **作用** |
| --- | --- | --- | --- |
| 01 | ls | list | 查看当前文件夹下的内容 |
| 02 | pwd | print wrok directory | 查看当前所在文件夹 |
| 03 | cd [目录名] | change directory | 切换文件夹 |
| 04 | touch [文件名] | touch | 如果文件不存在，新建文件 |
| 05 | mkdir [目录名] | make directory | 创建目录 |
| 06 | rm [文件名] | remove | 删除指定的文件名 |
| 07 | clear | clear | 清屏 |

*小技巧*

* ctrl + shift + = **放大**终端窗口的字体显示
* ctrl + - **缩小**终端窗口的字体显示

**4.3 终端实用技巧**

### 4.3.1 自动补全

* 在敲出 文件／目录／命令 的前几个字母之后，按下 tab 键
  + 如果输入的没有歧义，系统会自动补全
  + 如果还存在其他 文件／目录／命令，再按一下 tab 键，系统会提示可能存在的命令

### 4.3.2 曾经使用过的命令

* 按 上／下 光标键可以在曾经使用过的命令之间来回切换
* 如果想要退出选择，并且不想执行当前选中的命令，可以按 ctrl + c
* clear清屏

**五、文件和目录常用命令**

**目标**

* 查看目录内容
  + ls
* 切换目录
  + cd
* 创建和删除操作
  + touch
  + rm
  + mkdir
* 拷贝和移动文件
  + cp
  + mv
* 查看文件内容
  + cat
  + more
  + grep
* 其他
  + echo
  + 重定向 > 和 >>
  + 管道 |

**5.1. 查看目录内容**

### 5.1.1 ls 命令说明

* ls 是英文单词 **list** 的简写，其功能为列出目录的内容，是用户最常用的命令之一，类似于 **DOS** 下的 dir 命令

**Linux 下文件和目录的特点**

* Linux **文件** 或者 **目录** 名称最长可以有 256 个字符
* 以 . 开头的文件为隐藏文件，需要用 -a 参数才能显示
* **. 代表当前目录**
* **.. 代表上一级目录**

### 5.1.2 ls 常用选项

| **参数** | **含义** |
| --- | --- |
| -a | 显示指定目录下所有子目录与文件，包括隐藏文件 |
| -l | 以列表方式显示文件的详细信息 |
| -h | 配合 -l 以人性化的方式显示文件大小 |

#### 计算机中文件大小的表示方式（科普）

| **单位** | **英文** | **含义** |
| --- | --- | --- |
| 字节 | B（Byte） | 在计算机中作为一个数字单元，一般为 8 位二进制数 |
| 千 | K（Kibibyte） | 1 KB = 1024 B，千字节 （1024 = 2 \*\* 10） |
| 兆 | M（Mebibyte） | 1 MB = 1024 KB，百万字节 |
| 千兆 | G（Gigabyte） | 1 GB = 1024 MB，十亿字节，千兆字节 |
| 太 | T（Terabyte） | 1 TB = 1024 GB，万亿字节，太字节 |
| 拍 | P（Petabyte） | 1 PB = 1024 TB，千万亿字节，拍字节 |
| 艾 | E（Exabyte） | 1 EB = 1024 PB，百亿亿字节，艾字节 |
| 泽 | Z（Zettabyte） | 1 ZB = 1024 EB，十万亿亿字节，泽字节 |
| 尧 | Y（Yottabyte） | 1 YB = 1024 ZB，一亿亿亿字节，尧字节 |

### 5.1.3 ls 通配符的使用

| **通配符** | **含义** |
| --- | --- |
| \* | 代表任意个数个字符 |
| ? | 代表任意一个字符，至少 1 个 |
| [] | 表示可以匹配字符组中的任一一个 |
| [abc] | 匹配 a、b、c 中的任意一个 |
| [a-f] | 匹配从 a 到 f 范围内的的任意一个字符 |

**5.2. 切换目录**

### 5.2.1 cd

* cd 是英文单词 **change directory** 的简写，其功能为更改当前的工作目录，也是用户最常用的命令之一

*注意：Linux 所有的****目录****和****文件名****都是大小写敏感的*

| **命令** | **含义** |
| --- | --- |
| cd | 切换到当前用户的主目录(/home/用户目录) |
| cd ~ | 切换到当前用户的主目录(/home/用户目录) |
| cd . | 保持在当前目录不变 |
| cd .. | 切换到上级目录 |
| cd - | 可以在最近两次工作目录之间来回切换 |

### 5.2.2 相对路径和绝对路径

* **相对路径** 在输入路径时，最前面不是 **/** 或者 **~**，表示相对 **当前目录** 所在的目录位置
* **绝对路径** 在输入路径时，最前面是 **/** 或者 **~**，表示从 **根目录/家目录** 开始的具体目录位置

**5.3. 创建和删除操作**

### 5.3.1 touch

* 创建文件或修改文件时间
  + 如果文件 **不存在**，可以创建一个空白文件
  + 如果文件 **已经存在**，可以修改文件的末次修改日期

### 5.3.2 mkdir

* 创建一个新的目录

| **选项** | **含义** |
| --- | --- |
| -p | 可以递归创建目录 |

***新建目录的名称****不能与当前目录中****已有的目录或文件****同名*

### 5.3.3 rm

* 删除文件或目录

*使用 rm 命令要小心，因为文件删除后不能恢复*

| **选项** | **含义** |
| --- | --- |
| -f | 强制删除，忽略不存在的文件，无需提示 |
| -r | 递归地删除目录下的内容，**删除文件夹** 时必须加此参数 |

**5.4. 拷贝和移动文件**

| **序号** | **命令** | **对应英文** | **作用** |
| --- | --- | --- | --- |
| 01 | tree [目录名] | tree | 以树状图列出文件目录结构 |
| 02 | cp 源文件 目标文件 | copy | 复制文件或者目录 |
| 03 | mv 源文件 目标文件 | move | 移动文件或者目录／文件或者目录重命名 |

### 5.4.1 tree

* tree 命令可以以树状图列出文件目录结构

| **选项** | **含义** |
| --- | --- |
| -d | 只显示目录 |

### 5.4.2 cp

* cp 命令的功能是将给出的 **文件** 或 **目录** 复制到另一个 **文件** 或 **目录** 中，相当于 **DOS** 下的 copy 命令

| **选项** | **含义** |
| --- | --- |
| -i | 覆盖文件前提示 |
| -r | 若给出的源文件是目录文件，则 cp 将递归复制该目录下的所有子目录和文件，目标文件必须为一个目录名 |

### 5.4.3 mv

* mv 命令可以用来 **移动** **文件** 或 **目录**，也可以给 **文件或目录重命名**

| **选项** | **含义** |
| --- | --- |
| -i | 覆盖文件前提示 |

***备注：对于创建/删除/复制/剪切，创建对文件(touch)和文件夹(mkdir)是不同命令，删除(rm)/复制(cp)对文件与文件夹区分是加 –r，剪切(mv)对文件与文件夹不区分。***

**5.5. 查看文件内容**

| **序号** | **命令** | **对应英文** | **作用** |
| --- | --- | --- | --- |
| 01 | cat 文件名 | concatenate | 查看文件内容、创建文件、文件合并、追加文件内容等功能 |
| 02 | more 文件名 | more | 分屏显示文件内容 |
| 03 | grep 搜索文本 文件名 | grep | 搜索文本文件内容 |

### 5.5.1 cat

* cat 命令可以用来 **查看文件内容**、**创建文件**、**文件合并**、**追加文件内容** 等功能
* cat 会一次显示所有的内容，适合 **查看内容较少** 的文本文件

| **选项** | **含义** |
| --- | --- |
| -b | 对非空输出行编号 |
| -n | 对输出的所有行编号 |

*Linux 中还有一个 nl 的命令和 cat -b 的效果等价*

### 5.5.2 more

* more 命令可以用于分屏显示文件内容，每次只显示一页内容
* 适合于 **查看内容较多**的文本文件

使用 more 的操作键：

| **操作键** | **功能** |
| --- | --- |
| 空格键 | 显示手册页的下一屏 |
| Enter 键 | 一次滚动手册页的一行 |
| b | 回滚一屏 |
| f | 前滚一屏 |
| q | 退出 |
| /word | 搜索 **word** 字符串 |

### 5.5.3 grep

* Linux 系统中 grep 命令是一种强大的文本搜索工具
* grep允许对文本文件进行 **模式**查找，所谓模式查找，又被称为正则表达式，在就业班会详细讲解

| **选项** | **含义** |
| --- | --- |
| -n | 显示匹配行及行号 |
| -v | 显示不包含匹配文本的所有行（相当于求反） |
| -i | 忽略大小写 |

* 常用的两种模式查找

| **参数** | **含义** |
| --- | --- |
| ^a | 行首，搜寻以 **a** 开头的行 |
| ke$ | 行尾，搜寻以 **ke** 结束的行 |

**5.6. 其他**

### 5.6.1 echo 文字内容

* echo 会在终端中显示参数指定的文字，通常会和 **重定向** 联合使用

### 5.6.2 重定向 > 和 >>

* Linux 允许将命令执行结果 **重定向**到一个 **文件**
* 将本应显示在**终端上的内容** **输出／追加** 到**指定文件中**

其中

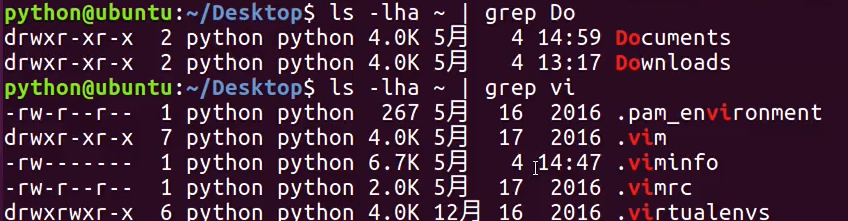
* > 表示输出，会覆盖文件原有的内容
* >> 表示追加，会将内容追加到已有文件的末尾

### 5.6.3 管道 |

* Linux 允许将 **一个命令的输出** 可以**通过管道** 做为 **另一个命令的输入**
* 可以理解现实生活中的管子，管子的一头塞东西进去，另一头取出来，这里 | 的左右分为两端，左端塞东西（写），右端取东西（读）

常用的管道命令有：

* more：分屏显示内容
* grep：在命令执行结果的基础上查询指定的文本



**六、远程管理常用命令**

**目标**

* 关机/重启
  + shutdown
* 查看或配置网卡信息
  + ifconfig
  + ping
* 远程登录和复制文件
  + ssh
  + scp

**6.1. 关机/重启**

| **序号** | **命令** | **对应英文** | **作用** |
| --- | --- | --- | --- |
| 01 | shutdown 选项 时间 | shutdown | 关机／重新启动 |

### 6.1.1 shutdown

* shutdown 命令可以 **安全** **关闭** 或者 **重新启动系统**

| **选项** | **含义** |
| --- | --- |
| -r | 重新启动 |

*提示：*

* ***不指定选项和参数****，默认表示****1 分钟****之后****关闭电脑***
* *远程维护服务器时，最好不要关闭系统，而应该重新启动系统*
* 常用命令示例

# 重新启动操作系统，其中 now 表示现在

$ shutdown -r now

# 立刻关机，其中 now 表示现在

$ shutdown now

# 系统在今天的 20:25 会关机

$ shutdown 20:25

# 系统再过十分钟后自动关机

$ shutdown +10

# 取消之前指定的关机计划

$ shutdown -c

**6.2. 查看或配置网卡信息**

| **序号** | **命令** | **对应英文** | **作用** |
| --- | --- | --- | --- |
| 01 | ifconfig | configure a network interface | 查看/配置计算机当前的网卡配置信息 |
| 02 | ping ip地址 | ping | 检测到目标 ip地址 的连接是否正常 |

### 6.2.1 网卡 和 IP 地址

#### 网卡

* 网卡是一个专门负责网络通讯的硬件设备
* **IP 地址**是设置在网卡上的地址信息

我们可以把 **电脑** 比作 **电话**，**网卡** 相当于 **SIM 卡**，**IP 地址** 相当于 **电话号码**

#### IP 地址

* **每台联网的电脑上**都有 **IP 地址**，**是保证电脑之间正常通讯的重要设置**

注意：每台电脑的 IP 地址不能相同，否则会出现 IP 地址冲突，并且没有办法正常通讯

提示：有关 **IP 地址**的详细内容，在就业班会详细讲解！

### 6.2.2 ifconfig

* ifconfig 可以查看／配置计算机当前的网卡配置信息

# 查看网卡配置信息

$ ifconfig

# 查看网卡对应的 IP 地址

$ ifconfig | grep inet

*提示：一台计算机中有可能会有一个****物理网卡****和****多个虚拟网卡****，在 Linux 中物理网卡的名字通常以 ensXX 表示*

* 127.0.0.1 被称为 **本地回环/环回地址**，一般用来测试本机网卡是否正常

### 6.2.3 ping

# 检测到目标主机是否连接正常

$ ping IP地址

# 检测本地网卡工作正常

$ ping 127.0.0.1

* ping 一般用于检测当前计算机到目标计算机之间的网络 **是否通畅**，**数值越大，速度越慢**
* *ping 的工作原理与潜水艇的声纳相似，ping 这个命令就是取自****声纳的声音***
* *网络管理员之间也常将 ping 用作动词 ——****ping 一下计算机X，看他是否开着***

原理：网络上的机器都有 **唯一确定的 IP 地址**，我们给**目标 IP 地址**发送一个数据包，对方就要返回一个数据包，根据返回的数据包以及时间，我们可以确定目标主机的存在

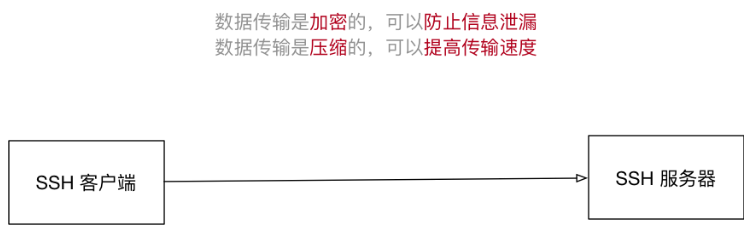
*提示：在 Linux 中，想要终止一个终端程序的执行，绝大多数都可以使用 CTRL + C*

**6.3. 远程登录和复制文件**

| **序号** | **命令** | **对应英文** | **作用** |
| --- | --- | --- | --- |
| 01 | ssh 用户名@ip | secure shell | 关机／重新启动 |
| 02 | scp 用户名@ip:文件名或路径 用户名@ip:文件名或路径 | secure copy | 远程复制文件 |

### 6.3.1 ssh 基础（重点）

在 Linux 中 SSH 是 **非常常用** 的工具，通过 **SSH 客户端** 我们可以连接到运行了 **SSH 服务器** 的远程机器上



* **SSH 客户端**是一种使用 Secure Shell（SSH） 协议连接到远程计算机的软件程序
* SSH 是目前较可靠，**专为远程登录会话和其他网络服务** 提供安全性的协议
  + 利用 SSH 协议 可以有效**防止远程管理过程中的信息泄露**
  + 通过 SSH 协议 可以对所有传输的数据进行加密，也能够防止 DNS 欺骗和 IP 欺骗
* SSH 的另一项优点是传输的数据可以是经过压缩的，所以可以加快传输的速度

#### 1) 域名 和 端口号

域名

* 由一串 **用点分隔** 的名字组成，例如：www.itcast.cn
* 是 **IP 地址** 的别名，方便用户记忆

端口号

* **IP 地址**：通过 **IP 地址** 找到网络上的 **计算机**
* **端口号**：通过 **端口号** 可以找到 **计算机上运行的应用程序**
  + **SSH 服务器** 的默认端口号是 22，如果是默认端口号，在连接的时候，可以省略
* 常见服务端口号列表：

| **序号** | **服务** | **端口号** |
| --- | --- | --- |
| 01 | SSH 服务器 | 22 |
| 02 | Web 服务器 | 80 |
| 03 | HTTPS | 443 |
| 04 | FTP 服务器 | 21 |

*提示：有关****端口号****的详细内容，在就业班会详细讲解！*

#### 2) SSH 客户端的简单使用

ssh [-p port] user@remote

* user 是在远程机器上的用户名，如果不指定的话默认为当前用户
* remote 是远程机器的地址，可以是 **IP**／**域名**，或者是 **后面会提到的别名**
* port 是 **SSH Server 监听的端口**，如果不指定，就为默认值 22

*提示：*

* *使用 exit 退出当前用户的登录*

*注意：*

* *ssh 这个终端命令只能在 Linux 或者 UNIX 系统下使用*
* *如果在 Windows 系统中，可以安装 PuTTY 或者 XShell 客户端软件即可*

*提示：*

* *在工作中，SSH 服务器的端口号很有可能不是 22，如果遇到这种情况就需要使用 -p 选项，指定正确的端口号，否则无法正常连接到服务器*

#### 3) Windows 下 SSH 客户端的安装

* Putty [http://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html](http://www.chiark.greenend.org.uk/%7Esgtatham/putty/latest.html)
* XShell <http://xshellcn.com>

*建议从官方网站下载正式的安装程序*

### 6.3.2 scp（掌握）

* scp 就是 secure copy，是一个在 Linux 下用来进行 **远程拷贝文件** 的命令
* 它的**地址格式与 ssh 基本相同**，**需要注意的是**，在指定端口时用的是大写的 -P 而不是小写的



# 把本地当前目录下的 01.py 文件 复制到 远程 家目录下的 Desktop/01.py

# 注意：`:` 后面的路径如果不是绝对路径，则以用户的家目录作为参照路径

scp -P port 01.py user@remote:Desktop/01.py

# 把远程 家目录下的 Desktop/01.py 文件 复制到 本地当前目录下的 01.py

scp -P port user@remote:Desktop/01.py 01.py

# 加上 -r 选项可以传送文件夹

# 把当前目录下的 demo 文件夹 复制到 远程 家目录下的 Desktop

scp -r demo user@remote:Desktop

# 把远程 家目录下的 Desktop 复制到 当前目录下的 demo 文件夹

scp -r user@remote:Desktop demo

| **选项** | **含义** |
| --- | --- |
| -r | 若给出的源文件是目录文件，则 scp 将递归复制该目录下的所有子目录和文件，目标文件必须为一个目录名 |
| -P | 若远程 SSH 服务器的端口不是 22，需要使用大写字母 -P 选项指定端口 |

*注意：*

* *scp 这个终端命令只能在 Linux 或者 UNIX 系统下使用*
* *如果在 Windows 系统中，可以安装 PuTTY，使用 pscp 命令行工具或者安装 FileZilla 使用 FTP 进行文件传输*

#### FileZilla

* 官方网站：<https://www.filezilla.cn/download/client>
* FileZilla 在传输文件时，使用的是 FTP 服务 而不是 SSH 服务，因此端口号应该设置为 21

### 6.3.3 SSH 高级（知道）

* 免密码登录
* 配置别名

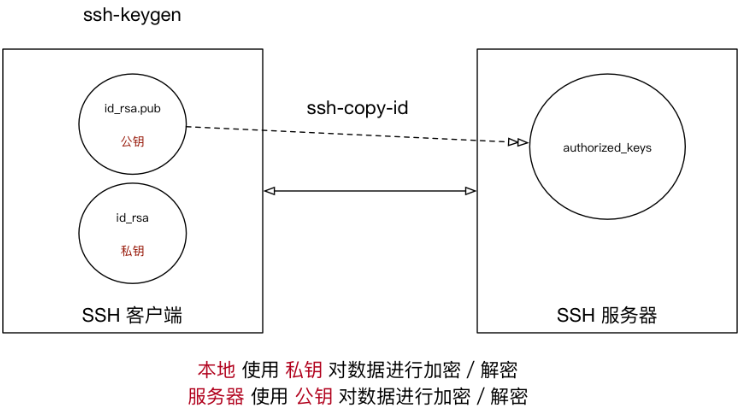
*提示：有关 SSH 配置信息都保存在用户家目录下的 .ssh 目录下*

#### 1）免密码登录

##### 步骤

* 配置公钥
  + 执行 ssh-keygen 即可生成 SSH 钥匙，一路回车即可
* 上传公钥到服务器
  + 执行 ssh-copy-id -p port user@remote，可以让远程服务器记住我们的公钥

##### 示意图



*非对称加密算法*

* *使用****公钥****加密的数据，需要使用****私钥****解密*
* *使用****私钥****加密的数据，需要使用****公钥****解密*

#### 2) 配置别名

每次都输入 ssh -p port user@remote，时间久了会觉得很麻烦，特别是当 user, remote 和 port 都得输入，而且还不好记忆

而 **配置别名** 可以让我们进一步偷懒，譬如用：ssh mac 来替代上面这么一长串，那么就在 ~/.ssh/config 里面追加以下内容：

Host mac

HostName ip地址

User itheima

Port 22

**保存之后，即可用 ssh mac 实现远程登录了，scp 同样可以使用**

**七、用户权限相关命令**

**目标**

* **用户** 和 **权限** 的基本概念
* **用户管理** 终端命令
* **组管理** 终端命令
* **修改权限** 终端命令

**7.1.**用户**和**权限**的基本概念**

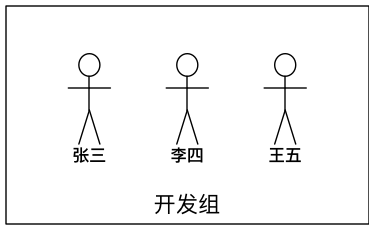
### 7.1.1 基本概念

* **用户** 是 Linux 系统工作中重要的一环，用户管理包括 **用户** 与 **组** 管理
* 在 Linux 系统中，不论是由本机或是远程登录系统，每个系统都**必须拥有一个账号**，并且**对于不同的系统资源拥有不同的使用权限**
* 在 Linux 中，可以指定 **每一个用户** 针对 **不同的文件或者目录** 的 **不同权限**
* 对 **文件／目录** 的权限包括：

| **序号** | **权限** | **英文** | **缩写** | **数字代号** |
| --- | --- | --- | --- | --- |
| 01 | 读 | read | r | 4 |
| 02 | 写 | write | w | 2 |
| 03 | 执行 | excute | x | 1 |

### 7.1.2 组

* 为了方便用户管理，提出了 **组** 的概念，如下图所示



* 在实际应用中，可以预先针对 **组** 设置好权限，然后 **将不同的用户添加到对应的组中**，从而**不用依次为每一个用户设置权限**
* 对相同权限的用户可以放在一个组

### 7.1.3 ls -l 扩展

* ls -l 可以查看文件夹下文件的详细信息，从左到右依次是：
  + **权限**，第 1 个字符如果是 d 表示目录
  + **硬链接数**，通俗地讲，就是有多少种方式，可以访问到当前目录／文件
  + **拥有者**，家目录下 文件／目录 的拥有者通常都是当前用户
  + **组**，在 Linux 中，很多时候，会出现组名和用户名相同的情况，后续会讲
  + **大小**
  + **时间**
  + **名称**



### 7.1.4 chmod使用（重要）

* chmod 可以修改 **用户／组** 对 **文件／目录** 的权限
* 命令格式如下：

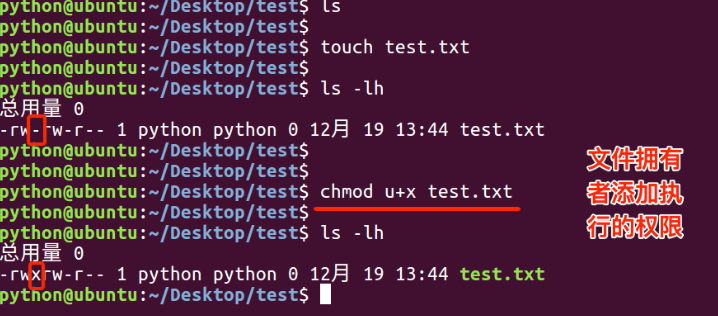
**chmod +/-rwx 文件名|目录名**

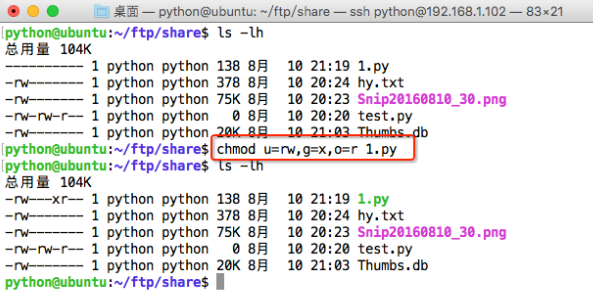
*提示：以上方式会一次性修改 拥有者 / 组 权限，有关 chmod 的高级用法，后续会讲*

**字母法：chmod u/g/o/a +/-/= rwx 文件**

| **[ u/g/o/a ]** | **含义** |
| --- | --- |
| u | user 表示该文件的所有者 |
| g | group 表示与该文件的所有者属于同一组( group )者，即用户组 |
| o | other 表示其他以外的人 |
| a | all 表示这三者皆是 |

| **[ +-= ]** | **含义** |
| --- | --- |
| + | 增加权限 |
| - | 撤销权限 |
| = | 设定权限 |





**行：chmod u=rwx,g=rx,o=r filename 就等同于：chmod u=7,g=5,o=4 filename**

### 7.1.5 超级用户

* Linux 系统中的 root 账号通常 **用于系统的维护和管理**，对操作系统的所有资源 **具有所有访问权限**
* 在大多数版本的 Linux 中，都不推荐 **直接使用 root 账号登录系统**
* 在 Linux 安装的过程中，系统会自动创建一个用户账号，而这个默认的用户就称为“标准用户”

#### sudo

* su 是 substitute user 的缩写，表示 **使用另一个用户的身份**
* sudo 命令用来以其他身份来执行命令，预设的身份为 root
* 用户使用 sudo 时，必须先输入密码，之后有 **5 分钟的有效期限**，超过期限则必须重新输入密码

*若其未经授权的用户企图使用 sudo，则会发出警告邮件给管理员*

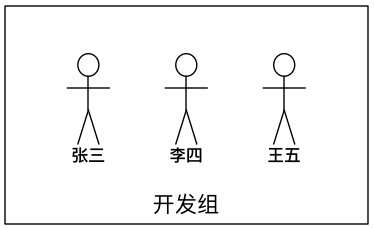
**7.2.**组管理**终端命令**

*提示：****创建组****/****删除组****的终端命令都需要通过 sudo 执行*

| **序号** | **命令** | **作用** |
| --- | --- | --- |
| 01 | groupadd 组名 | 添加组 |
| 02 | groupdel 组名 | 删除组 |
| 03 | cat /etc/group | 确认组信息 |
| 04 | chgrp -R 组名 文件/目录名 | 递归修改文件/目录的所属组 |

*提示：*

* *组信息保存在 /etc/group 文件中*
* */etc 目录是专门用来保存****系统配置信息****的目录*



* 在实际应用中，可以预先针对 **组** 设置好权限，然后 **将不同的用户添加到对应的组中**，从而**不用依次为每一个用户设置权限**

#### 演练目标

1. 在 python 用户的桌面文件夹下创建 Python学习 目录
2. 新建 dev 组
3. 将 Python学习 目录的组修改为 dev

**7.3.**用户管理**终端命令**

*提示：****创建用户****/****删除用户****/****修改其他用户密码****的终端命令都需要通过 sudo 执行*

### 7.3.1 创建用户／设置密码／删除用户

| **序号** | **命令** | **作用** | **说明** |
| --- | --- | --- | --- |
| 01 | useradd -m -g 组 新建用户名 | 添加新用户 | * -m 自动建立用户家目录 * -g 指定用户所在的组，否则会建立一个和同名的组 |
| 02 | passwd 用户名 | 设置用户密码 | 如果是普通用户，直接用 passwd 可以修改自己的账户密码 |
| 03 | userdel -r 用户名 | 删除用户 | -r 选项会自动删除用户家目录 |
| 04 | cat /etc/passwd | grep 用户名 | 确认用户信息 | 新建用户后，用户信息会保存在 /etc/passwd 文件中 |

*提示：*

* *创建用户时，如果忘记添加 -m 选项指定新用户的家目录 —— 最简单的方法就是****删除用户，重新创建***
* *创建用户时，默认会创建一个和****用户名****同名的****组名***
* *用户信息保存在 /etc/passwd 文件中*

### 7.3.2 查看用户信息

| **序号** | **命令** | **作用** |
| --- | --- | --- |
| 01 | id [用户名] | 查看用户 UID 和 GID 信息 |
| 02 | who | 查看当前所有登录的用户列表 |
| 03 | whoami | 查看当前登录用户的账户名 |

#### passwd 文件

/etc/passwd 文件存放的是用户的信息，由 6 个分号组成的 7 个信息，分别是

1. 用户名
2. 密码（x，表示加密的密码）
3. UID（用户标识）
4. GID（组标识）
5. 用户全名或本地帐号
6. 家目录
7. 登录使用的 Shell，就是登录之后，使用的终端命令，ubuntu 默认是 dash

#### usermod

* usermod 可以用来设置 **用户** 的 **主组** ／ **附加组** 和 **登录 Shell**，命令格式如下：
* **主组**：通常在新建用户时指定，在 etc/passwd 的第 4 列 **GID 对应的组**
* **附加组**：在 etc/group 中最后一列表示该组的用户列表，用于指定 **用户的附加权限**

*提示：设置了用户的附加组之后，需要重新登录才能生效！*

# 修改用户的主组（passwd 中的 GID）

usermod -g 组 用户名

**# 修改用户的附加组**

**usermod -G 组 用户名**

# 修改用户登录 Shell

usermod -s /bin/bash 用户名

*注意：默认使用 useradd 添加的用户是没有权限使用 sudo 以 root 身份执行命令的，可以使用以下命令，将用户添加到 sudo 附加组中*

*备注：建立账户时，默认采用/bin/sh，ubuntu是/bin/bash，为了效果更好需要将新增用户设置为bash*

usermod -G sudo 用户名

#### which（重要）

*提示*

* */etc/passwd 是用于保存用户信息的文件*
* */usr/bin/passwd 是用于修改用户密码的程序*
* which 命令可以查看执行命令所在位置，例如：

which ls

# 输出

# /bin/ls

which useradd

# 输出

# /usr/sbin/useradd

#### bin 和 sbin

* 在 Linux 中，绝大多数可执行文件都是保存在 /bin、/sbin、/usr/bin、/usr/sbin
* /bin（binary）是二进制执行文件目录，主要用于具体应用
* /sbin（system binary）是系统管理员专用的二进制代码存放目录，主要用于系统管理
* /usr/bin（user commands for applications）后期安装的一些软件
* /usr/sbin（super user commands for applications）超级用户的一些管理程序

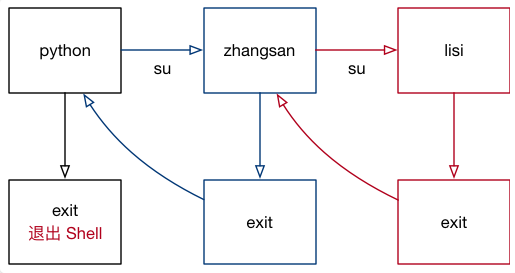
*提示：*

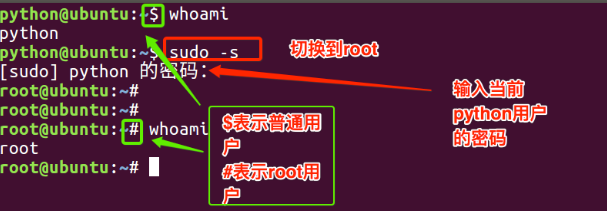
* *cd 这个终端命令是内置在系统内核中的，没有独立的文件，因此用 which 无法找到 cd 命令的位置*

### 7.3.3 切换用户

| **序号** | **命令** | **作用** | **说明** |
| --- | --- | --- | --- |
| 01 | su - 用户名 | 切换用户，并且切换目录 | - 可以切换到用户家目录，否则保持位置不变 |
| 02 | exit | 退出当前登录账户 |  |

* su 不接用户名，可以切换到 root，但是不推荐使用，因为不安全
* exit 示意图如下：





**7.4. 修改文件权限**

| **序号** | **命令** | **作用** |
| --- | --- | --- |
| 01 | chown | 修改拥有者 |
| 02 | chgrp | 修改组 |
| 03 | chmod | 修改权限 |

* 命令格式如下：

# 修改文件|目录的拥有者

chown 用户名 文件名|目录名

# 递归修改文件|目录的组

chgrp -R 组名 文件名|目录名

# 递归修改文件权限

chmod -R 755 文件名|目录名

* chmod 在设置权限时，可以简单地使用三个数字分别对应 **拥有者** ／ **组** 和 **其他** 用户的权限

# 直接修改文件|目录的 读|写|执行 权限，但是不能精确到 拥有者|组|其他

chmod +/-rwx 文件名|目录名



* 常见数字组合有（u表示用户／g表示组／o表示其他）：
  + 777 ===> u=rwx,g=rwx,o=rwx
  + 755 ===> u=rwx,g=rx,o=rx
  + 644 ===> u=rw,g=r,o=r

#### chmod 演练目标

1. 将 01.py 的权限修改为 u=rwx,g=rx,o=r
2. 将 123.txt 的权限修改为 u=rw,g=r,o=-
3. 将 test 目录以及目录下的 **所有** 文件权限修改为 u=rwx,g=rwx,o=rx

**八、系统信息相关命令**

* 本节内容主要是为了方便通过远程终端维护服务器时，查看服务器上当前 **系统日期和时间** ／ **磁盘空间占用情况** ／ **程序执行情况**
* 本小结学习的终端命令基本都是查询命令，通过这些命令对系统资源的使用情况有个了解

**目标**

* 时间和日期
* date
* cal
* 磁盘和目录空间
* df
* du
* 进程信息
* ps
* top
* kill

**8.1. 时间和日期**

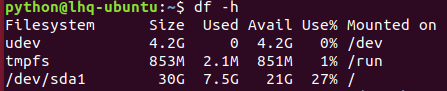
| **序号** | **命令** | **作用** |
| --- | --- | --- |
| 01 | date | 查看系统时间 |
| 02 | cal | calendar 查看日历，-y 选项可以查看一年的日历 |

**8.2. 磁盘信息**

| **序号** | **命令** | **作用** |
| --- | --- | --- |
| 01 | df -h | disk free 显示磁盘剩余空间 |
| 02 | du -h [目录名] | disk usage 显示目录下的文件大小 |

* 选项说明

| **参数** | **含义** |
| --- | --- |
| -h | 以人性化的方式显示文件大小 |



**8.3. 进程信息**

* 所谓 **进程**，通俗地说就是 **当前正在执行的一个程序**

| **序号** | **命令** | **作用** |
| --- | --- | --- |
| 01 | ps aux | process status 查看进程的详细状况 |
| 02 | top | 动态显示运行中的进程并且排序 |
| 03 | kill [-9] 进程代号 | 终止指定代号的进程，-9 表示强行终止 |

*ps 默认只会显示当前用户通过终端启动的应用程序*

* ps 选项说明功能

| **选项** | **含义** |
| --- | --- |
| a | 显示终端上的所有进程，包括其他用户的进程 |
| u | 显示进程的详细状态 |
| x | 显示没有控制终端的进程 |

*提示：使用 kill 命令时，最好只终止由当前用户开启的进程，而不要终止 root 身份开启的进程，否则可能导致系统崩溃*

* *要退出 top 可以直接输入 q*

**九、其他命令**

**目标**

* 查找文件
  + find
* 软链接
  + ln
* 打包和压缩
  + tar
* 软件安装
  + apt-get

**9.1. 查找文件**

* find 命令功能非常强大，通常用来在 **特定的目录下** **搜索** 符合条件的文件

| **序号** | **命令** | **作用** |
| --- | --- | --- |
| 01 | find [路径] -name "\*.py" | 查找指定路径下扩展名是 .py 的文件，包括子目录 |

* 如果省略路径，表示在当前文件夹下查找
* 之前学习的通配符，在使用 find 命令时同时可用
* 有关 find 的高级使用，在就业班会讲

### 演练目标

* 1. 搜索桌面目录下，文件名包含 1 的文件

find -name "\*1\*"

* 2. 搜索桌面目录下，所有以 .txt 为扩展名的文件

find -name "\*.txt"

* 3. 搜索桌面目录下，以数字 1 开头的文件

find -name "1\*"

常用用法：

| **命令** | **含义** |
| --- | --- |
| find ./ -name test.sh | 查找当前目录下所有名为test.sh的文件 |
| find ./ -name '\*.sh' | 查找当前目录下所有后缀为.sh的文件 |
| find ./ -name "[A-Z]\*" | 查找当前目录下所有以大写字母开头的文件 |
| find /tmp -size 2M | 查找在/tmp 目录下等于2M的文件 |
| find /tmp -size +2M | 查找在/tmp 目录下大于2M的文件 |
| find /tmp -size -2M | 查找在/tmp 目录下小于2M的文件 |
| find ./ -size +4k -size -5M | 查找当前目录下大于4k，小于5M的文件 |
| find ./ -perm 777 | 查找当前目录下权限为 777 的文件或目录 |

**9.2. 软链接**

| **序号** | **命令** | **作用** |
| --- | --- | --- |
| 01 | ln -s 被链接的源文件 链接文件 | 建立文件的软链接，用通俗的方式讲**类似于** Windows 下的**快捷方式** |

* 注意：
* 1. 没有 -s 选项建立的是一个 **硬链接文件**
  + 两个文件占用相同大小的硬盘空间，**工作中几乎不会建立文件的硬链接**
* 2. **源文件要使用绝对路径**，不能使用相对路径，这样可以方便移动链接文件后，仍然能够正常使用

### 演练目标

* 1. 将桌面目录下的 01.py **移动**到 demo/b/c 目录下
* 2. 在桌面目录下新建 01.py 的 **软链接** FirstPython
  + 分别使用 **相对路径** 和 **绝对路径** 建立 FirstPython 的软链接
* 3. 将 FirstPython **移动**到 demo 目录下，对比使用 **相对路径** 和 **绝对路径** 的区别

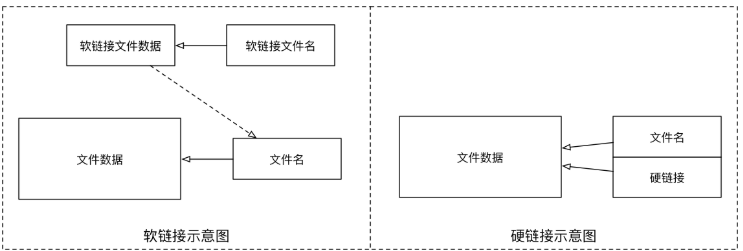
### 硬链接简介（知道）

* 在使用 ln 创建链接时，如果没有 -s 选项，会创建一个 **硬链接**，而不是软链接

#### 硬链接演练

* 1. 在 ~/Desktop/demo 目录下建立 ~/Desktop/demo/b/c/01.py 的硬链接 01\_hard
* 2. 使用 ls -l 查看文件的**硬链接数**（硬链接——有多少种方式可以访问文件或者目录）
* 3. 删除 ~/Desktop/demo/b/c/01.py，并且使用 tree 来确认 demo 目录下的三个链接文件

#### 文件软硬链接的示意图



*在 Linux 中，****文件名****和****文件的数据****是分开存储的*

* 提示：
  + 在 Linux 中，只有文件的 硬链接数 == 0 才会被删除
  + 使用 ls -l 可以查看一个文件的硬链接的数量
  + 在日常工作中，几乎不会建立文件的硬链接，知道即可

Linux链接文件类似于Windows下的快捷方式。

链接文件分为软链接和硬链接。

软链接：软链接不占用磁盘空间，源文件删除则软链接失效。

硬链接：硬链接只能链接普通文件，不能链接目录。

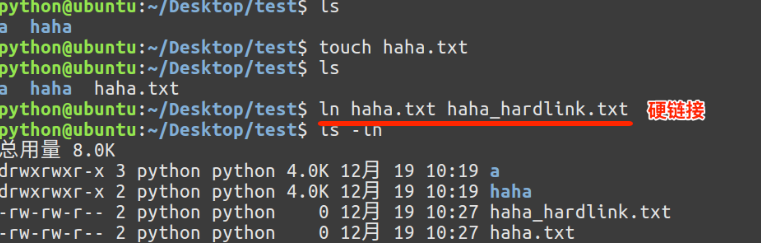
使用格式：

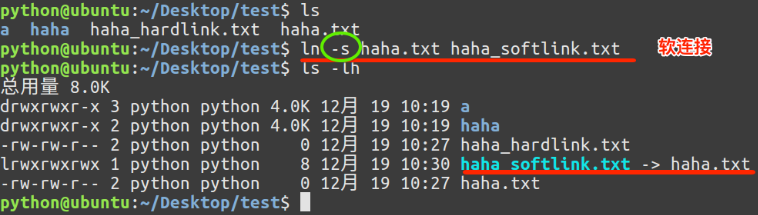
ln 源文件 链接文件

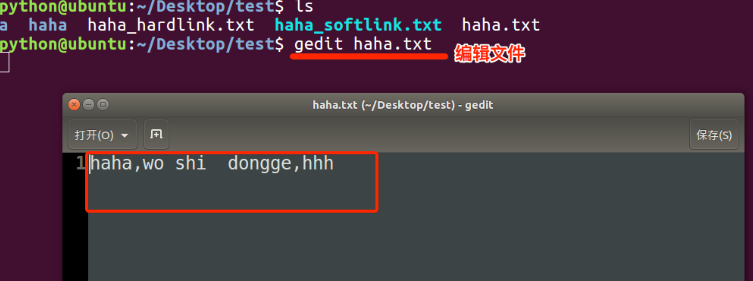
ln -s 源文件 链接文件

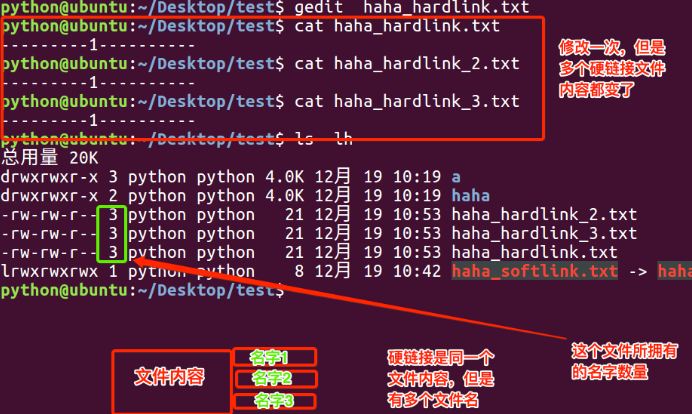
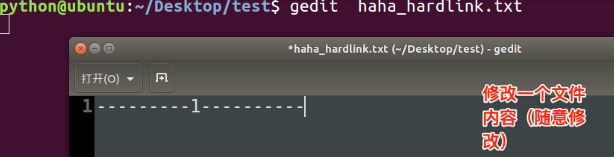
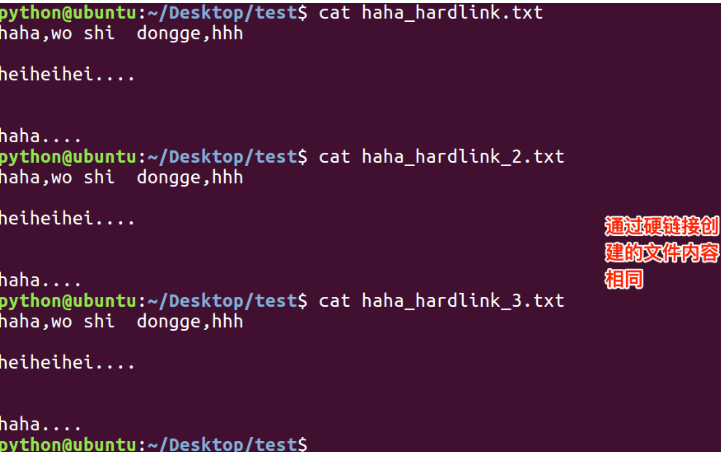
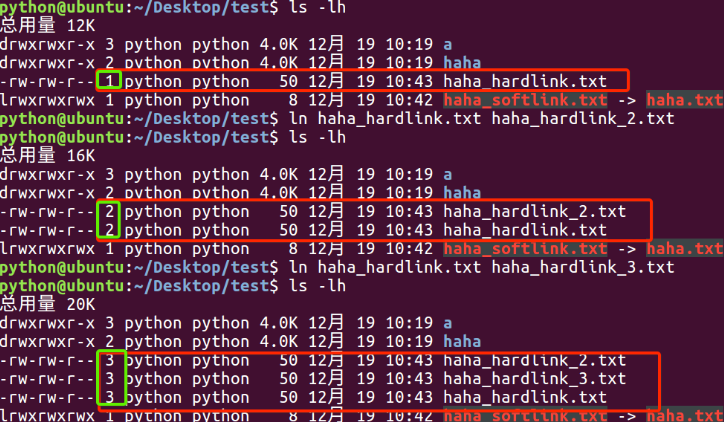
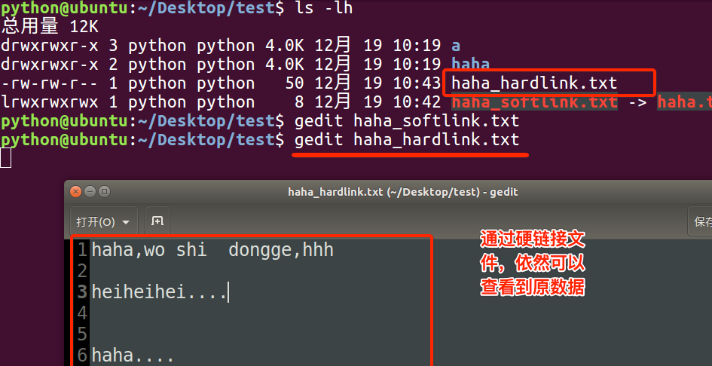
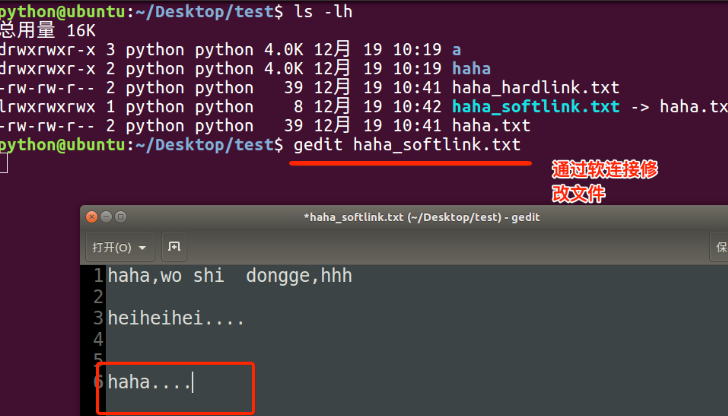
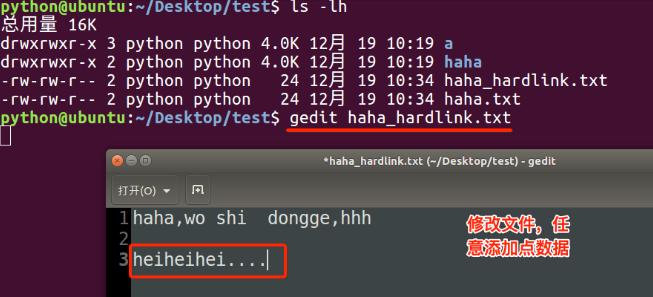
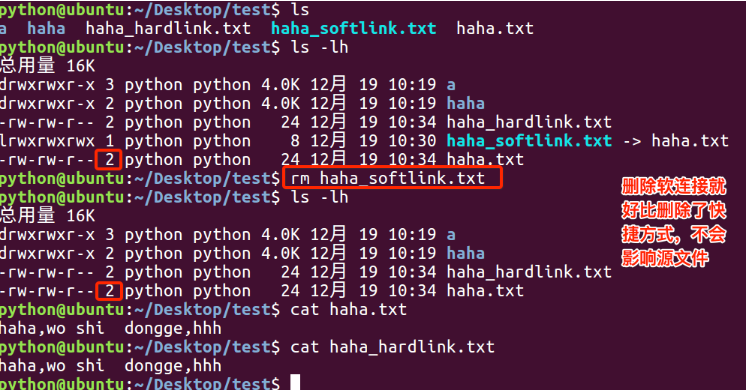
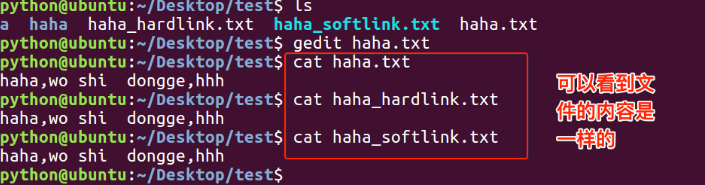
如果没有-s选项代表建立一个硬链接文件，两个文件占用相同大小的硬盘空间，即使删除了源文件，链接文件还是存在，所以-s选项是更常见的形式。

注意：如果软链接文件和源文件不在同一个目录，源文件要使用绝对路径，不能使用相对路径。









**9.3. 打包压缩**

* **打包压缩** 是日常工作中备份文件的一种方式
* 在不同操作系统中，常用的打包压缩方式是不同的
  + Windows 常用 rar
  + Mac 常用 zip
  + Linux 常用 tar.gz

### 9.3.1 打包 ／ 解包

* tar 是 Linux 中最常用的 **备份**工具，此命令可以 **把一系列文件** 打包到 **一个大文件中**，也可以把一个 **打包的大文件恢复成一系列文件**
* tar 的命令格式如下：

# 打包文件

tar -cvf 打包文件.tar 被打包的文件／路径...

# 解包文件

tar -xvf 打包文件.tar

* tar 选项说明

| **选项** | **含义** |
| --- | --- |
| c | 生成档案文件，创建打包文件 |
| x | 解开档案文件 |
| v | 列出归档解档的详细过程，显示进度 |
| f | 指定档案文件名称，f 后面一定是 .tar 文件，所以必须放选项最后 |

*注意：f 选项必须放在最后，其他选项顺序可以随意*

#### 打包解包演练

1. 删除桌面下的所有内容
2. 在桌面下新建三个空白文件 01.py、02.py、03.py
3. 将这三个文件打一个 py.tar 的包
4. 新建 tar 目录，并且将 py.tar 移动到 tar 目录下
5. 解包 py.tar

### 9.3.2 压缩／解压缩

#### 1) gzip

* tar 与 gzip 命令结合可以使用实现文件 **打包和压缩**
  + tar 只负责打包文件，但不压缩
  + 用 gzip 压缩 tar 打包后的文件，其扩展名一般用 xxx.tar.gz

*在 Linux 中，最常见的压缩文件格式就是 xxx.tar.gz*

* 在 tar 命令中有一个选项 **-z** 可以调用 gzip，从而可以方便的实现压缩和解压缩的功能
* 命令格式如下：

# 压缩文件

tar -zcvf 打包文件.tar.gz 被压缩的文件／路径...

# 解压缩文件

tar -zxvf 打包文件.tar.gz

# 解压缩到指定路径

tar -zxvf 打包文件.tar.gz -C 目标路径

| **选项** | **含义** |
| --- | --- |
| -C | 解压缩到指定目录，注意：要解压缩的目录必须存在 |

#### 2) bzip2(two)

* tar 与 bzip2 命令结合可以使用实现文件 **打包和压缩**（用法和 gzip 一样）
  + tar 只负责打包文件，但不压缩，
  + 用 bzip2 压缩 tar 打包后的文件，其扩展名一般用 xxx.tar.bz2
* 在 tar 命令中有一个选项 **-j** 可以调用 bzip2，从而可以方便的实现压缩和解压缩的功能
* 命令格式如下：

# 压缩文件

tar -jcvf 打包文件.tar.bz2 被压缩的文件／路径...

# 解压缩文件

tar -jxvf 打包文件.tar.bz2

**9.4. 软件安装**

### 9.4.1 通过 apt 安装／卸载软件

* apt 是 Advanced Packaging Tool，是 Linux 下的一款安装包管理工具
* 可以在终端中方便的 **安装**／**卸载**／**更新软件包**

# 1. 安装软件

$ sudo apt install 软件包

# 2. 卸载软件

$ sudo apt remove 软件名

# 3. 更新已安装的包

$ sudo apt upgrade

sudo apt remove package 删除包

sudo apt show package 获取包的相关信息，如说明、大小、版本等

sudo apt update 更新源

sudo apt upgrade 更新已安装的包

apt list --upgradeable：显示可升级的软件包。

apt list --installed：显示已安装的软件包。

老版本中安装程序需要使用sudo apt-get install <软件包名>，现在新版本中已建议直接使用apt替代apt-get命令。  
**sudo apt install -f  
使用此命令可修复依赖关系，假如有软件因依赖关系不满足而无法安装，就可以运行此命令自动修复安装程序包所依赖的包。**

# 一个小火车提示

$ sudo apt install sl

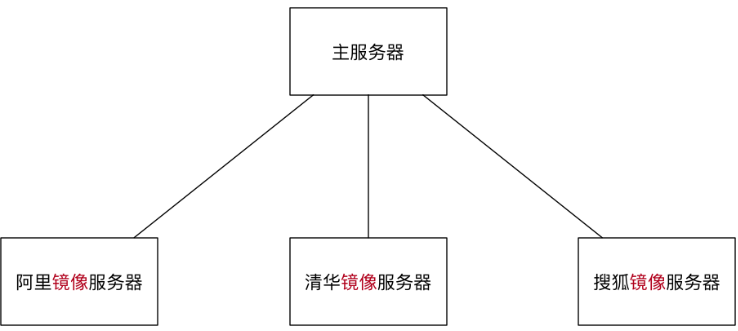
# 一个比较漂亮的查看当前进程排名的软件

$ sudo apt install htop

### 9.4.2 配置软件源

* 如果希望在 ubuntu 中安装软件，**更加快速**，可以通过设置**镜像源**，选择一个访问网速更快的服务器，来提供软件下载／安装服务
* 提示：更换服务器之后，需要一个相对比较长时间的更新过程，需要耐心等待。更新完成后，再安装软件都会从新设置的服务器下载软件了

*所谓镜像源，就是****所有服务器的内容是相同的（镜像）****，但是根据所在位置不同，国内服务器通常速度会更快一些！*





### 9.4.3 手动安装Deb软件包

#### dpkg -i

如果你从网站上下载了一个deb软件包，那么你需要dpkg工具来安装这个deb包．例如，你可以[下载WPS linux版](http://community.wps.cn/download/)．然后，在终端里将当前工作目录切换到下载目录，再输入下面的命令安装WPS  Linux版．

sudo dpkg -i wps-office\*.deb

#### gdebi

dpkg的一个缺点是它不能解决依赖关系．你必须手动安装依赖包．而gdebi可以帮助我们自动安装依赖包．输入下面的命令安装gdebi

sudo apt-get install gdebi

它的命令语法如下：

sudo gdebi <package.deb>

### 9.4.4 安装.rpm、.tar、.tar.gz

#### 1. 安装rpm包

说起RPM（REDHAT Package Management）标准的软件包，大家可能都会想起大名鼎鼎的 REDHAT 公司，正是RPM软件包发行方式的出现，使Linux中的应用软件安装变得简单。如果用户想安装按RPM方式发行的软件，只需简单地输入命令“rpm －ivh xxx．rpm”即可。比如用户想安装 OpenOffice．org－1．0．1．rpm 软件包，只需输入命令“rpm －ivh OpenOffice．org－1．0．1．rpm”即可。RPM软件包发行方式的另一个优点是它能够方便地对已经安装的RPM软件包进行删除， 只要使用 “rpm －e OpenOffice．org－1．0．1”命令就能将刚才安装的 OpenOffice．org－1．0．1．rpm 从硬盘上安全永久地删除。如果你是在 X－Window 环境中安装/删除软件，那便有更好的办法，如果使用的是KDE，可以使用KDE自带的Kpackage程序来对软件进行添加或删除，如果是使用 GNOME，则可以使用Gnorpm程序对软件进行管理。这两个程序都很像微软Windows中的“添加/删除程序”功能。以RPM软件包发行方式的软件 是最容易安装和管理的，建议大家以后多采用这种模式来管理软件。  
**安装方法：**rpm -ivh example-1.2.3-1.rpm  
**查看安装目录：** rpm -ql example-1.2.3-1  
**卸载**：rpm -e example-1.2.3-1

#### 2. 安装tar包

**安装方法：**  
（1）解压tar包： tar -xvf example-1.2.3-1.tar  
（2）进入到解压后的目录下：cd  example-1.2.3-1  
（3）再依次执行：

[root@jmwang example-1.2.3-1]# ./configure（有的版本无此命令，则可跳过这一步骤）

[root@jmwang example-1.2.3-1]# make

[root@jmwang example-1.2.3-1]# make install

[root@jmwang example-1.2.3-1]# make clear（非必需）

[root@jmwang example-1.2.3-1]# make distclear（非必需）

这样就完成了tar包的安装。

#### 3. 安装tar.gz(tgz)包

以tar.gz为扩展名的软件包，是用tar程序打包并用Gzip程序压缩的软件包。要安装这种软件包，需要先对软件包进行解压缩，使用 “tar －zxfv filename．tar．gz”可以对软件包进行解压缩，解压缩所得的文件在以Filename为名的目录中。进入该目录，可以看到解压缩出来的文件 了。各种软件都有不同的安装方法，但是一般每个软件包解压缩后都有Install和Readme文件，帮助文件中会有详细的安装指导。以 tar．gz（或tgz）包发行的软件有一个缺点，就是一般不带自动反安装程序，如果需要对已经安装的此类程序进行删除，就不得不仔细查看 Makefile 中的安装路径和文件名，这些对于初学者有一些难度。  
  
**安装方法：**  
（1）解压tar.gz包： tar -zxvf example-1.2.3-1.tar.gz  
（2）后方法和example-1.2.3-1.tar包一样

#### 4.安装.sh文件

本文介绍Linux下面用命令如何运行.sh文件的方法，有两种方法：

一、直接./加上文件名.sh，如运行hello.sh为./hello.sh【hello.sh必须有x权限】

二、直接sh 加上文件名.sh，如运行hello.sh为sh hello.sh【hello.sh可以没有x权限】

**十、vi —— 终端中的编辑器**

**目标**

* vi 简介
* 打开和新建文件
* 三种工作模式
* 常用命令
* 分屏命令
* 常用命令速查图

**10.1. vi 简介**

### 10.1.1 学习 vi 的目的

* 在工作中，要对 **服务器** 上的文件进行 **简单** 的修改，可以使用 ssh 远程登录到服务器上，并且使用 vi 进行快速的编辑即可
* 常见需要修改的文件包括：
* 源程序
* 配置文件，例如 ssh 的配置文件 ~/.ssh/config
* 在没有图形界面的环境下，要编辑文件，vi 是最佳选择！
* 每一个要使用 Linux 的程序员，都应该或多或少的学习一些 vi 的常用命令

### 10.1.2 vi 和 vim

* 在很多 Linux 发行版中，直接把 vi 做成 vim 的软连接

#### vi

* vi 是 Visual interface 的简称，是 Linux 中 **最经典** 的文本编辑器
* vi 的核心设计思想 —— **让程序员的手指始终保持在键盘的核心区域，就能完成所有的编辑操作**



* vi 的特点：
* **没有图形界面** 的 **功能强大** 的编辑器
* 只能是编辑 **文本内容**，不能对字体、段落进行排版
* **不支持鼠标操作**
* **没有菜单**
* **只有命令**
* vi 编辑器在 **系统管理**、**服务器管理** 编辑文件时，**其功能永远不是图形界面的编辑器能比拟的**

#### vim

**vim = vi improved**

* vim 是从 vi 发展出来的一个文本编辑器，支持 **代码补全**、**编译** 及 **错误跳转** 等方便编程的功能特别丰富，在程序员中被广泛使用，被称为 **编辑器之神**

#### 查询软连接命令（知道）

* 在很多 Linux 发行版中直接把 vi 做成 vim 的软连接

# 查找 vi 的运行文件  
​  
$ which vi  
$ ls -l /usr/bin/vi  
$ ls -l /etc/alternatives/vi  
$ ls -l /usr/bin/vim.basic  
​  
# 查找 vim 的运行文件  
$ which vim  
$ ls -l /usr/bin/vim  
$ ls -l /etc/alternatives/vim  
$ ls -l /usr/bin/vim.basic

**10.2. 打开和新建文件**

* 在终端中输入 vi **在后面跟上文件名** 即可

$ vi 文件名

* 如果文件已经存在，会直接打开该文件
* 如果文件不存在，会新建一个文件

### 10.2.1 打开文件并且定位行

* 在日常工作中，有可能会遇到 **打开一个文件，并定位到指定行** 的情况
* 例如：在开发时，**知道某一行代码有错误**，可以 **快速定位** 到出错代码的位置
* 这个时候，可以使用以下命令打开文件

$ vi 文件名 +行数

提示：如果只带上 + 而不指定行号，会直接定位到文件末尾

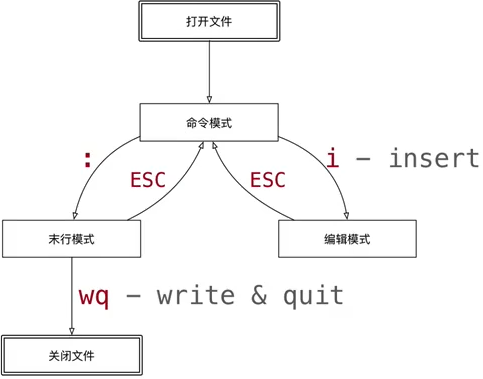
### 10.2.2 异常处理

* 如果 vi 异常退出，在磁盘上可能会保存有 **交换文件**
* 下次再使用 vi 编辑该文件时，会看到以下屏幕信息，按下字母 d 可以 **删除交换文件** 即可

提示：按下键盘时，注意关闭输入法

**10.3. 三种工作模式**

* vi 有三种基本工作模式：
  1. **命令模式**
     + **打开文件首先进入命令模式**，是使用 vi 的 **入口**
     + 通过 **命令** 对文件进行常规的编辑操作，例如：**定位**、**翻页**、**复制**、**粘贴**、**删除**……
     + 在其他图形编辑器下，通过 **快捷键** 或者 **鼠标** 实现的操作，都在 **命令模式** 下实现
  2. **末行模式** —— 执行 **保存**、**退出** 等操作
     + 要退出 vi 返回到控制台，需要在末行模式下输入命令
     + **末行模式** 是 vi 的 **出口**
  3. **编辑模式** —— 正常的编辑文字



提示：在 Touch Bar 的 Mac 电脑上 ，按 ESC 不方便，可以使用 CTRL + [ 替代

### 末行模式命令

| **命令** | **英文** | **功能** |
| --- | --- | --- |
| w | write | 保存 |
| q | quit | 退出，如果没有保存，不允许退出 |
| q! | quit | 强行退出，不保存退出 |
| wq | write & quit | 保存并退出 |
| x |  | 保存并退出 |

**10.4. 常用命令**

### 命令线路图

1. 重复次数

* 在命令模式下，**先输入一个数字**，**再跟上一个命令**，可以让该命令 **重复执行指定次数**

1. 移动和选择（**多练**）

* vi 之所以快，关键在于 **能够快速定位到要编辑的代码行**
* **移动命令** 能够 和 **编辑操作** 命令 **组合使用**

1. 编辑操作

* **删除**、**复制**、**粘贴**、**替换**、**缩排**

1. 撤销和重复
2. 查找替换
3. 编辑

#### 学习提示

1. vi 的命令较多，**不要期望一下子全部记住**，个别命令忘记了，只是会影响编辑速度而已
2. 在使用 vi 命令时，注意 **关闭中文输入法**

### 10.4.1 移动（基本）

* 要熟练使用 vi，首先应该学会怎么在 **命令模式** 下样快速移动光标
* **编辑操作命令**，能够和 **移动命令** 结合在一起使用

#### 1) 上、下、左、右

| **命令** | **功能** | **手指** |
| --- | --- | --- |
| h | 向左 | 食指 |
| j | 向下 | 食指 |
| k | 向上 | 中指 |
| l | 向右 | 无名指 |



#### 2) 行内移动

| **命令** | **英文** | **功能** |
| --- | --- | --- |
| w | word | 向后移动一个单词 |
| b | back | 向前移动一个单词 |
| 0 |  | 行首 |
| ^ |  | 行首，第一个不是空白字符的位置 |
| $ |  | 行尾 |

#### 3) 行数移动

| **命令** | **英文** | **功能** |
| --- | --- | --- |
| gg | go | 文件顶部 |
| G | go | 文件末尾 |
| 数字gg | go | 移动到 数字 对应行数 |
| 数字G | go | 移动到 数字 对应行数 |
| :数字 |  | 移动到 数字 对应行数 |

#### 4) 屏幕移动

| **命令** | **英文** | **功能** |
| --- | --- | --- |
| Ctrl + b | back | 向上翻页 |
| Ctrl + f | forward | 向下翻页 |
| H | Head | 屏幕顶部 |
| M | Middle | 屏幕中间 |
| L | Low | 屏幕底部 |

### 10.4.2 移动（程序）

#### 1) 段落移动

* vi 中使用 空行 来区分段落
* 在程序开发时，通常 **一段功能相关的代码会写在一起** —— 之间没有空行

| **命令** | **功能** |
| --- | --- |
| { | 上一段 |
| } | 下一段 |

#### 2) 括号切换

* 在程序世界中，()、[]、{} 使用频率很高，而且 **都是成对出现的**

| **命令** | **功能** |
| --- | --- |
| % | 括号匹配及切换 |

#### 3) 标记

* 在开发时，某一块代码可能**需要稍后处理**，例如：编辑、查看
* 此时先使用 m 增加一个标记，这样可以 **在需要时快速地跳转回来** 或者 **执行其他编辑操作**
* **标记名称** 可以是 a~z 或者 A~Z 之间的任意 **一个** 字母
* 添加了标记的 **行如果被删除**，**标记同时被删除**
* 如果 **在其他行添加了相同名称的标记**，**之前添加的标记也会被替换掉**

| **命令** | **英文** | **功能** |
| --- | --- | --- |
| mx | mark | 添加标记 x，x 是 a~z 或者 A~Z 之间的任意一个字母 |
| 'x |  | 直接定位到标记 x 所在位置 |

### 10.4.3 选中文本（可视模式）

* 学习 复制 命令前，应该先学会 **怎么样选中 要复制的代码**
* 在 vi 中要选择文本，需要先使用 Visual 命令切换到 **可视模式**
* vi 中提供了 **三种** 可视模式，可以方便程序员选择 **选中文本的方式**
* 按 ESC 可以放弃选中，返回到 **命令模式**

| **命令** | **模式** | **功能** |
| --- | --- | --- |
| v | 可视模式 | 从光标位置开始按照正常模式选择文本 |
| V | 可视行模式 | 选中光标经过的完整行 |
| Ctrl + v | 可视块模式 | 垂直方向选中文本 |

* **可视模式**下，可以和 **移动命令** 连用，例如：ggVG 能够选中所有内容

### 10.4.4 撤销和恢复撤销

* 在学习编辑命令之前，先要知道怎样撤销之前一次 **错误的** 编辑动作！

| **命令** | **英文** | **功能** |
| --- | --- | --- |
| u | undo | 撤销上次命令 |
| CTRL + r | redo | 恢复撤销的命令 |

### 10.4.5 删除文本

| **命令** | **英文** | **功能** |
| --- | --- | --- |
| x | cut | 删除光标所在字符，或者选中文字 |
| d(移动命令) | delete | 删除移动命令对应的内容 |
| dd | delete | 删除光标所在行，可以 ndd 复制多行 |
| D | delete | 删除至行尾 |

提示：如果使用 **可视模式** 已经选中了一段文本，那么无论使用 d 还是 x，都可以删除选中文本

* 删除命令可以和 **移动命令** 连用，以下是常见的组合命令：

\* dw       # 从光标位置删除到单词末尾  
\* d0       # 从光标位置删除到一行的起始位置  
\* d}       # 从光标位置删除到段落结尾  
\* ndd       # 从光标位置向下连续删除 n 行  
\* d代码行G   # 从光标所在行 删除到 指定代码行 之间的所有代码  
\* d'a       # 从光标所在行 删除到 标记a 之间的所有代码

### 10.4.6 复制、粘贴

* vi 中提供有一个 **被复制文本的缓冲区**
  + **复制** 命令会将选中的文字保存在缓冲区
  + **删除** 命令删除的文字会被保存在缓冲区
  + 在需要的位置，使用 **粘贴** 命令可以将缓冲区的文字插入到光标所在位置

| **命令** | **英文** | **功能** |
| --- | --- | --- |
| y(移动命令) | copy | 复制 |
| yy | copy | 复制一行，可以 nyy 复制多行 |
| p | paste | 粘贴 |

**提示**

* 命令 d、x 类似于图形界面的 **剪切操作** —— CTRL + X
* 命令 y 类似于图形界面的 **复制操作** —— CTRL + C
* 命令 p 类似于图形界面的 **粘贴操作** —— CTRL + V
* vi 中的 **文本缓冲区同样只有一个**，如果后续做过 **复制、剪切** 操作，之前缓冲区中的内容会被替换

**注意**

* vi 中的 **文本缓冲区** 和系统的 **剪贴板** 不是同一个
* 所以在其他软件中使用 CTRL + C 复制的内容，不能在 vi 中通过 P 命令粘贴
* 可以在 **编辑模式** 下使用 **鼠标右键粘贴**

### 10.4.7 替换

| **命令** | **英文** | **功能** | **工作模式** |
| --- | --- | --- | --- |
| r | replace | 替换当前字符 | 命令模式 |
| R | replace | 替换当前行光标后的字符 | 替换模式 |

* R 命令可以进入 **替换模式**，替换完成后，按下 ESC 可以回到 **命令模式**
* **替换命令** 的作用就是不用进入 **编辑模式**，对文件进行 **轻量级的修改**

### 10.4.8 缩排和重复执行

| **命令** | **功能** |
| --- | --- |
| >> | 向右增加缩进 |
| << | 向左减少缩进 |
| . | 重复上次命令 |

* **缩排命令** 在开发程序时，**统一增加代码的缩进** 比较有用！
  + 一次性 **在选中代码前增加 4 个空格**，就叫做 **增加缩进**
  + 一次性 **在选中代码前删除 4 个空格**，就叫做 **减少缩进**
* 在 **可视模式** 下，缩排命令只需要使用 **一个** > 或者 <

在程序中，**缩进** 通常用来表示代码的归属关系

* 前面空格越少，代码的级别越高
* 前面空格越多，代码的级别越低

### 10.4.9 查找

#### 常规查找

| **命令** | **功能** |
| --- | --- |
| /str | 查找 str |

* 查找到指定内容之后，使用 Next 查找下一个出现的位置：
  + n: 查找下一个
  + N: 查找上一个
* 如果不想看到高亮显示，可以随便查找一个文件中不存在的内容即可

#### 单词快速匹配

| **命令** | **功能** |
| --- | --- |
| \* | 向后查找当前光标所在单词 |
| # | 向前查找当前光标所在单词 |

* 在开发中，通过单词快速匹配，可以快速看到这个单词在其他什么位置使用过

### 10.4.10 查找并替换

* 在 vi 中查找和替换命令需要在 **末行模式** 下执行
* 记忆命令格式：

:%s///g

#### 1) 全局替换

* **一次性**替换文件中的 **所有出现的旧文本**
* 命令格式如下：

:%s/旧文本/新文本/g

#### 2) 可视区域替换

* **先选中** 要替换文字的 **范围**
* 命令格式如下：

:s/旧文本/新文本/g

#### 3) 确认替换

* 如果把末尾的 g 改成 gc 在替换的时候，会有提示！**推荐使用！**

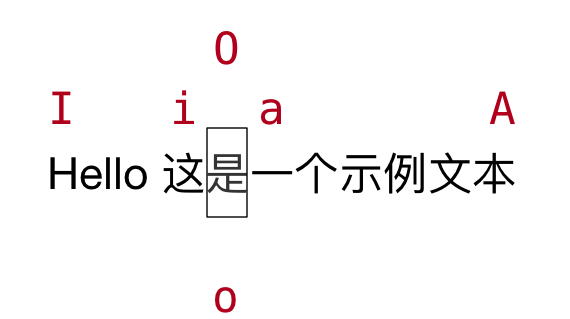
:%s/旧文本/新文本/gc

1. y - yes 替换
2. n - no 不替换
3. a - all 替换所有
4. q - quit 退出替换
5. l - last 最后一个，并把光标移动到行首
6. ^E 向下滚屏
7. ^Y 向上滚屏

### 10.4.11 插入命令

* 在 vi 中除了常用的 i 进入 **编辑模式** 外，还提供了以下命令同样可以进入编辑模式：

| **命令** | **英文** | **功能** | **常用** |
| --- | --- | --- | --- |
| i | insert | 在当前字符前插入文本 | 常用 |
| I | insert | 在行首插入文本 | 较常用 |
| a | append | 在当前字符后添加文本 |  |
| A | append | 在行末添加文本 | 较常用 |
| o |  | 在当前行后面插入一空行 | 常用 |
| O |  | 在当前行前面插入一空行 | 常用 |



#### 演练 1 —— 编辑命令和数字连用

* 在开发中，可能会遇到连续输入 N 个同样的字符

在 Python 中有简单的方法，但是其他语言中通常需要自己输入

* 例如：\*\*\*\*\*\*\*\*\*\* 连续 10 个星号

要实现这个效果可以在 **命令模式** 下

1. 输入 10，表示要重复 10 次
2. 输入 i 进入 **编辑模式**
3. 输入 \* 也就是重复的文字
4. 按下 ESC 返回到 **命令模式**，返回之后 vi 就会把第 2、3 两步的操作重复 10 次

提示：正常开发时，在 **进入编辑模式之前，不要按数字**

#### 演练 2 —— 利用 可视块 给多行代码增加注释

* 在开发中，可能会遇到一次性给多行代码 **增加注释** 的情况

在 Python 中，要给代码增加注释，可以在代码前增加一个 #

要实现这个效果可以在 **命令模式** 下

1. 移动到要添加注释的 **第 1 行代码**，按 ^ 来到行首
2. 按 CTRL + v 进入 **可视块** 模式
3. 使用 j 向下连续选中要添加的代码行
4. 输入 I 进入 **编辑模式**，并在 **行首插入**，注意：一定要使用 **I**
5. 输入 # 也就是注释符号
6. 按下 ESC 返回到 **命令模式**，返回之后 vi 会在之前选中的每一行代码 **前** 插入 #

**10.5. 分屏命令**

* 属于 vi 的高级命令 —— 可以 **同时编辑和查看多个文件**

### 10.5.1 末行命令扩展

**末行命令** 主要是针对文件进行操作的：**保存**、**退出**、**保存&退出**、**搜索&替换**、**另存**、**新建**、**浏览文件**

| **命令** | **英文** | **功能** |
| --- | --- | --- |
| :e . | edit | 会打开内置的文件浏览器，浏览要当前目录下的文件 |
| :n 文件名 | new | 新建文件 |
| :w 文件名 | write | 另存为，但是仍然编辑当前文件，并不会切换文件 |

提示：切换文件之前，必须保证当前这个文件已经被保存！

* 已经学习过的 **末行命令**：

| **命令** | **英文** | **功能** |
| --- | --- | --- |
| :w | write | 保存 |
| :q | quit | 退出，如果没有保存，不允许退出 |
| :q! | quit | 强行退出，不保存退出 |
| :wq | write & quit | 保存并退出 |
| :x |  | 保存并退出 |
| :%s///gc |  | 确认搜索并替换 |

在实际开发中，可以使用 w 命令 **阶段性的备份代码**

### 10.5.2 分屏命令

* 使用 **分屏命令**，可以 **同时编辑和查看多个文件**

| **命令** | **英文** | **功能** |
| --- | --- | --- |
| :sp [文件名] | split | 横向增加分屏 |
| :vsp [文件名] | vertical split | 纵向增加分屏 |

#### 1) 切换分屏窗口

分屏窗口都是基于 CTRL + W 这个快捷键的，w 对应的英文单词是 window

| **命令** | **英文** | **功能** |
| --- | --- | --- |
| w | window | 切换到下一个窗口 |
| r | reverse | 互换窗口 |
| c | close | 关闭当前窗口，但是不能关闭最后一个窗口 |
| q | quit | 退出当前窗口，如果是最后一个窗口，则关闭 vi |
| o | other | 关闭其他窗口 |

#### 2) 调整窗口大小

分屏窗口都是基于 CTRL + W 这个快捷键的，w 对应的英文单词是 window

| **命令** | **英文** | **功能** |
| --- | --- | --- |
| + |  | 增加窗口高度 |
| - |  | 减少窗口高度 |
| > |  | 增加窗口宽度 |
| < |  | 减少窗口宽度 |
| = |  | 等分窗口大小 |

调整窗口宽高的命令可以和数字连用，例如：5 CTRL + W + 连续 5 次增加高度

**10.6. 常用命令速查图**

vimrc

* vimrc 是 vim 的配置文件，可以设置 vim 的配置，包括：**热键**、**配色**、**语法高亮**、**插件** 等
* Linux 中 vimrc 有两个位置，**家目录下的配置文件优先级更高**

/etc/vim/vimrc  
~/.vimrc

* 常用的插件有：
  + 代码补全
  + 代码折叠
  + 搜索
  + Git 集成
  + ……
* 网上有很多高手已经配置好的针对 python 开发的 vimrc 文件，可以下载过来直接使用，或者等大家多 Linux 比较熟悉后，再行学习！

