

Table of Contents

课程介绍	0
Linux、命令	1
认识Linux	1.1
Linux的不同版本以及应用领域	1.2
文件和目录	1.3
Linux命令概述	1.4
Linux命令-文件	1.5
Linux命令-系统管理、磁盘管理	1.6
Linux命令-用户、权限管理	1.7
编辑器、服务器	2
gedit编辑器	2.1
sublime编辑器	2.2
编辑器之神-vim	2.3
ubuntu软件安装与卸载方法	2.4
Linux常用服务器构建-ftp服务器	2.5
Linux常用服务器构建-ssh和scp	2.6
Linux常用服务器构建-samba	2.7
附录-vim分屏操作	2.8

课程介绍

Python基础班共有13天的课程

第1-2天：Linux操作相关

通过2天的学习与操作，让大家从很陌生达到灵活操作

第3-10天：Python基础语法

涵盖Python基础的所有知识，让大家掌握编程的能力

第11天：Python强化练习

企业面试题练习与python知识强化

第12~13天：项目实战

通过Python基础班知识的学习，进行应用编程，完成第一个Python项目

$$1.01^{365} = 37.8$$

$$0.99^{365} = 0.03$$

积跬步以至千里，积懈怠以致深渊。

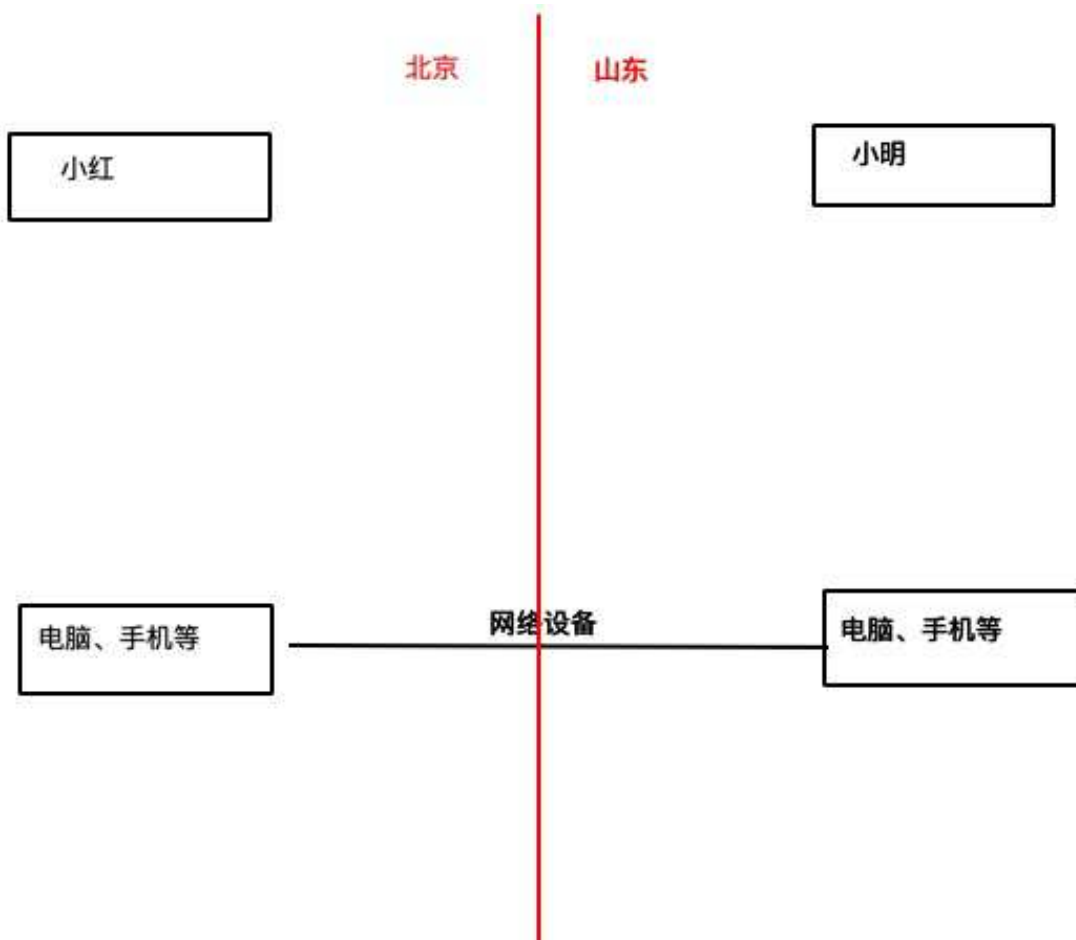
$$1.02^{365} = 1377.4$$

$$0.98^{365} = 0.0006$$

只比你努力一点的人，其实已经甩你太远。

认识Linux

1.什么是操作系统



2.现实生活中的操作系统

win7



Mac



Android



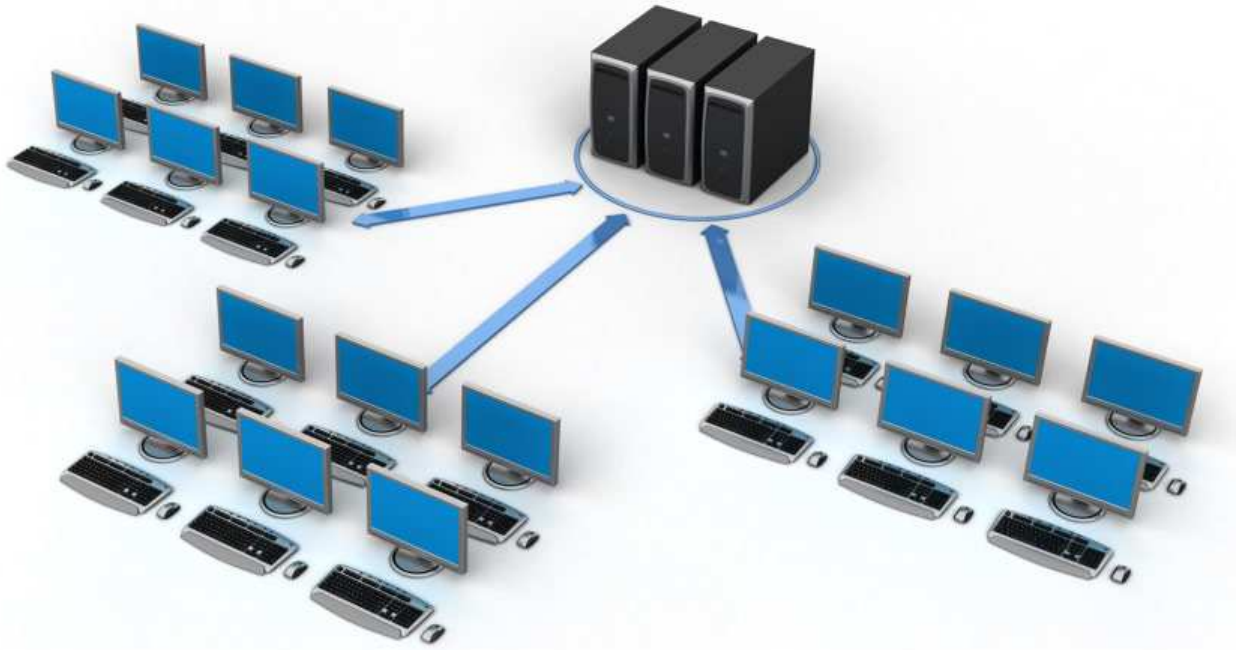
iOS



3. 操作系统的发展史

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年左右由贝尔实验室、麻省理工学院 以及 通用电气共同发起了Multics项目，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”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 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。

因为B语言的跨平台性较差，为了能够在其他的电脑上也能够运行这个非常棒的Unix操作系统，Dennis Ritchie和Ken Thompson 从B语言的基础上准备研究一个更好的语言



肯·汤普逊（左）和丹尼斯·里奇（右）

1972年，美国贝尔实验室的 Dennis 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

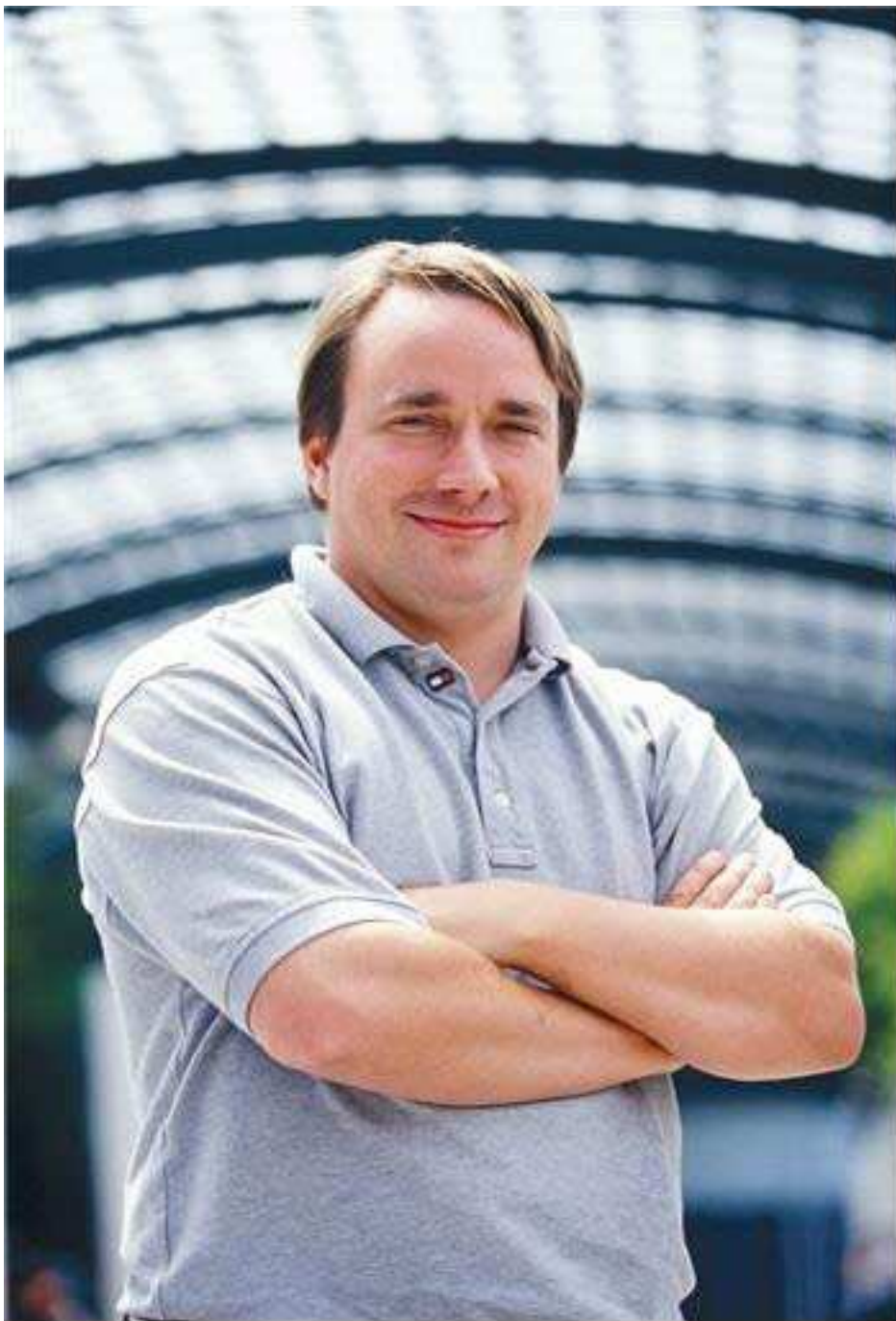
Minix

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实务细节，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。他以小型UNIX (mini-UNIX) 之意，将它称为MINIX。

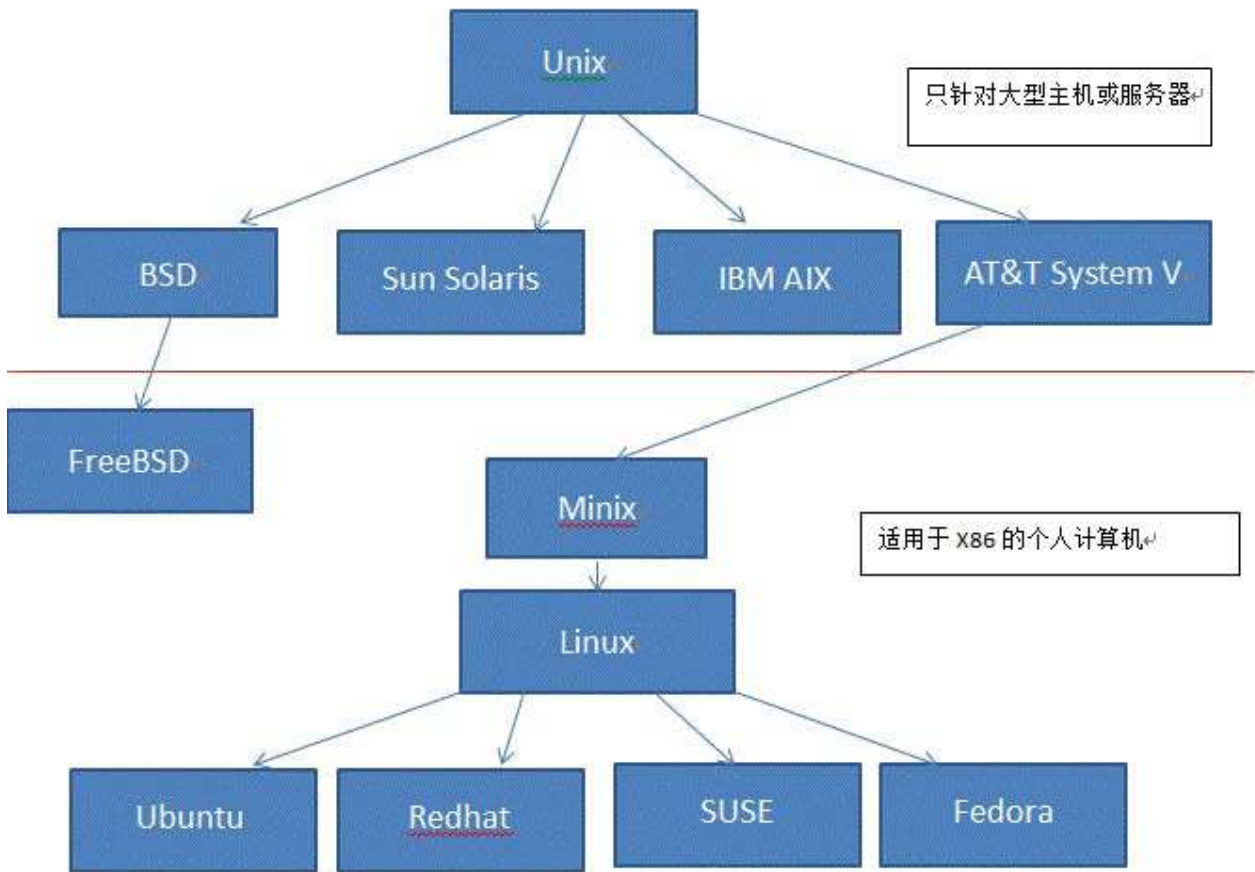
Linux

因为Minix只是教学使用，因此功能并不强，因此Torvalds利用GNU的bash当做开发环境，gcc当做编译工具，编写了Linux内核-v0.02，但是一开始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，因为Unix是遵循POSIX规范的，因此Torvalds修改了Linux，并遵循POSIX (Portable Operating System Interface，他规范了应用程序与内核的接口规范)；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；





操作系统的发展



一类是不知道自己在用Linux :



一类是知道自己在用Linux :





Minix没有火起来的原因

Minix的创始人说，MINIX 3没有统治世界是源于他在1992年犯下的一个错误，当时他认为BSD必然会一统天下，因为它是一个更稳定和更成熟的系统，其它操作系统难以与之竞争。因此他的MINIX的重心集中在教育上。四名BSD开发者已经成立了一家公司销售BSD系统，他们甚至还有一个有趣的电话号码1-800-ITS-UNIX。然而他们正因为这个电话号码而惹火上身。美国电话电报公司因电话号码而提起诉讼。官司打了三年才解决。在此期间，BSD陷于停滞，而Linux则借此一飞冲天。他的错误在于没有意识官司竟然持续了如此长的时间，以及BSD会因此受到削弱。如果美国电话电报公司没有起诉，Linux永远不会流行起来，BSD将统治世界。

Linux的不同版本以及应用领域

1.Linux内核及发行版介绍

<1>Linux内核版本

内核(kernel)是系统的核心，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。

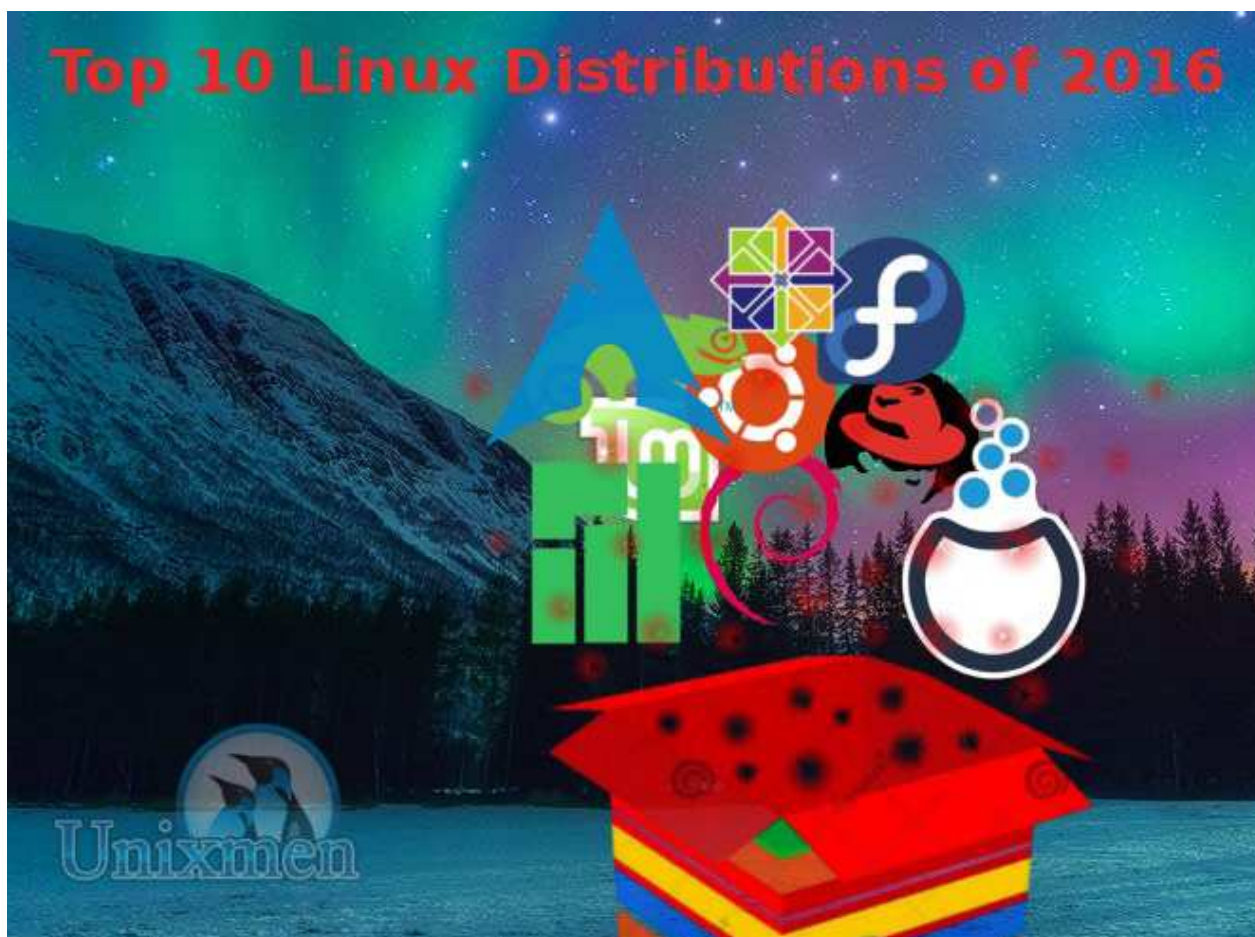
Linux内核版本又分为稳定版和开发版，两种版本是相互关联，相互循环：

- 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>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。



排名	2016	2015
1	Linux Mint	Linux Mint
2	Debian	Debian
3	Ubuntu	Ubuntu
4	openSUSE	openSUSE
5	redhat	Fedora
6	Fedora	Mageia
7	Manjaro	Manjaro
8	Mageia	CentOS
9	CentOS	Arch
10	Arch	Elementary

Fedora



Redhat



Ubuntu



2.应用领域

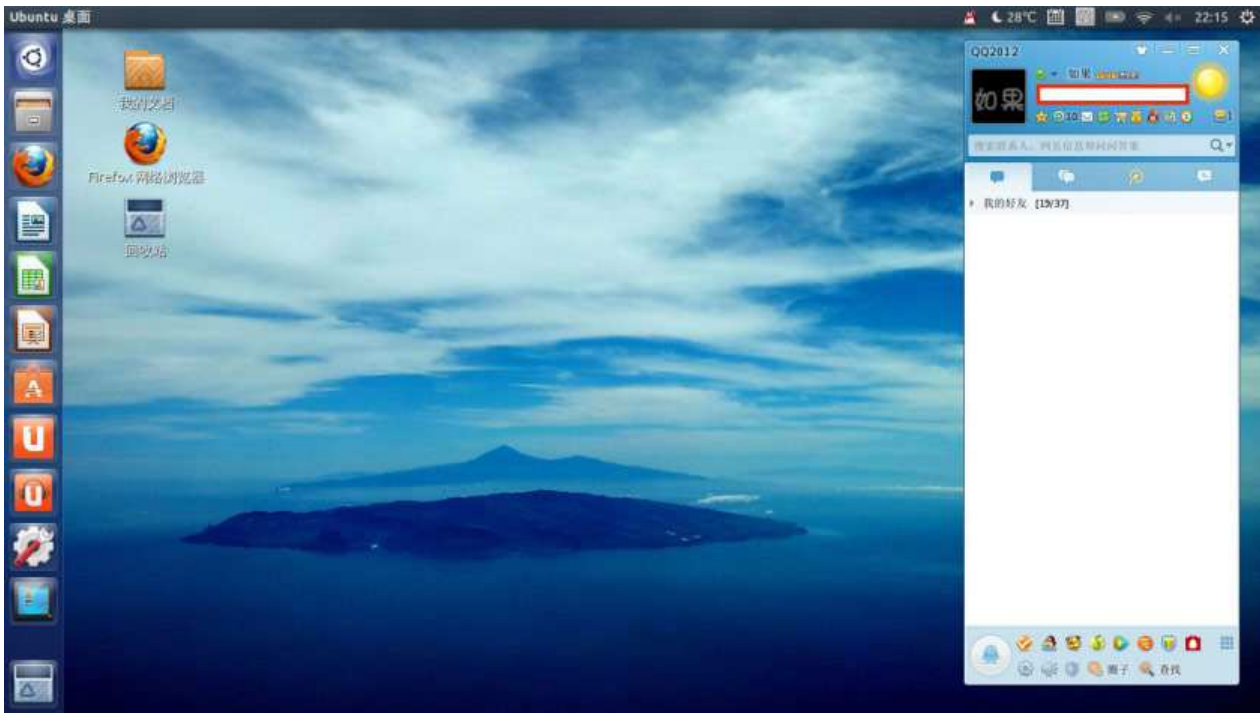
个人桌面领域的应用

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，但近些年来随着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

典型代表：ubuntu、fedora、suse linux

在Ubuntu中玩QQ





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，但早期因为维护、运行等原因同样受到了很大的限制，但近些年来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

典型代表：

- Red Hat公司的AS系列
- 完全开源的debian系列
- suse EnterPrise 11系列等

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，得到了摩托罗拉、三星、NEC、Google等公司的大力推广

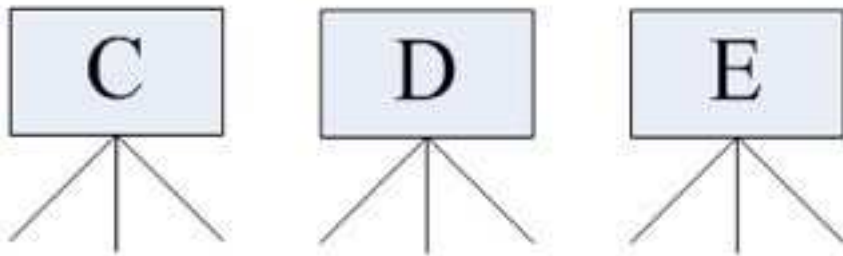
文件和目录

Windows和Linux文件系统区别

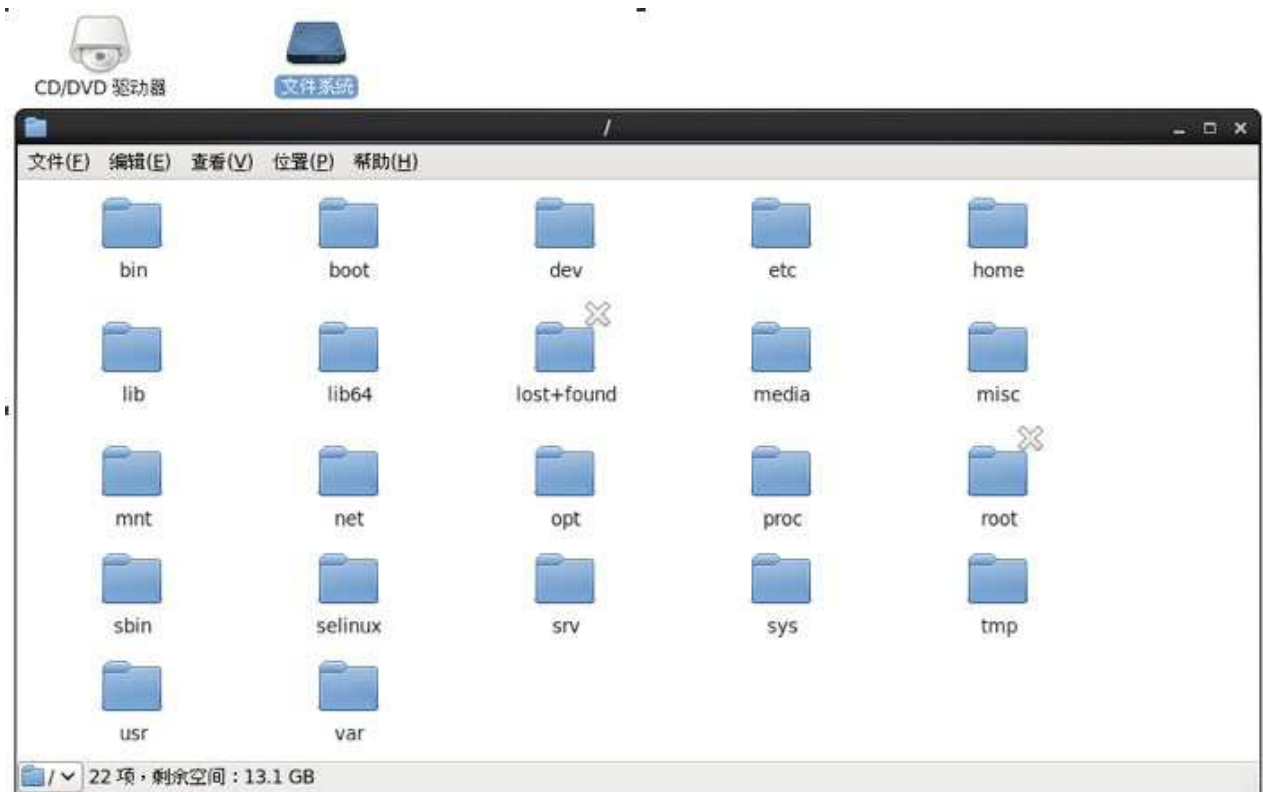
在 windows 平台下，打开“计算机”，我们看到的是一个一个的驱动器盘符：

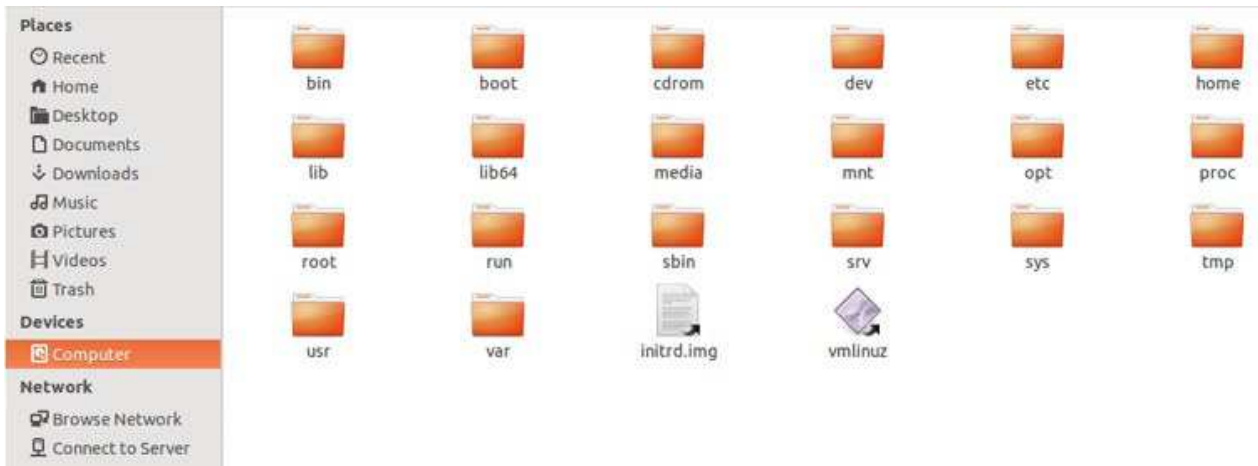


每个驱动器都有自己的根目录结构，这样形成了多个树并列的情形，如图所示：



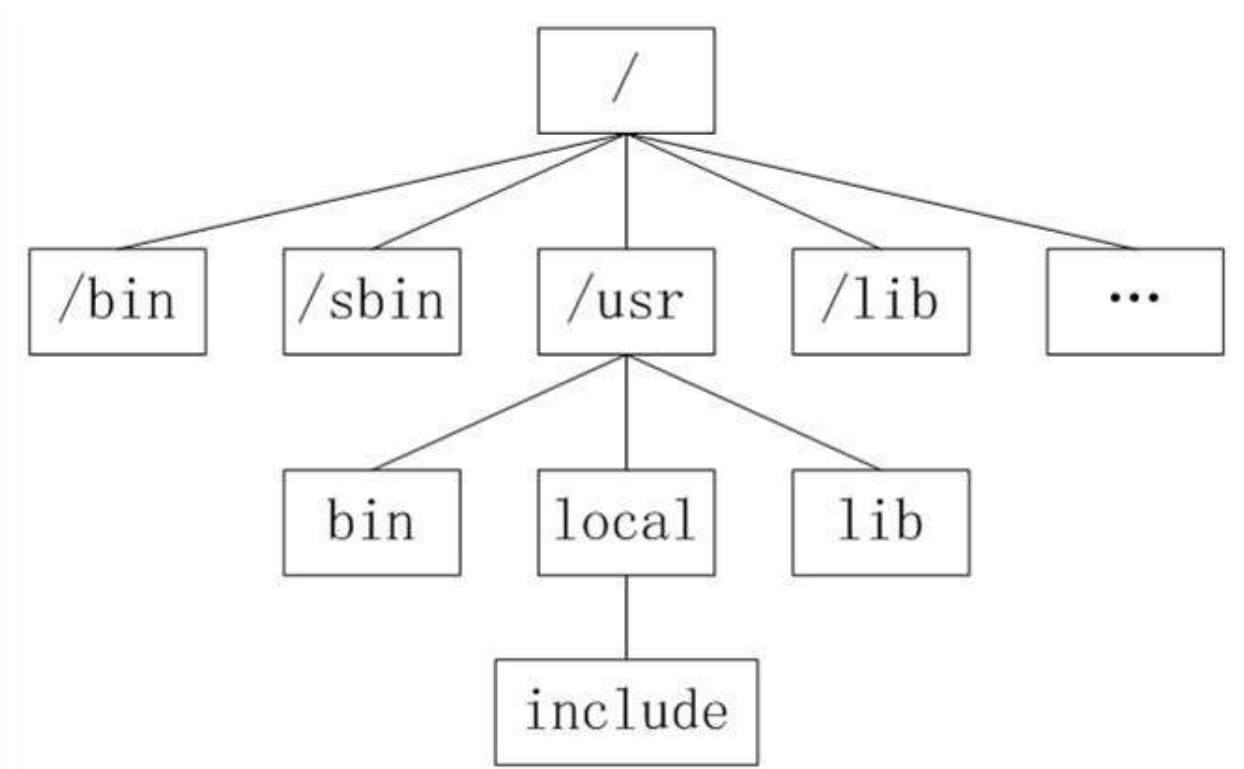
在 Linux 下，我们是看不到这些驱动器盘符，我们看到的是文件夹（目录）：





类Unix系统目录结构

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



Linux 目录

- /: 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。当你在终端里输入“/home”，你其实是在告诉电脑，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，比较重要的目录有 /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，如随时更改的日志文件 /var/log，/var/log/message：所有的登录文件存放目录，/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。

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  
~
```

相对路径和绝对路径

绝对路径

从/目录开始描述的路径为绝对路径，如：

```
cd /home
ls /usr
```

相对路径

从当前位置开始描述的路径为相对路径，如：

```
cd ../../
ls abc/def
```

.和..

每个目录下都有.和..

. 表示当前目录

.. 表示上一级目录，即父目录

根目录下的.和..都表示当前目录

文件权限

文件权限就是文件的访问控制权限，即哪些用户和组群可以访问文件以及可以执行什么样的操作。

Unix/Linux系统是一个典型的多用户系统，不同的用户处于不同的地位，对文件和目录有不同的访问权限。为了保护系统的安全性，Unix/Linux系统除了对用户权限作了严格的界定外，还在用户身份认证、访问控制、传输安全、文件读写权限等方面作了周密的控制。

在 Unix/Linux中的每一个文件或目录都包含有访问权限，这些访问权限决定了谁能访问和如何访问这些文件和目录。

访问用户

通过设定权限可以从以下三种访问方式限制访问权限：

- 只允许用户自己访问（所有者） 所有者就是创建文件的用户，用户是所有用户所创建文件的所有者，用户可以允许所在的用户组能访问用户的文件。
- 允许一个预先指定的用户组中的用户访问（用户组） 用户都组合成用户组，例如，某一类或某一项目中的所有用户都能够被系统管理员归为一个用户组，一个用户能够授予所在用户组的其他成员的文件访问权限。
- 允许系统中的任何用户访问（其他用户） 用户也将自己的文件向系统内的所有用户开放，在这种情况下，系统内的所有用户都能够访问用户的目录或文件。在这种意义上，系统内的其他所有用户就是 other 用户类

这有点类似于 QQ 空间的访问权限：





- 这个 QQ 空间是属于我的，我相当于管理者（也就是“所有者”），我想怎么访问就怎么访问。
- 同时，我可以设置允许 QQ 好友访问，而这些 QQ 好友则类似于“用户组”。
- 当然，我可以允许所有人访问，这里的所有人则类似于“其他用户”。

访问权限

用户能够控制一个给定的文件或目录的访问程度，一个文件或目录可能有读、写及执行权限：

- 读权限 (r) 对文件而言，具有读取文件内容的权限；对目录来说，具有浏览目录的权限。
- 写权限 (w) 对文件而言，具有新增、修改文件内容的权限；对目录来说，具有删除、移动目录内文件的权限。
- 可执行权限 (x) 对文件而言，具有执行文件的权限；对目录来说该用户具有进入目录的权限。

注意：通常，Unix/Linux系统只允许文件的属主(所有者)或超级用户改变文件的读写权限。

示例说明

```

root@ubuntu:~# ls -lh
总用量 56K
drwxrwxr-x 2 python python 4.0K 7月 30 15:02 dbs
drwxr-xr-x 4 python python 4.0K 12月 17 12:42 Desktop
drwxr-xr-x 2 python python 4.0K 5月 16 2016 Documents
drwxr-xr-x 4 python python 4.0K 10月 13 14:14 Downloads
-rw-rw-r-- 1 python python 18 7月 30 22:19 dump.rdb
-rw-r--r-- 1 python python 8.8K 5月 16 2016 examples.desktop
drwxr-xr-x 2 python python 4.0K 5月 16 2016 Music
drwxr-xr-x 2 python python 4.0K 5月 16 2016 Pictures
drwxr-xr-x 2 python python 4.0K 5月 16 2016 Public
drwxr-xr-x 2 python python 4.0K 5月 16 2016 Templates
drwxr-xr-x 2 python python 4.0K 5月 16 2016 Videos
drwxrwxr-x 2 python python 4.0K 6月 14 2016 workspace
root@ubuntu:~#

```

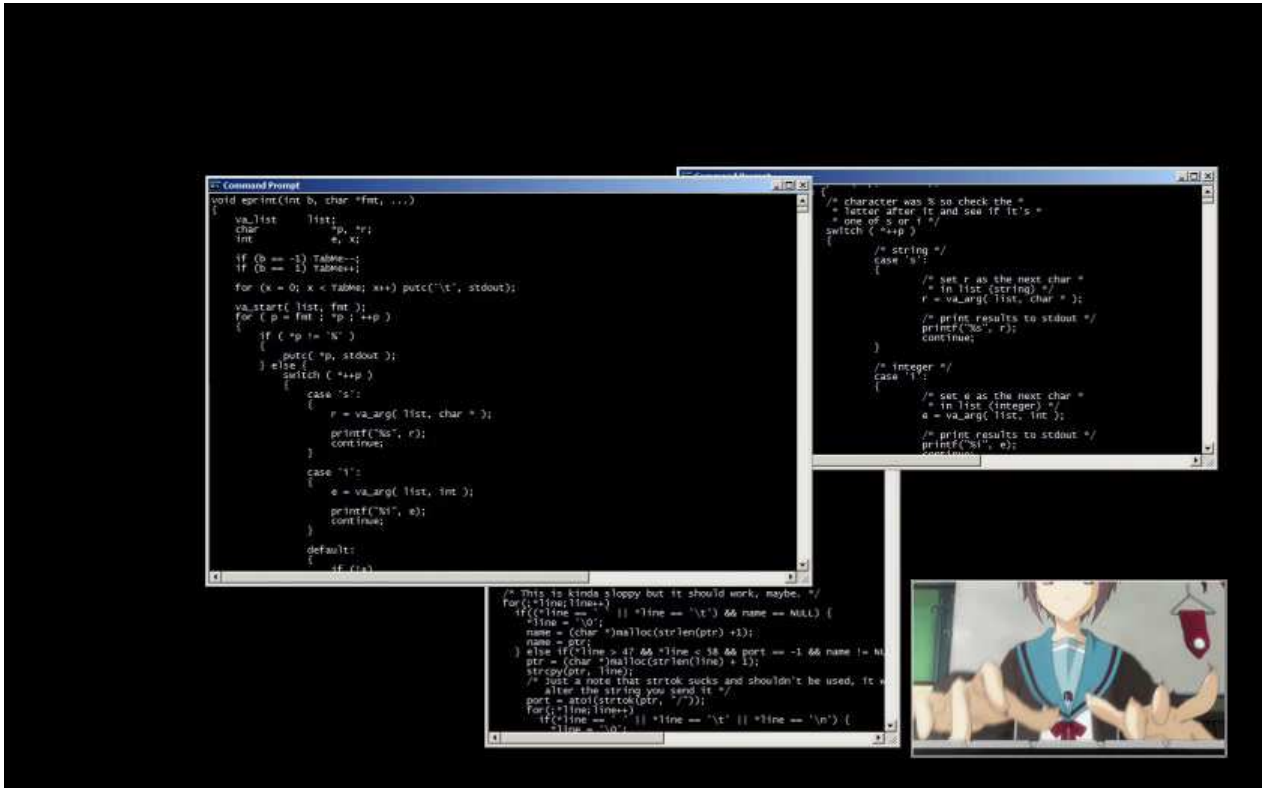
第1个字母代表文件的类型：“d”代表文件夹、“-”代表普通文件、“c”代表硬件字符设备、“b”代表硬件块设备、“s”表示管道文件、“l”代表软链接文件。后9个字母分别代表三组权限：文件所有者、用户者、其他用户拥有的权限。

每一个用户都有它自身的读、写和执行权限。

- 第一组权限控制访问自己的文件权限，即所有者权限。
- 第二组权限控制用户组访问其中一个用户的文件的权限。
- 第三组权限控制其他所有用户访问一个用户的文件的权限。

这三组权限赋予用户不同类型（即所有者、用户组和其他用户）的读、写及执行权限就构成了一个有9种类型的权限组。

常用基本命令



- 很多人可能在电视或电影中看到过类似的场景，黑客面对一个黑色的屏幕，上面飘着密密麻麻的字符，梆梆一顿敲，就完成了窃取资料的任务。

- Linux 刚出世时没有什么图形界面，所有的操作全靠命令完成，就如同电视里的黑客那样，充满了神秘与晦涩。
- 近几年来，尽管 Linux 发展得非常迅速，图形界面越来越友好，但是在真正的开发过程中，Linux 命令行的应用还是占有非常重要的席位，而且许多Linux功能在命令行界面要比图形化界面下运行的快。可以说不会命令行，就不算会 Linux。
- Linux 提供了大量的命令，利用它可以有效地完成大量的工作，如磁盘操作、文件存取、目录操作、进程管理、文件权限设定等。Linux 发行版本最少的命令也有 200 多个，这里只介绍比较重要和使用频率最多的命令。

1.命令使用方法

Linux命令格式:

```
command [-options] [parameter1] ...
```

说明:

- command: 命令名,相应功能的英文单词或单词的缩写 [-options]: 选项,可用来对命令进行控制,也可以省略, []代表可选 parameter1 ...: 传给命令的参数: 可以是零个一个或多个

例:

```
root@ubuntu:~# ls -a ./
.                .gnome2          .python_history
..               .gnome2_private .ssh
.bash_history    .gnupg           .stardict
.bash_logout    .ICEauthority    .sudo_as_admin_successful
.bashrc         .ipython         .sunpinyin
.cache          .local           Templates
.codeintel      .mongorc.js     Videos
.config         .mozilla         .vim
dbs             Music            .viminfo
.dbshell        .mysql_history  .vimrc
.dbus           .node_repl_history
Desktop         .npm             .wget-hsts
.dmrc          .pam_environment workspace
Documents      Pictures         .Xauthority
Downloads      .pki             .xinputrc
dump.rdb       .presage        .xsession-errors
examples.desktop
.profile       .psql_history   .xsession-errors.old
.gconf        .psql_history   Public
.gitconfig    Public
```

2.查看帮助文档

<1>--help

一般是linux命令自带的帮助信息

如：ls --help

<2>man(有问题找男人， manual)

man是linux提供的一个手册，包含了绝大部分的命令、函数使用说明

该手册分成很多章节（section），使用man时可以指定不同的章节来浏览。

例：man ls ; man 2 printf

man中各个section意义如下：

1. Standard commands（标准命令）
2. System calls（系统调用，如open,write）
3. Library functions（库函数，如printf,fopen）
4. Special devices（设备文件的说明，/dev下各种设备）
5. File formats（文件格式，如passwd）
6. Games and toys（游戏和娱乐）
7. Miscellaneous（杂项、惯例与协定等，例如Linux档案系统、网络协定、ASCII码；environ全局变量）
8. Administrative Commands（管理员命令，如ifconfig）

man是按照手册的章节号的顺序进行搜索的。

man设置了如下的功能键：

功能键	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man命令
h	列出所有功能键

/word	搜索word字符串
-------	-----------


```

LS(1)                                User Commands                                LS(1)
NAME
  ls - list directory contents 名字解释
SYNOPSIS
  ls [OPTION]... [FILE]... 使用规则
DESCRIPTION
  List information about the FILES (the current directory by
  default). Sort entries alphabetically if none of -cftuvSUX nor
  --sort is specified.

  Mandatory arguments to long options are mandatory for short
  options too. 用法的相
关描述
  -a, --all                do not ignore entries starting with .
  -A, --almost-all       do not list implied . and ..
  --author                 with -l, print the author of each file q 退出
  -b, --escape             print C-style escapes for nongraphic characters
  --block-size=SIZE
Manual page ls(1) line 1 (press h for help or q to quit)

```

注意：实际上，我们不用指定第几个章节也用查看，如，man ls

3.自动补全：

在敲出命令的前几个字母的同时，按下tab键，系统会自动帮我们补全命令

4.历史命令：

当系统执行过一些命令后，可按上下键翻看以前的命令，history将执行过的命令列举出来

Linux命令-文件、磁盘管理

1.文件管理

<1>查看文件信息：ls

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，它类似于DOS下的dir命令。

Linux文件或者目录名称最长可以有265个字符，“.”代表当前目录，“..”代表上一级目录，以“.”开头的文件为隐藏文件，需要用 -a 参数才能显示。

ls常用参数：

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


```
python@ubuntu:~/Desktop/01-Python基础班$ touch .haha.txt
python@ubuntu:~/Desktop/01-Python基础班$ ls
01-day      06-day      1.jpg      spider
02-day      07-day      2.jpg      test
03-DaFeiJi 07-dayfff   baidu.jpg  打飞机代码
03-day      08-day      beautifulsoup4-4.3.2 应用：打飞机
04-day      11-day      beautifulsoup4-4.3.2.tar.gz
05-day      12-day      daFeiji.py
python@ubuntu:~/Desktop/01-Python基础班$
python@ubuntu:~/Desktop/01-Python基础班$
python@ubuntu:~/Desktop/01-Python基础班$ ls -a
.          03-day      07-dayfff   2.jpg      .haha.txt
..         04-day      08-day      baidu.jpg  spider
01-day     05-day      11-day      beautifulsoup4-4.3.2  test
02-day     06-day      12-day      beautifulsoup4-4.3.2.tar.gz 打飞机代码
03-DaFeiJi 07-day      1.jpg      daFeiji.py 应用：打飞机
python@ubuntu:~/Desktop/01-Python基础班$
python@ubuntu:~/Desktop/01-Python基础班$ ls -l
总用量 356
drwxrwxr-x 2 python python 4096 10月 18 17:42 01-day
drwxrwxr-x 2 python python 4096 10月 18 17:24 02-day
drwxr-xr-x 3 python python 4096 10月 26 09:40 03-DaFeiJi
drwxrwxr-x 2 python python 4096 10月 19 17:13 03-day
drwxrwxr-x 2 python python 4096 10月 22 17:16 04-day
drwxrwxr-x 3 python python 4096 10月 23 11:16 05-day
drwxrwxr-x 2 python python 4096 10月 24 17:25 06-day
drwxrwxr-x 2 python python 4096 10月 25 19:08 07-day
drwxrwxr-x 2 python python 4096 10月 25 11:34 07-dayfff
drwxrwxr-x 3 python python 4096 10月 26 14:01 08-day
drwxrwxr-x 7 python python 4096 10月 28 16:36 11-day
drwxrwxr-x 2 python python 4096 10月 29 09:37 12-day
-rw-rw-r-- 1 python python 7105 10月 28 21:39 1.jpg
-rw-rw-r-- 1 python python 58598 10月 29 10:24 2.jpg
-rw-rw-r-- 1 python python 79763 10月 29 09:52 baidu.jpg
drwxrwxr-x 6 python python 4096 10月 28 22:15 beautifulsoup4-4.3.2
-rwxrwxrwx 1 python python 143356 8月 23 19:13 beautifulsoup4-4.3.2.tar.g
z
-rw-rw-r-- 1 python python 0 10月 25 21:04 daFeiji.py
drwxrwxr-x 2 python python 4096 10月 28 22:04 spider
drwxrwxr-x 3 python python 4096 10月 26 18:58 test
drwxr-xr-x 3 python python 4096 10月 26 08:39 打飞机代码
drwxr-xr-x 3 python python 4096 10月 30 15:24 应用：打飞机
python@ubuntu:~/Desktop/01-Python基础班$
python@ubuntu:~/Desktop/01-Python基础班$ ls -h
01-day      06-day      1.jpg      spider
02-day      07-day      2.jpg      test
03-DaFeiJi 07-dayfff   baidu.jpg  打飞机代码
03-day      08-day      beautifulsoup4-4.3.2 应用：打飞机
04-day      11-day      beautifulsoup4-4.3.2.tar.gz
05-day      12-day      daFeiji.py
python@ubuntu:~/Desktop/01-Python基础班$
python@ubuntu:~/Desktop/01-Python基础班$
python@ubuntu:~/Desktop/01-Python基础班$ ls -l -h
总用量 356K
drwxrwxr-x 2 python python 4.0K 10月 18 17:42 01-day
drwxrwxr-x 2 python python 4.0K 10月 18 17:24 02-day
drwxr-xr-x 3 python python 4.0K 10月 26 09:40 03-DaFeiJi
drwxrwxr-x 2 python python 4.0K 10月 19 17:13 03-day
drwxrwxr-x 2 python python 4.0K 10月 22 17:16 04-day
```

文件大小
单位是字节

需要和选项配合

```

drwxrwxr-x 3 python python 4.0K 10月 23 11:16 05-day
drwxrwxr-x 2 python python 4.0K 10月 24 17:25 06-day
drwxrwxr-x 2 python python 4.0K 10月 25 19:08 07-day
drwxrwxr-x 2 python python 4.0K 10月 25 11:34 07-dayfff
drwxrwxr-x 3 python python 4.0K 10月 26 14:01 08-day
drwxrwxr-x 7 python python 4.0K 10月 28 16:36 11-day
drwxrwxr-x 2 python python 4.0K 10月 29 09:37 12-day
-rw-rw-r-- 1 python python 7.0K 10月 28 21:39 1.jpg
-rw-rw-r-- 1 python python 58K 10月 29 10:24 2.jpg
-rw-rw-r-- 1 python python 78K 10月 29 09:52 baidu.jpg
python@ubuntu:~/Desktop/01-Python基础班$ ls -lh
总用量 356K
drwxrwxr-x 2 python python 4.0K 10月 18 17:42 01-day
drwxrwxr-x 2 python python 4.0K 10月 18 17:24 02-day
drwxr-xr-x 3 python python 4.0K 10月 26 09:40 03-DaFeiJi
drwxrwxr-x 2 python python 4.0K 10月 19 17:13 03-day
drwxrwxr-x 2 python python 4.0K 10月 22 17:16 04-day
drwxrwxr-x 3 python python 4.0K 10月 23 11:16 05-day
drwxrwxr-x 2 python python 4.0K 10月 24 17:25 06-day
drwxrwxr-x 2 python python 4.0K 10月 25 19:08 07-day
drwxrwxr-x 2 python python 4.0K 10月 25 11:34 07-dayfff
drwxrwxr-x 3 python python 4.0K 10月 26 14:01 08-day
drwxrwxr-x 7 python python 4.0K 10月 28 16:36 11-day
drwxrwxr-x 2 python python 4.0K 10月 29 09:37 12-day
-rw-rw-r-- 1 python python 7.0K 10月 28 21:39 1.jpg
-rw-rw-r-- 1 python python 58K 10月 29 10:24 2.jpg
-rw-rw-r-- 1 python python 78K 10月 29 09:52 baidu.jpg
drwxrwxr-x 6 python python 4.0K 10月 28 22:15 beautifulsoup4-4.3.2
-rwxrwxrwx 1 python python 140K 8月 23 19:13 beautifulsoup4-4.3.2.tar.gz
-rw-rw-r-- 1 python python 0 10月 25 21:04 daFeiji.py
drwxrwxr-x 2 python python 4.0K 10月 28 22:04 spider
    
```

多个选项可以合并到一起写，没有先后顺序，即-lh和-hl都可以

图中列出的信息含义如下图所示：



与DOS下的文件操作类似，在Unix/Linux系统中，也同样允许使用特殊字符来同时引用多个文件名，这些特殊字符被称为通配符。

通配符	含义
*	文件代表文件名中所有字符
ls te*	查找以te开头的文件
ls *html	查找结尾为html的文件
?	代表文件名中任意一个字符

<code>ls ?.c</code>	只找第一个字符任意，后缀为.c的文件
<code>ls a.?</code>	只找只有3个字符，前2字符为a.，最后一个字符任意的文件
<code>[]</code>	[]和[]将字符组括起来，表示可以匹配字符组中的任意一个。“-”用于表示字符范围。
<code>[abc]</code>	匹配a、b、c中的任意一个
<code>[a-f]</code>	匹配从a到f范围内的的任意一个字符
<code>ls [a-f]*</code>	找到从a到f范围内的的任意一个字符开头的文件
<code>ls a-f</code>	查找文件名为a-f的文件,当“-”处于方括号之外失去通配符的作用
<code>\</code>	如果要使通配符作为普通字符使用，可以在其前面加上转义字符。“?”和“*”处于方括号内时不用使用转义字符就失去通配符的作用。
<code>ls *a</code>	查找文件名为*a的文件

<2>输出重定向命令：>

Linux允许将命令执行结果重定向到一个文件，本应显示在终端上的内容保存到指定文件中。

如：`ls > test.txt` (`test.txt` 如果不存在，则创建，存在则覆盖其内容)

```
python@ubuntu:~/Desktop/01-Python基础班$ ls
01-day      06-day      1.jpg      spider
02-day      07-day      2.jpg      test
03-DaFeiJi 07-dayfff  baidu.jpg  打飞机代码
03-day      08-day      beautifulsoup4-4.3.2 应用：打飞机
04-day      11-day      beautifulsoup4-4.3.2.tar.gz
05-day      12-day      daFeiji.py
python@ubuntu:~/Desktop/01-Python基础班$ ls > 重定向的文件.txt
python@ubuntu:~/Desktop/01-Python基础班$ ls
01-day      06-day      1.jpg      spider
02-day      07-day      2.jpg      test
03-DaFeiJi 07-dayfff  baidu.jpg  打飞机代码
03-day      08-day      beautifulsoup4-4.3.2 应用：打飞机
04-day      11-day      beautifulsoup4-4.3.2.tar.gz 重定向的文件.txt
05-dav     12-dav     daFeiiv.py
python@ubuntu:~/Desktop/01-Python基础班$ cat 重定向的文件.txt
01-day
02-day
03-DaFeiJi
03-day
04-day
05-day
06-day
07-day
07-dayfff
08-day
11-day
12-day
1.jpg
2.jpg
baidu.jpg
beautifulsoup4-4.3.2
beautifulsoup4-4.3.2.tar.gz
daFeiji.py
spider
test
打飞机代码
应用：打飞机
重定向的文件.txt
```

ls本来显示的内容都已经存储到大于号后面的文件中

可以使用cat查看文件的内容

总结：所谓重定向就是修改了默认的输出方向

注意：>输出重定向会覆盖原来的内容，>>输出重定向则会追加到文件的尾部。

<3>分屏显示：more

查看内容时，在信息过长无法在一屏上显示时，会出现快速滚屏，使得用户无法看清文件的内容，此时可以使用more命令，每次只显示一页，按下空格键可以显示下一页，按下q键退出显示，按下h键可以获取帮助。

```
python@ubuntu:~/Desktop/test$ more feiji.py
#coding=utf-8
'''
    12.飞机大战，基础功能成型版本
    升级内容：1. 从玩家飞机类、敌机类、子弹类中抽象了父类方法
'''
import pygame
#导入按键的检测
from pygame.locals import *
import time
import random

#定义基础类
class Base(object):
    def __init__(self, x, y, planeImageName):
        self.x = x
        self.y = y

        #选择一个图片'
        self.image = pygame.image.load(planeImageName).convert()

    def draw(self):
        screen.blit(self.image, (self.x, self.y))

#定义飞机类
class Plane(Base):
```

<4>管道：|

管道：一个命令的输出可以通过管道做为另一个命令的输入。

管道我们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)。

```
python@ubuntu:~/Desktop/01-Python基础班/应用：打飞机/feiji$ ls -lh | more
总用量 1.7M
-rwxr-xr-x 1 root root 36K 10月 26 10:04 background.png
-rwxr-xr-x 1 root root 418K 10月 26 10:04 bg.png
-rwxr-xr-x 1 root root 2.6K 10月 26 10:04 bomb-1.gif
-rwxr-xr-x 1 root root 2.1K 10月 26 10:04 bomb-2.gif
-rwxr-xr-x 1 root root 5.5K 10月 26 10:04 bomb.png
-rwxr-xr-x 1 root root 8.8K 10月 26 10:04 btn_finish.png
-rwxr-xr-x 1 root root 122 10月 26 10:04 bullet-1.gif
-rwxr-xr-x 1 root root 490 10月 26 10:04 bullet1.png
-rwxr-xr-x 1 root root 151 10月 26 10:04 bullet-2.gif
-rwxr-xr-x 1 root root 498 10月 26 10:04 bullet2.png
-rwxr-xr-x 1 root root 401 10月 26 10:04 bullet-3.gif
-rwxr-xr-x 1 root root 23K 10月 26 10:04 bullet.png
-rwxr-xr-x 1 root root 29K 10月 26 10:04 button_nor.png
-rwxr-xr-x 1 root root 29K 10月 26 10:04 button_p.png
-rwxr-xr-x 1 root root 3.4K 10月 26 10:04 enemy0_down1.png
-rwxr-xr-x 1 root root 3.8K 10月 26 10:04 enemy0_down2.png
```

<5>清屏：clear

clear作用为清除终端上的显示(类似于DOS的cls清屏功能)，也可使用快捷键：Ctrl + l (“l”为字母)。

<6>切换工作目录：cd

在使用Unix/Linux的时候，经常需要更换工作目录。cd命令可以帮助用户切换工作目录。Linux所有的目录和文件名大小写敏感

cd后面可跟绝对路径，也可以跟相对路径。如果省略目录，则默认切换到当前用户的主目录。

命令	含义
cd	切换到当前用户的主目录(/home/用户目录)，用户登陆的时候，默认的目录就是用户的主目录。
cd ~	切换到当前用户的主目录(/home/用户目录)
cd .	切换到当前目录
cd ..	切换到上级目录
cd -	可进入上次所在的目录

```

python@ubuntu:~/Desktop/01-Python基础班/应用：打飞机/feiji$ pwd
/home/python/Desktop/01-Python基础班/应用：打飞机/feiji
python@ubuntu:~/Desktop/01-Python基础班/应用：打飞机/feiji$
python@ubuntu:~/Desktop/01-Python基础班/应用：打飞机/feiji$
python@ubuntu:~/Desktop/01-Python基础班/应用：打飞机/feiji$ cd ..
python@ubuntu:~/Desktop/01-Python基础班/应用：打飞机$
python@ubuntu:~/Desktop/01-Python基础班/应用：打飞机$
python@ubuntu:~/Desktop/01-Python基础班/应用：打飞机$ pwd
/home/python/Desktop/01-Python基础班/应用：打飞机
python@ubuntu:~/Desktop/01-Python基础班/应用：打飞机$ cd .
python@ubuntu:~/Desktop/01-Python基础班/应用：打飞机$ pwd
/home/python/Desktop/01-Python基础班/应用：打飞机
python@ubuntu:~/Desktop/01-Python基础班/应用：打飞机$ cd ..
python@ubuntu:~/Desktop/01-Python基础班$ pwd
/home/python/Desktop/01-Python基础班
python@ubuntu:~/Desktop/01-Python基础班$ ls
01-day      06-day      1.jpg      spider
02-day      07-day      2.jpg      test
03-DaFeiJi 07-dayfff   baidu.jpg  打飞机代码
03-day      08-day      beautifulsoup4-4.3.2 应用：打飞机
04-day      11-day     beautifulsoup4-4.3.2.tar.gz 重定向的文件.txt
05-day      12-day     daFeiji.py
python@ubuntu:~/Desktop/01-Python基础班$ cd test
python@ubuntu:~/Desktop/01-Python基础班/test$ pwd
/home/python/Desktop/01-Python基础班/test
python@ubuntu:~/Desktop/01-Python基础班/test$ cd /home
python@ubuntu:/home$ pwd
/home
python@ubuntu:/home$ cd /etc
python@ubuntu:/etc$ pwd
/etc
python@ubuntu:/etc$

```

注意：

- 如果路径是从根路径开始的，则路径的前面需要加上“/”，如“/mnt”，通常进入某个目录里的文件夹，前面不用加“/”。

<7>显示当前路径：pwd

使用pwd命令可以显示当前的工作目录，该命令很简单，直接输入pwd即可，后面不带参数。

```

python@ubuntu:/etc$ pwd
/etc
python@ubuntu:/etc$ cd /home
python@ubuntu:/home$ pwd
/home
python@ubuntu:/home$

```

<8>创建目录：mkdir

通过mkdir命令可以创建一个新的目录。参数-p可递归创建目录。

需要注意的是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，并且目录创建者必须对当前目录具有写权限。

```
python@ubuntu:~/Desktop/test$ pwd
/home/python/Desktop/test
python@ubuntu:~/Desktop/test$ ls
1
python@ubuntu:~/Desktop/test$ mkdir haha
python@ubuntu:~/Desktop/test$ ls
1 haha
python@ubuntu:~/Desktop/test$ mkdir a/b/c/d
mkdir: 无法创建目录"a/b/c/d": 没有那个文件或目录
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ mkdir a/b/c/d -p
python@ubuntu:~/Desktop/test$ ls
1 a haha
python@ubuntu:~/Desktop/test$ tree
.
├── 1
├── a
│   ├── b
│   │   ├── c
│   │   └── d
└── haha
```

在当前路径下
创建haha文
件夹

在当前路径下
递归创建文件
夹

以目录树的方
式显示

6 directories, 0 files
python@ubuntu:~/Desktop/test\$

<9>删除目录：rmdir

可使用rmdir命令删除一个目录。必须离开目录，并且目录必须为空目录，不然提示删除失败。

<10>删除文件：rm

可通过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。为了防止文件误删，可以在rm后使用-i参数以逐个确认要删除的文件。

常用参数及含义如下表所示：

参数	含义
-i	以进行交互式方式执行
-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


```

python@ubuntu:~/Desktop/test$ ls
1 a haha
python@ubuntu:~/Desktop/test$ touch haha.txt
python@ubuntu:~/Desktop/test$ ls
1 a haha haha.txt
python@ubuntu:~/Desktop/test$ rm haha.txt -i 会询问是否真的要删除
rm: 是否删除普通空文件 'haha.txt'? y
python@ubuntu:~/Desktop/test$ ls
1 a haha
python@ubuntu:~/Desktop/test$ touch haha.txt
python@ubuntu:~/Desktop/test$ ls
1 a haha haha.txt
python@ubuntu:~/Desktop/test$ rm haha.txt 不询问
python@ubuntu:~/Desktop/test$ ls
1 a haha
python@ubuntu:~/Desktop/test$

python@ubuntu:~/Desktop/test$ ls
1 a haha
python@ubuntu:~/Desktop/test$ rm 1
rm: 无法删除'1': 是一个目录
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ rm 1 -r
python@ubuntu:~/Desktop/test$ ls
a haha
python@ubuntu:~/Desktop/test$

```

<11>建立链接文件：ln

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。

```
python@ubuntu:~/Desktop/test$ ls
a haha
python@ubuntu:~/Desktop/test$ touch haha.txt
python@ubuntu:~/Desktop/test$ ls
a haha haha.txt
python@ubuntu:~/Desktop/test$ ln haha.txt haha_hardlink.txt 硬链接
python@ubuntu:~/Desktop/test$ ls -ln
总用量 8.0K
drwxrwxr-x 3 python python 4.0K 12月 19 10:19 a
drwxrwxr-x 2 python python 4.0K 12月 19 10:19 haha
-rw-rw-r-- 2 python python 0 12月 19 10:27 haha_hardlink.txt
-rw-rw-r-- 2 python python 0 12月 19 10:27 haha.txt
python@ubuntu:~/Desktop/test$ ls
a haha haha_hardlink.txt haha.txt
python@ubuntu:~/Desktop/test$ ln -s haha.txt haha_softlink.txt 软链接
python@ubuntu:~/Desktop/test$ ls -ln
总用量 8.0K
drwxrwxr-x 3 python python 4.0K 12月 19 10:19 a
drwxrwxr-x 2 python python 4.0K 12月 19 10:19 haha
-rw-rw-r-- 2 python python 0 12月 19 10:27 haha_hardlink.txt
lrwxrwxrwx 1 python python 8 12月 19 10:30 haha_softlink.txt -> haha.txt
-rw-rw-r-- 2 python python 0 12月 19 10:27 haha.txt
python@ubuntu:~/Desktop/test$ █
python@ubuntu:~/Desktop/test$ ls
a haha haha_hardlink.txt haha_softlink.txt haha.txt
python@ubuntu:~/Desktop/test$ gedit haha.txt 编辑文件
```




```
python@ubuntu:~/Desktop/test$ ls
a haha haha_hardlink.txt haha_softlink.txt haha.txt
python@ubuntu:~/Desktop/test$ gedit haha.txt
python@ubuntu:~/Desktop/test$ cat haha.txt
haha,wo shi dongge,hhh
python@ubuntu:~/Desktop/test$ cat haha_hardlink.txt
haha,wo shi dongge,hhh
python@ubuntu:~/Desktop/test$ cat haha_softlink.txt
haha,wo shi dongge,hhh
python@ubuntu:~/Desktop/test$

python@ubuntu:~/Desktop/test$ ls
a haha haha_hardlink.txt haha_softlink.txt haha.txt
python@ubuntu:~/Desktop/test$ ls -lh
总用量 16K
drwxrwxr-x 3 python python 4.0K 12月 19 10:19 a
drwxrwxr-x 2 python python 4.0K 12月 19 10:19 haha
-rw-rw-r-- 2 python python 24 12月 19 10:34 haha_hardlink.txt
lrwxrwxrwx 1 python python 8 12月 19 10:30 haha_softlink.txt -> haha.txt
-rw-rw-r-- 2 python python 24 12月 19 10:34 haha.txt
python@ubuntu:~/Desktop/test$ rm haha_softlink.txt
python@ubuntu:~/Desktop/test$ ls -lh
总用量 16K
drwxrwxr-x 3 python python 4.0K 12月 19 10:19 a
drwxrwxr-x 2 python python 4.0K 12月 19 10:19 haha
-rw-rw-r-- 2 python python 24 12月 19 10:34 haha_hardlink.txt
-rw-rw-r-- 2 python python 24 12月 19 10:34 haha.txt
python@ubuntu:~/Desktop/test$ cat haha.txt
haha,wo shi dongge,hhh
python@ubuntu:~/Desktop/test$ cat haha_hardlink.txt
haha,wo shi dongge,hhh
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ ls -lh
总用量 16K
drwxrwxr-x 3 python python 4.0K 12月 19 10:19 a
drwxrwxr-x 2 python python 4.0K 12月 19 10:19 haha
-rw-rw-r-- 2 python python 24 12月 19 10:34 haha_hardlink.txt
-rw-rw-r-- 2 python python 24 12月 19 10:34 haha.txt
python@ubuntu:~/Desktop/test$ gedit haha_hardlink.txt
```

可以看到文件的内容是一样的

删除软连接就好比删除了快捷方式，不会影响源文件

修改文件，任意添加点数据



```
python@ubuntu:~/Desktop/test$ ls -lh
总用量 16K
drwxrwxr-x 3 python python 4.0K 12月 19 10:19 a
drwxrwxr-x 2 python python 4.0K 12月 19 10:19 haha
-rw-rw-r-- 2 python python 24 12月 19 10:34 haha_hardlink.txt
-rw-rw-r-- 2 python python 24 12月 19 10:34 haha.txt
python@ubuntu:~/Desktop/test$ gedit haha_hardlink.txt
python@ubuntu:~/Desktop/test$ cat haha_hardlink.txt
haha,wo shi dongge,hhh

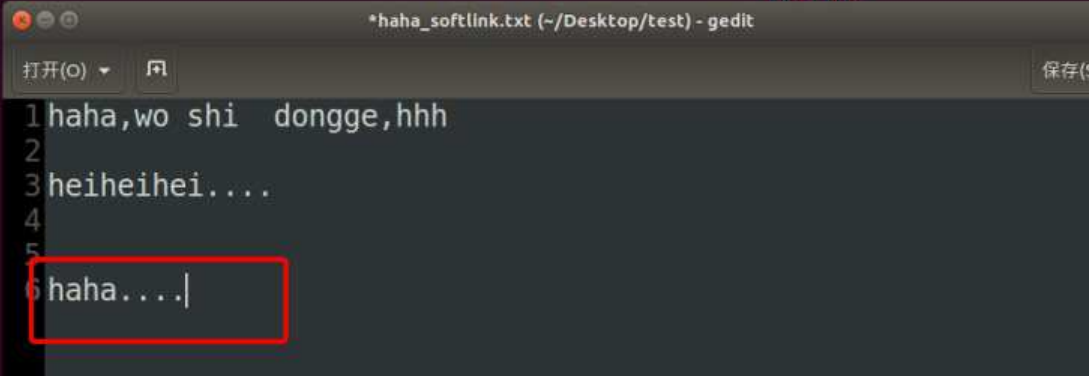
heiheihei....
python@ubuntu:~/Desktop/test$ cat haha.txt
haha,wo shi dongge,hhh

heiheihei....
python@ubuntu:~/Desktop/test$
```

通过硬链接可以修改源文件


```
python@ubuntu:~/Desktop/test$ ls -lh
总用量 16K
drwxrwxr-x 3 python python 4.0K 12月 19 10:19 a
drwxrwxr-x 2 python python 4.0K 12月 19 10:19 haha
-rw-rw-r-- 2 python python 39 12月 19 10:41 haha_hardlink.txt
lrwxrwxrwx 1 python python 8 12月 19 10:42 haha_softlink.txt -> haha.txt
-rw-rw-r-- 2 python python 39 12月 19 10:41 haha.txt
python@ubuntu:~/Desktop/test$ gedit haha_softlink.txt
```

通过软链接修改文件

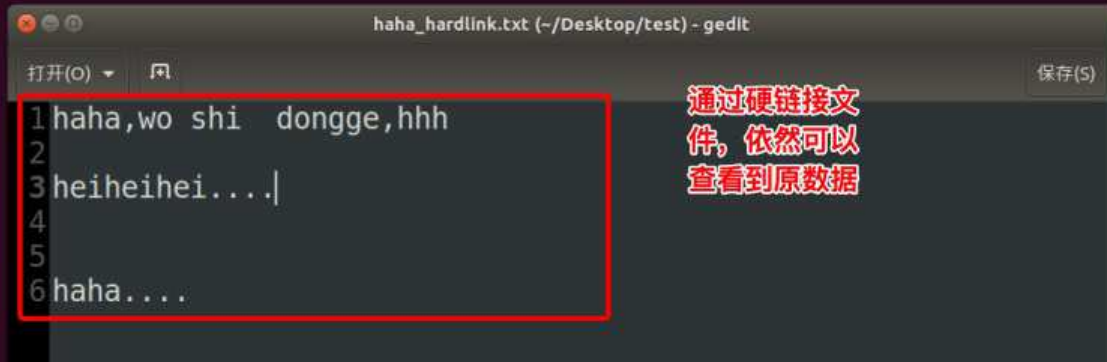


```
python@ubuntu:~/Desktop/test$ ls
a haha haha_hardlink.txt haha_softlink.txt haha.txt
python@ubuntu:~/Desktop/test$ rm haha.txt
python@ubuntu:~/Desktop/test$ ls -lh
总用量 12K
drwxrwxr-x 3 python python 4.0K 12月 19 10:19 a
drwxrwxr-x 2 python python 4.0K 12月 19 10:19 haha
-rw-rw-r-- 1 python python 50 12月 19 10:43 haha_hardlink.txt
lrwxrwxrwx 1 python python 8 12月 19 10:42 haha_softlink.txt -> haha.txt
python@ubuntu:~/Desktop/test$ gedit haha_softlink.txt
```

删除源文件后,再打开软连接后,看不到数据



```
python@ubuntu:~/Desktop/test$ ls -lh
总用量 12K
drwxrwxr-x 3 python python 4.0K 12月 19 10:19 a
drwxrwxr-x 2 python python 4.0K 12月 19 10:19 haha
-rw-rw-r-- 1 python python 50 12月 19 10:43 haha_hardlink.txt
lrwxrwxrwx 1 python python 8 12月 19 10:42 haha_softlink.txt -> haha.txt
python@ubuntu:~/Desktop/test$ gedit haha_softlink.txt
python@ubuntu:~/Desktop/test$ gedit haha_hardlink.txt
```



```
python@ubuntu:~/Desktop/test$ ls -lh
总用量 12K
drwxrwxr-x 3 python python 4.0K 12月 19 10:19 a
drwxrwxr-x 2 python python 4.0K 12月 19 10:19 haha
-rw-rw-r-- 1 python python 50 12月 19 10:43 haha_hardlink.txt
lrwxrwxrwx 1 python python 8 12月 19 10:42 haha_softlink.txt -> haha.txt
python@ubuntu:~/Desktop/test$ ln haha_hardlink.txt haha_hardlink_2.txt
python@ubuntu:~/Desktop/test$ ls -lh
总用量 16K
drwxrwxr-x 3 python python 4.0K 12月 19 10:19 a
drwxrwxr-x 2 python python 4.0K 12月 19 10:19 haha
-rw-rw-r-- 2 python python 50 12月 19 10:43 haha_hardlink_2.txt
-rw-rw-r-- 2 python python 50 12月 19 10:43 haha_hardlink.txt
lrwxrwxrwx 1 python python 8 12月 19 10:42 haha_softlink.txt -> haha.txt
python@ubuntu:~/Desktop/test$ ln haha_hardlink.txt haha_hardlink_3.txt
python@ubuntu:~/Desktop/test$ ls -lh
总用量 20K
drwxrwxr-x 3 python python 4.0K 12月 19 10:19 a
drwxrwxr-x 2 python python 4.0K 12月 19 10:19 haha
-rw-rw-r-- 3 python python 50 12月 19 10:43 haha_hardlink_2.txt
-rw-rw-r-- 3 python python 50 12月 19 10:43 haha_hardlink_3.txt
-rw-rw-r-- 3 python python 50 12月 19 10:43 haha_hardlink.txt
lrwxrwxrwx 1 python python 8 12月 19 10:42 haha_softlink.txt -> haha.txt
python@ubuntu:~/Desktop/test$
```

```
python@ubuntu:~/Desktop/test$ cat haha_hardlink.txt
haha,wo shi dongge,hhh

heiheihei....

haha....
python@ubuntu:~/Desktop/test$ cat haha_hardlink_2.txt
haha,wo shi dongge,hhh


heiheihei....

haha....
python@ubuntu:~/Desktop/test$ cat haha_hardlink_3.txt
haha,wo shi dongge,hhh

heiheihei....

haha....
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ gedit haha_hardlink.txt
```

通过硬链接创建的文件内容相同



修改一个文件内容 (随意修改)

```
python@ubuntu:~/Desktop/test$ gedit haha_hardlink.txt
python@ubuntu:~/Desktop/test$ cat haha_hardlink.txt
-----1-----
python@ubuntu:~/Desktop/test$ cat haha_hardlink_2.txt
-----1-----
python@ubuntu:~/Desktop/test$ cat haha_hardlink_3.txt
-----1-----
python@ubuntu:~/Desktop/test$ ls -lh
总用量 20K
drwxrwxr-x 3 python python 4.0K 12月 19 10:19 a
drwxrwxr-x 2 python python 4.0K 12月 19 10:19 haha
-rw-rw-r-- 3 python python 21 12月 19 10:53 haha_hardlink_2.txt
-rw-rw-r-- 3 python python 21 12月 19 10:53 haha_hardlink_3.txt
-rw-rw-r-- 3 python python 21 12月 19 10:53 haha_hardlink.txt
lrwxrwxrwx 1 python python 8 12月 19 10:42 haha_softlink.txt -> haha.txt
python@ubuntu:~/Desktop/test$
```

修改一次,但是多个硬链接文件内容都变了

文件内容

名字1
名字2
名字3

硬链接是同一个文件内容,但是有多个文件名

这个文件所拥有的名字数量

<12>查看或者合并文件内容: cat

```

python@ubuntu:~/Desktop/test$ ls
a      haha_hardlink_2.txt  haha_hardlink.txt
haha   haha_hardlink_3.txt  haha_softlink.txt
python@ubuntu:~/Desktop/test$ cat haha_hardlink.txt
-----1-----
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ cat /etc/samba/smb.conf
#
# Sample configuration file for the Samba suite for Debian GNU/Linux.
#
#
# This is the main Samba configuration file. You should read the
# smb.conf(5) manual page in order to understand the options listed
# here. Samba has a huge number of configurable options most of which
# are not shown in this example
#
# Some options that are often worth tuning have been included as
# commented-out examples in this file.
python@ubuntu:~/Desktop/test$ ls
test2.txt  test.txt
python@ubuntu:~/Desktop/test$ cat test.txt
-----1-1-1--1-1-1---
python@ubuntu:~/Desktop/test$ cat test2.txt
-2-2--2----2-2-2-2--
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ cat test.txt test2.txt > heihei.txt
python@ubuntu:~/Desktop/test$ ls
heihei.txt test2.txt test.txt
python@ubuntu:~/Desktop/test$ cat heihei.txt
-----1-1-1--1-1-1---
-2-2--2----2-2-2-2--
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ cat test2.txt test.txt > heihei.txt
python@ubuntu:~/Desktop/test$ cat heihei.txt
-2-2--2----2-2-2-2--
-----1-1-1--1-1-1---
python@ubuntu:~/Desktop/test$ cat test2.txt test.txt >> heihei.txt
python@ubuntu:~/Desktop/test$ cat heihei.txt
-2-2--2----2-2-2-2--
-----1-1-1--1-1-1---
-2-2--2----2-2-2-2--
-----1-1-1--1-1-1---
python@ubuntu:~/Desktop/test$

```

查看当前路径下的文件内容

查看绝对路径下的文件内容

把这两个文件的内容都重定向到 heihei.txt 文件中

>表示:
先清空, 后添加

>>表示:
直接添加

<13>文本搜索：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。如果找到匹配模式，grep打印包含模式的所有行。

grep一般格式为：

```
grep [-选项] '搜索内容串' 文件名
```

在grep命令中输入字符串参数时，最好引号或双引号括起来。例如：`grep'a '1.txt`。

常用选项说明：

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

grep搜索内容串可以是正则表达式。

正则表达式是对字符串操作的一种逻辑公式，就是用事先定义好的一些特定字符、及这些特定字符的组合，组成一个“规则字符串”，这个“规则字符串”用来表达对字符串的一种过滤逻辑。

grep常用正则表达式：

参数	含义
<code>^a</code>	行首,搜寻以 m 开头的行； <code>grep -n '^a' 1.txt</code>
<code>ke\$</code>	行尾,搜寻以 ke 结束的行； <code>grep -n 'ke\$' 1.txt</code>
<code>[Ss]igna[Ll]</code>	匹配 [] 里中一系列字符中的一个；搜寻匹配单词signal、signal、Signal、SignalL的行； <code>grep -n '[Ss]igna[Ll]' 1.txt</code>
<code>.</code>	(点)匹配一个非换行符的字符；匹配 e 和 e 之间有任何一个字符，可以匹配 eee, eae, eve, 但是不匹配 ee, eaae； <code>grep -n 'e.e' 1.txt</code>


```
python@ubuntu:~/Desktop/test/dongge/test$ cat test.txt
abcdefg
Abcdefg
----1----m
----2----n
----3----m
ahelloworld
nihaome
python
Python
txt
tnt
tpt

python@ubuntu:~/Desktop/test/dongge/test$ grep -n '^a' test.txt
1:abcdefg
6:ahelloworld
python@ubuntu:~/Desktop/test/dongge/test$ grep -n 'm$' test.txt
3:----1----m
5:----3----m
python@ubuntu:~/Desktop/test/dongge/test$ grep -n 't[xn]t' test.txt
10:txt
11:tnt
python@ubuntu:~/Desktop/test/dongge/test$ grep -n 't.t' test.txt
10:txt
11:tnt
12:tpt
python@ubuntu:~/Desktop/test/dongge/test$
```

<14>查找文件：find

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，也可以用来搜索特定用户属主的文件。

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

<15>拷贝文件：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。

常用选项说明：

选项	含义
-a	该选项通常在复制目录时使用，它保留链接、文件属性，并递归地复制目录，简单而言，保持文件原有属性。
-f	已经存在的目标文件而不提示
-i	交互式复制，在覆盖目标文件之前将给出提示要求用户确认
-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。
-v	显示拷贝进度

```
python@ubuntu:~/Desktop/test$ tree
.
├── haha_hardlink_2.txt
├── haha_hardlink_3.txt
├── haha_softlink.txt -> haha.txt
├── python_haha.txt
├── test1
│   ├── haha
│   │   ├── a
│   │   │   ├── b
│   │   │   │   ├── c
│   │   │   │   └── d
│   └── test2
└── test2

7 directories, 4 files
python@ubuntu:~/Desktop/test$ cp test1/haha test2 -ivr
'test1/haha' -> 'test2/haha'
'test1/haha/a' -> 'test2/haha/a'
'test1/haha/a/b' -> 'test2/haha/a/b'
'test1/haha/a/b/c' -> 'test2/haha/a/b/c'
'test1/haha/a/b/c/d' -> 'test2/haha/a/b/c/d'
python@ubuntu:~/Desktop/test$ tree
.
├── haha_hardlink_2.txt
├── haha_hardlink_3.txt
├── haha_softlink.txt -> haha.txt
├── python_haha.txt
├── test1
│   ├── haha
│   │   ├── a
│   │   │   ├── b
│   │   │   │   ├── c
│   │   │   │   └── d
│   └── test2
├── test2
│   ├── haha
│   │   ├── a
│   │   │   ├── b
│   │   │   │   ├── c
│   │   │   │   └── d
└── test2

12 directories, 4 files
```

<16>移动文件：mv

用户可以使用mv命令来移动文件或目录，也可以给文件或目录重命名。

常用选项说明：

选项	含义
-f	禁止交互式操作，如有覆盖也不会给出提示
-i	确认交互方式操作，如果mv操作将导致对已存在的目标文件的覆盖，系统会询问是否重写，要求用户回答以避免误覆盖文件

-v 显示移动进度

```
python@ubuntu:~/Desktop/test$ ls
a      haha_hardlink_2.txt  haha_hardlink.txt
haha   haha_hardlink_3.txt  haha_softlink.txt
python@ubuntu:~/Desktop/test$ mv haha_hardlink.txt  python_haha.txt
python@ubuntu:~/Desktop/test$ ls
a      haha_hardlink_2.txt  haha_softlink.txt
haha   haha_hardlink_3.txt  python_haha.txt
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ ls
haha   haha_hardlink_3.txt  python_haha.txt
haha_hardlink_2.txt  haha_softlink.txt
python@ubuntu:~/Desktop/test$ mkdir test1
python@ubuntu:~/Desktop/test$ mkdir test2
python@ubuntu:~/Desktop/test$ ls
haha   haha_hardlink_3.txt  python_haha.txt  test2
haha_hardlink_2.txt  haha_softlink.txt  test1
python@ubuntu:~/Desktop/test$ mv haha test1
python@ubuntu:~/Desktop/test$ ls
haha_hardlink_2.txt  haha_softlink.txt  test1
haha_hardlink_3.txt  python_haha.txt   test2
python@ubuntu:~/Desktop/test$ tree
.
├── haha_hardlink_2.txt
├── haha_hardlink_3.txt
├── haha_softlink.txt -> haha.txt
├── python_haha.txt
├── test1
│   ├── haha
│   │   ├── a
│   │   │   ├── b
│   │   │   │   ├── c
│   │   │   │   └── d
│   └── test2
└── test2
```

修改文件的名字

将haha文件夹移动到test1文件夹下

<17>归档管理：tar

计算机中的数据经常需要备份，tar是Unix/Linux中最常用的备份工具，此命令可以把一系列文件归档到一个大文件中，也可以把档案文件解开以恢复数据。

tar使用格式 tar [参数] 打包文件名 文件

tar命令很特殊，其参数前面可以使用“-”，也可以不使用。

常用参数：

参数	含义
-c	生成档案文件，创建打包文件

-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
-t	列出档案中包含的文件
-x	解开档案文件

注意：除了f需要放在参数的最后，其它参数的顺序任意。

```

python@ubuntu:~/Desktop/test$ ls
haha_hardlink_2.txt  haha_softlink.txt  test1
haha_hardlink_3.txt  python_haha.txt    test2
python@ubuntu:~/Desktop/test$ tar -cvf test.tar *
haha_hardlink_2.txt
haha_hardlink_3.txt
haha_softlink.txt
python_haha.txt
test1/
test1/haha/
test1/haha/a/
test1/haha/a/b/
test1/haha/a/b/c/
test1/haha/a/b/c/d/
test2/
test2/haha/
test2/haha/a/
test2/haha/a/b/
test2/haha/a/b/c/
test2/haha/a/b/c/d/
python@ubuntu:~/Desktop/test$ ls
haha_hardlink_2.txt  haha_softlink.txt  test1  test.tar
haha_hardlink_3.txt  python_haha.txt    test2
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ ls
haha_hardlink_2.txt  haha_softlink.txt  test1  test.tar
haha_hardlink_3.txt  python_haha.txt    test2
python@ubuntu:~/Desktop/test$ rm *.txt
python@ubuntu:~/Desktop/test$ ls
test1  test2  test.tar
python@ubuntu:~/Desktop/test$ rm *[12]
rm: 无法删除'test1': 是一个目录
rm: 无法删除'test2': 是一个目录
python@ubuntu:~/Desktop/test$ rm *[12] -r
python@ubuntu:~/Desktop/test$ ls
test.tar
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ tar -xvf test.tar
haha_hardlink_2.txt
haha_hardlink_3.txt
haha_softlink.txt
python_haha.txt
test1/
test1/haha/
test1/haha/a/
test1/haha/a/b/
test1/haha/a/b/c/
test1/haha/a/b/c/d/
python@ubuntu:~/Desktop/test$ ls
haha_hardlink_2.txt  haha_softlink.txt  test1  test.tar
haha_hardlink_3.txt  python_haha.txt    test2

```

<18>文件压缩解压：gzip

tar与gzip命令结合使用实现文件打包、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xxxx.tar.gz。

gzip使用格式如下：

```
gzip [选项] 被压缩文件
```

常用选项：

选项	含义
-d	解压
-r	压缩所有子目录

```
python@ubuntu:~/Desktop/test$ ls
test.tar
python@ubuntu:~/Desktop/test$ ls -lh
总用量 12K
-rw-rw-r-- 1 python python 10K 12月 19 11:20 test.tar
python@ubuntu:~/Desktop/test$ gzip -r test.tar test.tar.gz
python@ubuntu:~/Desktop/test$ ls -lh
总用量 4.0K
-rw-rw-r-- 1 python python 392 12月 19 11:20 test.tar.gz
python@ubuntu:~/Desktop/test$ 占用空间小了
                             很多

python@ubuntu:~/Desktop/test$ ls
test.tar.gz
python@ubuntu:~/Desktop/test$ gzip -d test.tar.gz
python@ubuntu:~/Desktop/test$ ls
test.tar
python@ubuntu:~/Desktop/test$ 解压缩

python@ubuntu:~/Desktop/test$ gzip test.tar
python@ubuntu:~/Desktop/test$ ls
test.tar.gz
python@ubuntu:~/Desktop/test$ 也是压缩
```

tar这个命令并没有压缩的功能，它只是一个打包的命令，但是在tar命令中增加一个选项(-z)可以调用gzip实现了一个压缩的功能，实行一个先打包后压缩的过程。

压缩用法：tar cvzf 压缩包包名 文件1 文件2 ...

```
-z : 指定压缩包的格式为：file.tar.gz
```

```
python@ubuntu:~/Desktop/test$ ls
haha_hardlink_2.txt  haha_softlink.txt  test1
haha_hardlink_3.txt  python_haha.txt    test2
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ tar -zcvf test.tar.gz *
haha_hardlink_2.txt
haha_hardlink_3.txt
haha_softlink.txt
python_haha.txt
test1/
test1/haha/
test1/haha/a/
test1/haha/a/b/
test1/haha/a/b/c/
test1/haha/a/b/c/d/
test2/
test2/haha/
test2/haha/a/
test2/haha/a/b/
test2/haha/a/b/c/
test2/haha/a/b/c/d/
python@ubuntu:~/Desktop/test$ ls
haha_hardlink_2.txt  haha_softlink.txt  test1  test.tar.gz
haha_hardlink_3.txt  python_haha.txt    test2
python@ubuntu:~/Desktop/test$ rm *.txt *[12] -r
python@ubuntu:~/Desktop/test$ ls
test.tar.gz
python@ubuntu:~/Desktop/test$
```

将当前路径下所有的文件以及文件夹打包并压缩到 test.tar.gz 文件中

解压用法： tar zxvf 压缩包包名

-z:指定压缩包的格式为: file.tar.gz

```
python@ubuntu:~/Desktop/test$ ls
test.tar.gz
python@ubuntu:~/Desktop/test$ tar -zxvf test.tar.gz
haha_hardlink_2.txt
haha_hardlink_3.txt
haha_softlink.txt
python_haha.txt
test1/
test1/haha/
test1/haha/a/
test1/haha/a/b/
test1/haha/a/b/c/
test1/haha/a/b/c/d/
test2/
test2/haha/
test2/haha/a/
test2/haha/a/b/
test2/haha/a/b/c/
test2/haha/a/b/c/d/
python@ubuntu:~/Desktop/test$ ls
haha_hardlink_2.txt  haha_softlink.txt  test1  test.tar.gz
haha_hardlink_3.txt  python_haha.txt    test2
python@ubuntu:~/Desktop/test$
```

解压到当前路径下

解压到指定目录: -C (大写字母“C”)

```
python@ubuntu:~/Desktop/test$ ls
test.tar.gz
python@ubuntu:~/Desktop/test$ mkdir dongge
python@ubuntu:~/Desktop/test$ ls
dongge test.tar.gz
python@ubuntu:~/Desktop/test$ tar -zxvf test.tar.gz -C dongge/
haha_hardlink_2.txt
haha_hardlink_3.txt
haha_softlink.txt
python_haha.txt
test1/
test1/haha/
test1/haha/a/
test1/haha/a/b/
test1/haha/a/b/c/
test1/haha/a/b/c/d/
test2/
test2/haha/
test2/haha/a/
test2/haha/a/b/
test2/haha/a/b/c/
test2/haha/a/b/c/d/
python@ubuntu:~/Desktop/test$ ls
dongge test.tar.gz
python@ubuntu:~/Desktop/test$ ls dongge/
haha_hardlink_2.txt haha_softlink.txt test1
haha_hardlink_3.txt python_haha.txt test2
python@ubuntu:~/Desktop/test$
```

解压到指定路径

<19>文件压缩解压: bzip2

tar与bzip2命令结合使用实现文件打包、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xxxx.tar.gz2。

在tar命令中增加一个选项(-j)可以调用bzip2实现了一个压缩的功能,实行一个先打包后压缩的过程。

压缩用法: tar -jcvf 压缩包包名 文件...(tar jcvf bk.tar.bz2 *.c)

解压用法: tar -jxvf 压缩包包名 (tar jxvf bk.tar.bz2)

<20>文件压缩解压: zip、unzip

通过zip压缩文件的目标文件不需要指定扩展名,默认扩展名为zip。

压缩文件: zip [-r] 目标文件(没有扩展名) 源文件

解压文件: unzip -d 解压后目录文件 压缩文件

```
python@ubuntu:~/Desktop/test/dongge$ ls
haha_hardlink_2.txt  haha_softlink.txt  test1
haha_hardlink_3.txt  python_haha.txt    test2
python@ubuntu:~/Desktop/test/dongge$ zip myzip *
zip warning: name not matched: haha_softlink.txt
adding: haha_hardlink_2.txt (deflated 52%)
adding: haha_hardlink_3.txt (deflated 52%)
adding: python_haha.txt (deflated 52%)
adding: test1/ (stored 0%)
adding: test2/ (stored 0%)
python@ubuntu:~/Desktop/test/dongge$ ls
haha_hardlink_2.txt  haha_softlink.txt  python_haha.txt  test2
haha_hardlink_3.txt  myzip.zip         test1
python@ubuntu:~/Desktop/test/dongge$ rm *.txt *[12] -r
python@ubuntu:~/Desktop/test/dongge$ ls
myzip.zip
python@ubuntu:~/Desktop/test/dongge$ 
python@ubuntu:~/Desktop/test/dongge$ ls
myzip.zip
python@ubuntu:~/Desktop/test/dongge$ unzip -d ./test myzip.zip
Archive:  myzip.zip
  inflating: ./test/haha_hardlink_2.txt
  inflating: ./test/haha_hardlink_3.txt
  inflating: ./test/python_haha.txt
   creating: ./test/test1/
   creating: ./test/test2/
python@ubuntu:~/Desktop/test/dongge$ ls
myzip.zip  test
python@ubuntu:~/Desktop/test/dongge$ ls test/
haha hardlink 2.txt  haha hardlink 3.txt  python haha.txt  test1  test2
python@ubuntu:~/Desktop/test/dongge$
```

压缩所有文件到myzip文件中

解压到test文件夹

少了一个文件原因是：需要的那个软链接需要的原文件不存在

<21>查看命令位置： which

```
python@ubuntu:~/Desktop/test$ which ls
/bin/ls
python@ubuntu:~/Desktop/test$ which LSS
python@ubuntu:~/Desktop/test$
```

如果找到就显示这个命令的路径

Linux命令-系统管理

<1>查看当前日历： cal

cal命令用于查看当前日历，-y显示整年日历：

```
root@ubuntu:~# cal
      十二月 2016
日 一 二 三 四 五 六
           1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31

root@ubuntu:~# █
```

<2>显示或设置时间： date

设置时间格式（需要管理员权限）：

```
date [MMDDhhmm[[CC]YY][.ss]] +format
```

CC为年前两位yy为年的后两位，前两位的mm为月，后两位的mm为分钟，dd为天，hh为小时，ss为秒。如： date 010203042016.55。

显示时间格式（date '+%y,%m,%d,%H,%M,%S'）：

format格式	含义
%Y, %y	年
%m	月
%d	日

%H	时
%M	分
%S	秒

```
python@ubuntu:/home$ date
2016年 12月 18日 星期日 22:27:14 CST
python@ubuntu:/home$
python@ubuntu:/home$ date '+%Y/%m/%d'
2016/12/18
python@ubuntu:/home$
python@ubuntu:/home$ date '+%Y=%m=%d'
2016=12=18
python@ubuntu:/home$
```

<3>查看进程信息：ps

进程是一个具有一定独立功能的程序，它是操作系统动态执行的基本单元。

ps命令可以查看进程的详细状况，常用选项(选项可以不加“-”)如下：

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
-w	显示加宽，以便显示更多的信息
-r	只显示正在运行的进程

```
python@ubuntu:/home$ ps -a
  PID TTY          TIME CMD
 7966 pts/4    00:00:00 ps
python@ubuntu:/home$
python@ubuntu:/home$
python@ubuntu:/home$ ps -aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1   0.0   0.4 119948 4364 ?        Ss   19:20   0:02 /sbin/in
root         2   0.0   0.0     0     0 ?        S    19:20   0:00 [kthread
root         3   0.0   0.0     0     0 ?        S    19:20   0:00 [ksoftir
root         5   0.0   0.0     0     0 ?        S<   19:20   0:00 [kworker
root         7   0.0   0.0     0     0 ?        R    19:20   0:04 [rcu_sch
```

<4>动态显示进程：top

top命令用来动态显示运行中的进程。top命令能够在运行后，在指定的时间间隔更新显示信息。可以在使用top命令时加上-d 来指定显示信息更新的时间间隔。

在top命令执行后，可以按下按键得到对显示的结果进行排序：

按键	含义
M	根据内存使用量来排序
P	根据CPU占有率来排序
T	根据进程运行时间的长短来排序
U	可以根据后面输入的用户名来筛选进程
K	可以根据后面输入的PID来杀死进程。
q	退出
h	获得帮助

```
Tasks: 285 total, 1 running, 284 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1.0 us, 0.7 sy, 0.0 ni, 98.1 id, 0.2 wa, 0.0 hi, 0.0 si,
KiB Mem : 998348 total, 161132 free, 497624 used, 339592 buff/c
KiB Swap: 4192252 total, 3571820 free, 620432 used. 308292 avail
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+
1	root	20	0	119948	4364	2648	S	0.0	0.4	0:02.79
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00
3	root	20	0	0	0	0	S	0.0	0.0	0:00.34
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00
7	root	20	0	0	0	0	S	0.0	0.0	0:04.87
8	root	20	0	0	0	0	S	0.0	0.0	0:00.00
9	root	rt	0	0	0	0	S	0.0	0.0	0:00.00
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.08
11	root	20	0	0	0	0	S	0.0	0.0	0:00.00
12	root	0	-20	0	0	0	S	0.0	0.0	0:00.00
13	root	0	-20	0	0	0	S	0.0	0.0	0:00.00
14	root	20	0	0	0	0	S	0.0	0.0	0:00.07
15	root	0	-20	0	0	0	S	0.0	0.0	0:00.00
16	root	25	5	0	0	0	S	0.0	0.0	0:00.00
17	root	39	19	0	0	0	S	0.0	0.0	0:02.48
18	root	0	-20	0	0	0	S	0.0	0.0	0:00.00
19	root	0	-20	0	0	0	S	0.0	0.0	0:00.00
20	root	0	-20	0	0	0	S	0.0	0.0	0:00.00

<5>终止进程：kill

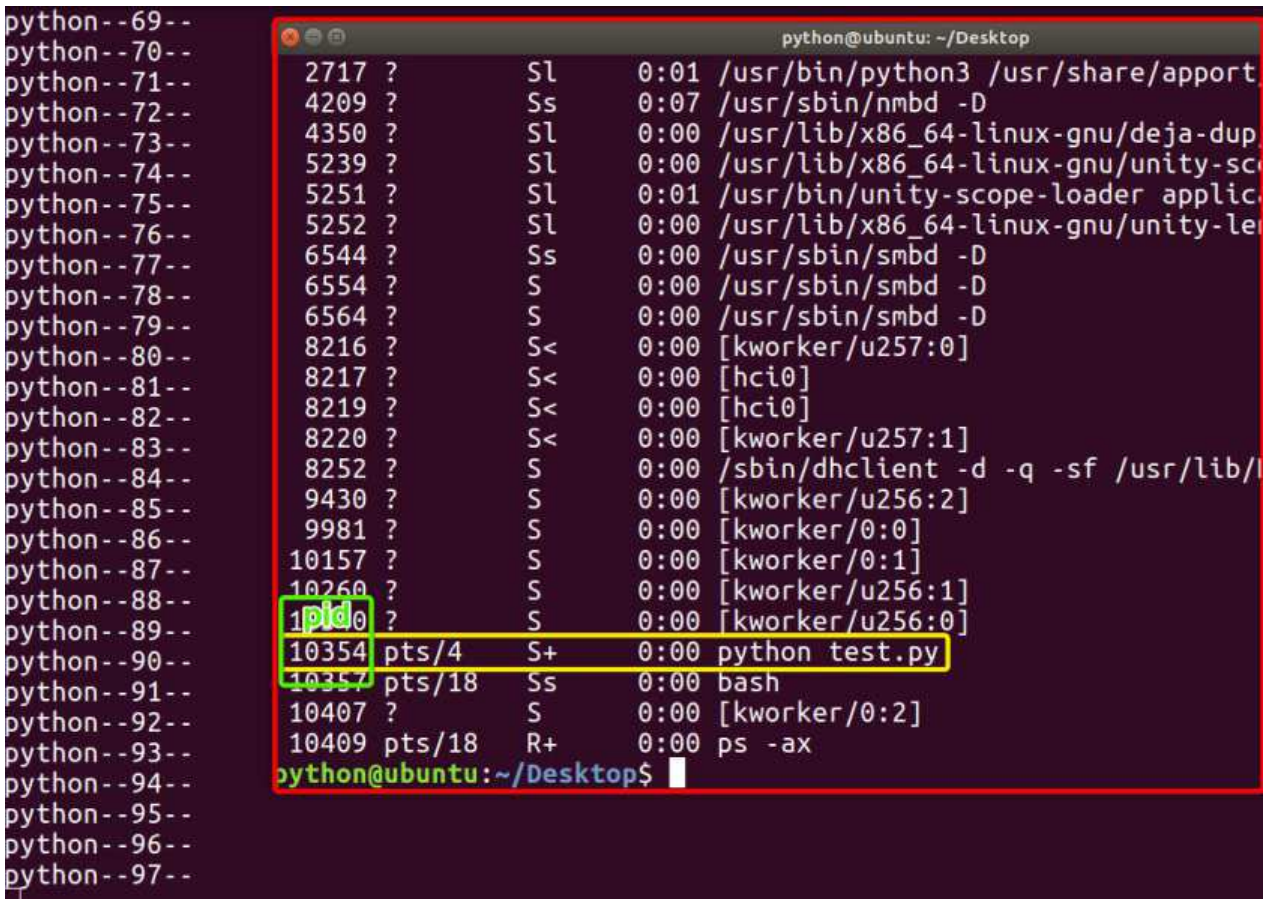
kill命令指定进程号的进程，需要配合 ps 使用。

使用格式：

```
kill [-signal] pid
```

信号值从0到15，其中9为绝对终止，可以处理一般信号无法终止的进程。

kill 9133：9133 为应用程序所对应的进程号



```
python--69--
python--70--
python--71--
python--72--
python--73--
python--74--
python--75--
python--76--
python--77--
python--78--
python--79--
python--80--
python--81--
python--82--
python--83--
python--84--
python--85--
python--86--
python--87--
python--88--
python--89--
python--90--
python--91--
python--92--
python--93--
python--94--
python--95--
python--96--
python--97--

python@ubuntu: ~/Desktop
2717 ?      Sl      0:01 /usr/bin/python3 /usr/share/apport
4209 ?      Ss      0:07 /usr/sbin/nmbd -D
4350 ?      Sl      0:00 /usr/lib/x86_64-linux-gnu/deja-dup
5239 ?      Sl      0:00 /usr/lib/x86_64-linux-gnu/unity-sc
5251 ?      Sl      0:01 /usr/bin/unity-scope-loader applic
5252 ?      Sl      0:00 /usr/lib/x86_64-linux-gnu/unity-le
6544 ?      Ss      0:00 /usr/sbin/smbd -D
6554 ?      S       0:00 /usr/sbin/smbd -D
6564 ?      S       0:00 /usr/sbin/smbd -D
8216 ?      S<      0:00 [kworker/u257:0]
8217 ?      S<      0:00 [hci0]
8219 ?      S<      0:00 [hci0]
8220 ?      S<      0:00 [kworker/u257:1]
8252 ?      S       0:00 /sbin/dhclient -d -q -sf /usr/lib/l
9430 ?      S       0:00 [kworker/u256:2]
9981 ?      S       0:00 [kworker/0:0]
10157 ?     S       0:00 [kworker/0:1]
10260 ?     S       0:00 [kworker/u256:1]
10354 pts/4    S+      0:00 python test.py
10357 pts/18   Ss      0:00 bash
10407 ?      S       0:00 [kworker/0:2]
10409 pts/18   R+      0:00 ps -ax
python@ubuntu:~/Desktop$
```

```
python--144--
python--145--
python--146--      4209 ?      Ss      0:07 /usr/sbin/nmbd -D
python--147--      4350 ?      Sl      0:00 /usr/lib/x86_64-linux-gnu/deja-du
python--148--      5239 ?      Sl      0:00 /usr/lib/x86_64-linux-gnu/unity-s
python--149--      5251 ?      Sl      0:01 /usr/bin/unity-scope-loader appli
python--150--      5252 ?      Sl      0:00 /usr/lib/x86_64-linux-gnu/unity-l
python--151--      6544 ?      Ss      0:00 /usr/sbin/smbd -D
python--152--      6554 ?      S       0:00 /usr/sbin/smbd -D
python--153--      6564 ?      S       0:00 /usr/sbin/smbd -D
python--154--      8216 ?      S<      0:00 [kworker/u257:0]
python--155--      8217 ?      S<      0:00 [hci0]
python--156--      8219 ?      S<      0:00 [hci0]
python--157--      8220 ?      S<      0:00 [kworker/u257:1]
python--158--      8252 ?      S       0:00 /sbin/dhclient -d -q -sf /usr/lib
python--159--      9430 ?      S       0:00 [kworker/u256:2]
python--160--      9981 ?      S       0:00 [kworker/0:0]
python--161--     10157 ?      S       0:00 [kworker/0:1]
python--162--     10260 ?      S       0:00 [kworker/u256:1]
python--163--     10340 ?      S       0:00 [kworker/u256:0]
python--164--     10354 pts/4      S+      0:00 python test.py
python--165--     10357 pts/18     Ss      0:00 bash
python--166--     10407 ?      S       0:00 [kworker/0:2]
python--167--     10409 pts/18     R+      0:00 ps -ax
python@ubuntu:~/Desktop$ kill 10354
python@ubuntu:~/Desktop$
```

杀死10354
对应的程序
即强制结束

有些进程不能直接杀死，这时候我们需要加一个参数“-9”，“-9”代表强制结束：

<6>关机重启：reboot、shutdown、init

命令	含义
reboot	重新启动操作系统
shutdown -r now	重新启动操作系统，shutdown会给别的用户提示
shutdown -h now	立刻关机，其中now相当于时间为0的状态
shutdown -h 20:25	系统在今天的20:25 会关机
shutdown -h +10	系统再过十分钟后自动关机
init 0	关机
init 6	重启

<7>检测磁盘空间：df

df命令用于检测文件系统的磁盘空间占用和空余情况，可以显示所有文件系统对节点和磁盘块的使用情况。

选项	含义
-a	显示所有文件系统的磁盘使用情况
-m	以1024字节为单位显示
-t	显示各指定文件系统的磁盘空间使用情况
-T	显示文件系统

```
python@ubuntu:/home$ sudo df -m
文件系统      1M-块  已用  可用  已用%  挂载点
udev          469    0   469    0%  /dev
tmpfs         98    10   89    10%  /run
/dev/sda1    21038  9454 10493   48%  /
tmpfs         488    1   488    1%  /dev/shm
tmpfs         5      1    5     1%  /run/lock
tmpfs         488    0   488    0%  /sys/fs/cgroup
tmpfs         98    1    98    1%  /run/user/108
tmpfs         98    1    98    1%  /run/user/1000
python@ubuntu:/home$
```

<8>检测目录所占磁盘空间：du

du命令用于统计目录或文件所占磁盘空间的大小，该命令的执行结果与df类似，du更侧重于磁盘的使用状况。

du命令的使用格式如下： du [选项] 目录或文件名

选项	含义
-a	递归显示指定目录中各文件和子目录中文件占用的数据块
-s	显示指定文件或目录占用的数据块
-b	以字节为单位显示磁盘占用情况
-l	计算所有文件大小，对硬链接文件计算多次


```
python@ubuntu:~/Desktop/01-Python基础班/beautifulsoup4-4.3.2$ du
52      ./build/lib.linux-x86_64-2.7/bs4/builder
124     ./build/lib.linux-x86_64-2.7/bs4/tests
324     ./build/lib.linux-x86_64-2.7/bs4
328     ./build/lib.linux-x86_64-2.7
52      ./build/lib/bs4/builder
124     ./build/lib/bs4/tests
324     ./build/lib/bs4
328     ./build/lib
660     ./build
12      ./scripts
92      ./bs4/builder
124     ./bs4/tests
452     ./bs4
140     ./doc/source
152     ./doc
1344    .
python@ubuntu:~/Desktop/01-Python基础班/beautifulsoup4-4.3.2$ du -h
52K     ./build/lib.linux-x86_64-2.7/bs4/builder
124K    ./build/lib.linux-x86_64-2.7/bs4/tests
324K    ./build/lib.linux-x86_64-2.7/bs4
328K    ./build/lib.linux-x86_64-2.7
52K     ./build/lib/bs4/builder
124K    ./build/lib/bs4/tests
324K    ./build/lib/bs4
328K    ./build/lib
660K    ./build
12K     ./scripts
92K     ./bs4/builder
python@ubuntu:~/Desktop/01-Python基础班/beautifulsoup4-4.3.2$ du -s build/ -h
660K    build/
python@ubuntu:~/Desktop/01-Python基础班/beautifulsoup4-4.3.2$ █ 指定路径
```

<9>查看或配置网卡信息：ifconfig

如果，我们只是敲：ifconfig，它会显示所有网卡的信息：

```
python@ubuntu:/home$ ifconfig
ens33  Link encap:以太网 硬件地址 00:0c:29:2b:0b:2b
       inet 地址:192.168.1.107 广播:192.168.1.255 掩码:255.255.255.0
       inet6 地址: fe80::e48b:c9c7:ea2c:78b2/64 Scope:Link
       UP BROADCAST RUNNING MULTICAST MTU:1500 跃点数:1
       接收数据包:22515 错误:0 丢弃:0 过载:0 帧数:0
       发送数据包:3830 错误:0 丢弃:0 过载:0 载波:0
       碰撞:0 发送队列长度:1000
       接收字节:7333381 (7.3 MB) 发送字节:967383 (967.3 KB)

lo     Link encap:本地环回 ip地址
       inet 地址:127.0.0.1 掩码:255.0.0.0
       inet6 地址: ::1/128 Scope:Host
       UP LOOPBACK RUNNING MTU:65536 跃点数:1
       接收数据包:2924 错误:0 丢弃:0 过载:0 帧数:0
       发送数据包:2924 错误:0 丢弃:0 过载:0 载波:0
       碰撞:0 发送队列长度:1
       接收字节:855307 (855.3 KB) 发送字节:855307 (855.3 KB)
```

```
python@ubuntu:/home$ ifconfig ens33 192.168.1.108
SIOCSIFADDR: 不允许的操作
SIOCSIFFLAGS: 不允许的操作
python@ubuntu:/home$
python@ubuntu:/home$
python@ubuntu:/home$ sudo ifconfig ens33 192.168.1.108 修改ens33
python@ubuntu:/home$ 权限 的ip
python@ubuntu:/home$ ifconfig
ens33    Link encap:以太网  硬件地址 00:0c:29:2b:0b:2b
         inet 地址:192.168.1.108  广播:192.168.1.255  掩码:255.255.255.0
         inet6 地址: fe80::e48b:c9c7:ea2c:78b2/64  Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  跃点数:1
         接收数据包:22550  错误:0  丢弃:0  过载:0  帧数:0
         发送数据包:3857  错误:0  丢弃:0  过载:0  载波:0
         碰撞:0  发送队列长度:1000
         接收字节:7336241 (7.3 MB)  发送字节:970854 (970.8 KB)

lo       Link encap:本地环回
         inet 地址:127.0.0.1  掩码:255.0.0.0
         inet6 地址: ::1/128  Scope:Host
         UP LOOPBACK RUNNING  MTU:65536  跃点数:1
         接收数据包:2964  错误:0  丢弃:0  过载:0  帧数:0
         发送数据包:2964  错误:0  丢弃:0  过载:0  载波:0
         碰撞:0  发送队列长度:1
         接收字节:864523 (864.5 KB)  发送字节:864523 (864.5 KB)
```

<10>测试远程主机连通性：ping

```
python@ubuntu:/home$ ping www.baidu.com
PING www.a.shifen.com (61.135.169.121) 56(84) bytes of data.
64 bytes from 61.135.169.121: icmp_seq=1 ttl=47 time=5.33 ms
64 bytes from 61.135.169.121: icmp_seq=2 ttl=47 time=24.5 ms
^C
--- www.a.shifen.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 5.332/14.923/24.515/9.592 ms
python@ubuntu:/home$
python@ubuntu:/home$
python@ubuntu:/home$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=1.09 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=9.87 ms
^C
--- 192.168.1.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.099/5.488/9.878/4.390 ms
python@ubuntu:/home$
```

ip、域名都可

Linux命令-用户、权限管理

用户是Unix/Linux系统工作中重要的一环，用户管理包括用户与组账号的管理。

在Unix/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。

Unix/Linux系统中的root账号通常用于系统的维护和管理，它对Unix/Linux操作系统的所有部分具有不受限制的访问权限。

在Unix/Linux安装的过程中，系统会自动创建许多用户账号，而这些默认的用户就称为“标准用户”。

在大多数版本的Unix/Linux中，都不推荐直接使用root账号登录系统。

<1>查看当前用户：whoami

whoami该命令用户查看当前系统当前账号的用户名。可通过cat /etc/passwd查看系统用户信息。

由于系统管理员通常需要使用多种身份登录系统，例如通常使用普通用户登录系统，然后再以su命令切换到root身份对传统进行管理。这时候就可以使用whoami来查看当前用户的身份。

```
python@ubuntu:~/Desktop$ whoami
python
python@ubuntu:~/Desktop$ █
```

<2>查看登录用户：who

who命令用于查看当前所有登录系统的用户信息。

常用选项：

选项	含义
-m或am l	只显示运行who命令的用户名、登录终端和登录时间
-q或--count	只显示用户的登录账号和登录用户的数量
-u或--heading	显示列标题

```
python@ubuntu:~/Desktop$ who
python  tty7          2016-12-18 14:21 (:0)
python  pts/20           2016-12-19 13:20 (172.16.0.150)
python@ubuntu:~/Desktop$
```

<3>退出登录账户： **exit**

如果是图形界面，退出当前终端；

如果是使用ssh远程登录，退出登陆账户；

如果是切换后的登陆用户，退出则返回上一个登陆账号。

<4>添加用户账号： **useradd**

在Unix/Linux中添加用户账号可以使用adduser或useradd命令，因为adduser命令是指向useradd命令的一个链接，因此，这两个命令的使用格式完全一样。

useradd命令的使用格式如下： `useradd [参数] 新建用户账号`

参数	含义
-d	指定用户登录系统时的主目录，如果不使用该参数，系统自动在/home目录下建立与用户名同名目录为主目录
-m	自动建立目录
-g	指定组名称

相关说明：

- Linux每个用户都要有一个主目录，主目录就是第一次登陆系统，用户的默认当前目录(/home/用户)；
- 每一个用户必须有一个主目录，所以用useradd创建用户的时候，一定给用户指定一个主目录；
- 用户的主目录一般要放到根目录的home目录下，用户的主目录和用户名是相同的；
- 如果创建用户的时候，不指定组名，那么系统会自动创建一个和用户名一样的组名。

命令	含义
<code>useradd -d /home/abc abc -m</code>	创建abc用户，如果/home/abc目录不存在，就自动创建这个目录，同时用户属于abc组

useradd -d /home/a a -g test -m	创建一个用户名字叫a，主目录在/home/a，如果主目录不存在，就自动创建主目录，同时用户属于test组
cat /etc/passwd	查看系统当前用户名

```
python@ubuntu:/home$ ls
dongGe laosong lili python share
python@ubuntu:/home$ useradd dong4716138 -m -d /home/dong4716138
useradd: Permission denied.
useradd: 无法锁定 /etc/passwd, 请稍后再试。
python@ubuntu:/home$ sudo useradd dong4716138 -m -d /home/dong4716138
[sudo] python 的密码:
python@ubuntu:/home$ ls
dong4716138 dongGe laosong lili python share
```

这个新用户的
密码

<5>设置用户密码：passwd

在Unix/Linux中，超级用户可以使用passwd命令为普通用户设置或修改用户口令。用户也可以直接使用该命令来修改自己的口令，而无需在命令后面使用用户名。

```
python@ubuntu:/home$ ls
dong4716138 dongGe laosong lili python share
python@ubuntu:/home$ passwd dong4716138
passwd: 您不能查看或更改 dong4716138 的密码信息。
python@ubuntu:/home$ sudo passwd dong4716138
输入新的 UNIX 密码:
重新输入新的 UNIX 密码:
passwd: 已成功更新密码
```

修改密码

<6>删除用户：userdel

命令	含义
userdel abc(用户名)	删除abc用户，但不会自动删除用户的主目录
userdel -r abc(用户名)	删除用户，同时删除用户的主目录

```

root@ubuntu:/home# userdel -r dongGe
userdel: dongGe 邮件池 (/var/mail/dongGe) 未找到
root@ubuntu:/home# ls
python
root@ubuntu:/home# groupmod
adm          fax          mlocate     root         systemd-journal
audio       floppy      mongodb     rtkit        systemd-network
avahi       ftp         mysql       sambashare   systemd-resolve
avahi-autoipd  games     netdev      saned        systemd-timesync
backup      gnats      news        sasl         tape
bin         input     nogroup     scanner      tty
bluetooth   irc        nopasswdlogin shadow        users
cdrom       kmem      operator    src          utmp
color       lightdm   plugdev     ssh          uucp
cron*      list      postgres   ssl-cert    uidd
daemon     lp        proxy       staff        video
dialout    lpadmin   pulse      sudo         voice
dip        mail     pulse-access sys          whoopsie
disk       man      pulse-access syslog       www-data
elasticsearch  messagebus  redis      systemd-bus-proxy

```

<7>切换用户：su

可以通过su命令切换用户，su后面可以加“-”。su和su -命令不同之处在于，su -切换到对应的用户时会将当前的工作目录自动转换到切换后的用户主目录：

```

python@ubuntu:/home$ whoami
python
python@ubuntu:/home$
python@ubuntu:/home$
python@ubuntu:/home$ su dong4716138
密码：
dong4716138@ubuntu:/home$
dong4716138@ubuntu:/home$ whoami
dong4716138
dong4716138@ubuntu:/home$ pwd
/home
dong4716138@ubuntu:/home$ exit
exit
python@ubuntu:/home$
python@ubuntu:/home$ whoami
python
python@ubuntu:/home$
python@ubuntu:/home$ su - dong4716138
密码：
dong4716138@ubuntu:~$ whoami
dong4716138
dong4716138@ubuntu:~$ pwd
/home/dong4716138
dong4716138@ubuntu:~$

```

不会主动切换到
dong4716138
这个用户的
家目录

登录成功后主
动切换到
dong4716138
的家目录

注意：如果是ubuntu平台，需要在命令前加“sudo”，如果在某些操作需要管理员才能操作，ubuntu无需切换到root用户即可操作，只需加“sudo”即可。sudo是ubuntu平台下允许系统管理员让普通用户执行一些或者全部的root命令的一个工具，减少了root用户的登陆和管理时间，提高了安全性。

命令	含义
su	切换到root用户

su root	切换到root用户
su -	切换到root用户，同时切换目录到/root
su - root	切换到root用户，同时切换目录到/root
su 普通用户	切换到普通用户
su - 普通用户	切换到普通用户，同时切换普通用户所在的目录

Ubuntu下切换到root的简单命令:

```
python@ubuntu:~$ whoami
python
python@ubuntu:~$ sudo -s
[sudo] python 的密码:
root@ubuntu:~#
root@ubuntu:~#
root@ubuntu:~# whoami
root
root@ubuntu:~#
```

切换到root

输入当前python用户的密码

\$表示普通用户
#表示root用户

<8>查看有哪些用户组

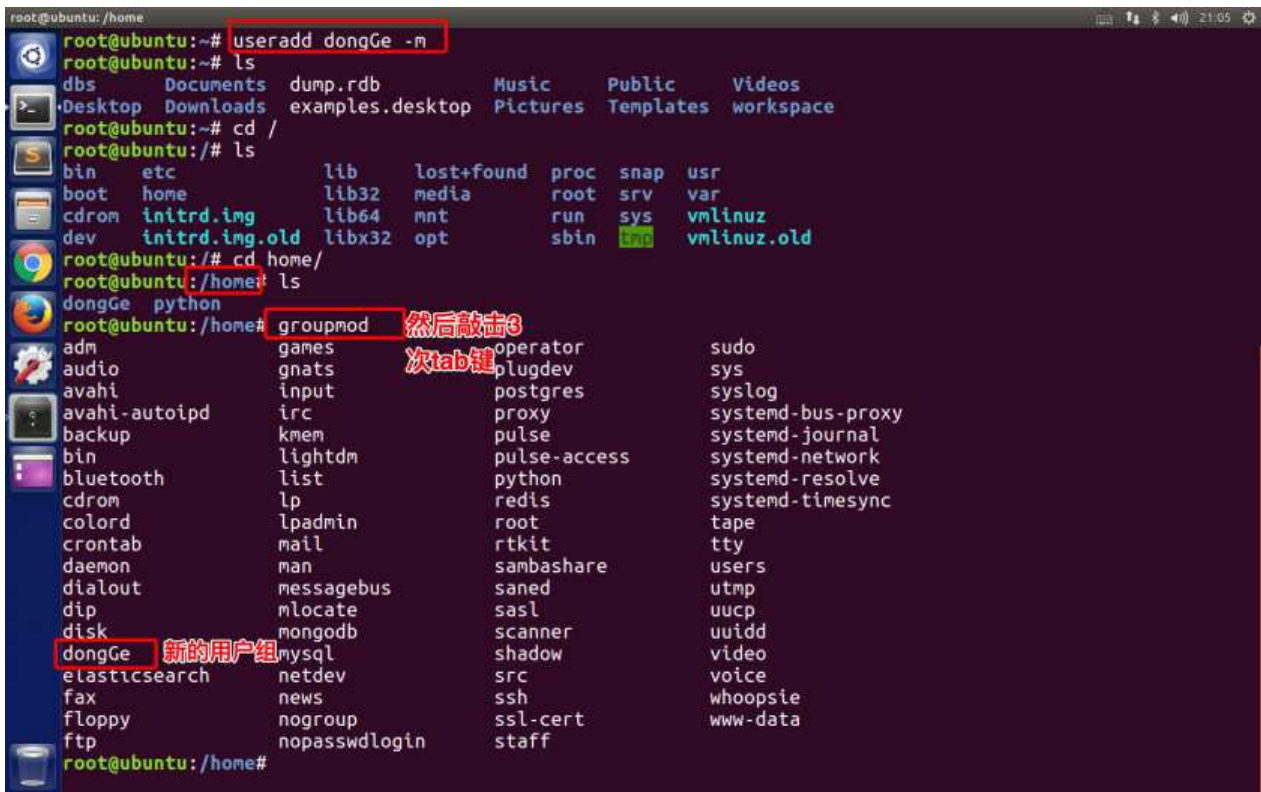
方法一:

```
cat /etc/group
```

```
root@ubuntu:/home# cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,python
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:
man:x:12:
proxy:x:13:
kmem:x:15:
dialout:x:20:
fax:x:21:
voice:x:22:
cdrom:x:24:python
floppy:x:25:
tape:x:26:
sudo:x:27:python
```

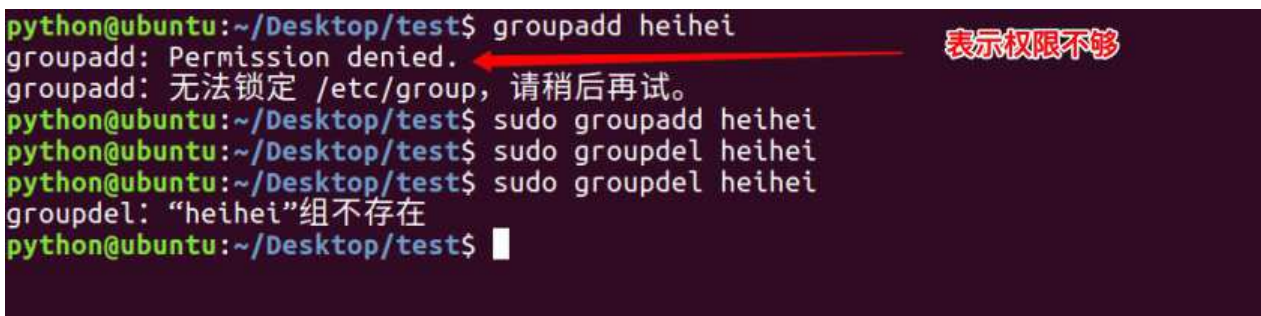

方法二:

groupmod +三次tab键



<9>添加、删除组账号: groupadd、groupdel

groupadd 新建组账号 groupdel 组账号 cat /etc/group 查看用户组



```

root@ubuntu:/home# groupadd XXX
root@ubuntu:/home# useradd dongGe -m
root@ubuntu:/home# ls
dongGe python
root@ubuntu:/home# groupmod
adm          fax          mongodb     sambashare  systemd-timesync
audio       floppy      mysql       saned       tape
avahi       ftp         netdev      sasl        tty
avahi-autoipd  games     news        scanner     users
backup     gnats      nogroup     shadow      utmp
bin        input     nopasswdlogin  src         uucp
bluetooth  irc        operator    ssh         uuid
cdrom     kmem      plugdev     ssl-cert   video
colord    lightdm   postgres   staff       voice
crontab   list      proxy       sudo       whoopsie
daemon    lp        pulse      sys        www-data
dialout   lpadmin   pulse-access syslog      XXX
dip       mail      python     systemd-bus-proxy
disk     man       redis      systemd-journal
dongGe   messagebus  root      systemd-network
elasticsearch  mlocate   rtkit     systemd-resolve
root@ubuntu:/home# groupmod █

```

```

root@ubuntu:/home# groupmod
adm          fax          mongodb     sambashare  systemd-timesync
audio       floppy      mysql       saned       tape
avahi       ftp         netdev      sasl        tty
avahi-autoipd  games     news        scanner     users
backup     gnats      nogroup     shadow      utmp
bin        input     nopasswdlogin  src         uucp
bluetooth  irc        operator    ssh         uuid
cdrom     kmem      plugdev     ssl-cert   video
colord    lightdm   postgres   staff       voice
crontab   list      proxy       sudo       whoopsie
daemon    lp        pulse      sys        www-data
dialout   lpadmin   pulse-access syslog      XXX
dip       mail      python     systemd-bus-proxy
disk     man       redis      systemd-journal
dongGe   messagebus  root      systemd-network
elasticsearch  mlocate   rtkit     systemd-resolve
root@ubuntu:/home# █

```

<10>修改用户所在组：usermod

使用方法：usermod -g 用户组 用户名

```
root@ubuntu:/home# usermod -G XXX dongGe
```

```

root@ubuntu:/home#
root@ubuntu:/home# cat /etc/group | grep XXX
XXX:x:1001:dongGe
root@ubuntu:/home#
root@ubuntu:/home#
root@ubuntu:/home#
root@ubuntu:/home# cat /etc/group | grep dongGe
XXX:x:1001:dongGe
dongGe:x:1002:
root@ubuntu:/home# █

```

<11>查看用户在哪些组

```

root@ubuntu:/home# groups dongGe
dongGe : ZZZ MMM
root@ubuntu:/home# usermod -a -G MMM dongGe
root@ubuntu:/home# groups dongGe
dongGe : ZZZ MMM
root@ubuntu:/home# usermod -a -G YYY dongGe
root@ubuntu:/home# groups dongGe
dongGe : ZZZ YYY MMM
root@ubuntu:/home# usermod -a -G XXX dongGe
root@ubuntu:/home# groups dongGe
dongGe : ZZZ XXX YYY MMM

```

<12>为创建的普通用户添加sudo权限

新创建的用户，默认不能sudo，需要进行一下操作

```

sudo usermod -a -G adm 用户名

sudo usermod -a -G sudo 用户名

```

<13>usermod -g 与 -G的区别

-g 用来制定这个用户默认的用户组

-G 一般配合'-a'来完成向其它组添加

```

dongGe@ubuntu:~$ groups dongGe
dongGe : XXX adm sudo
dongGe@ubuntu:~$ touch 123.py
dongGe@ubuntu:~$ ls -lh
总用量 12K
-rw-r--r-- 1 dongGe MMM 0 10月 12 21:57 123.py
-rw-r--r-- 1 dongGe XXX 8.8K 4月 20 16:47 examples.desktop
dongGe@ubuntu:~$

```

```

dongGe@ubuntu:~$ sudo usermod -g YYY dongGe

```

```

dongGe@ubuntu:~$ groups dongGe
dongGe : YYY adm sudo XXX
dongGe@ubuntu:~$ touch 456.py
dongGe@ubuntu:~$ ls -lh
总用量 12K
-rw-r--r-- 1 dongGe MMM 0 10月 12 21:57 123.py
-rw-r--r-- 1 dongGe MMM 0 10月 12 21:58 456.py
-rw-r--r-- 1 dongGe YYY 8.8K 4月 20 16:47 examples.desktop
dongGe@ubuntu:~$

```

<14>修改文件权限：chmod

chmod 修改文件权限有两种使用格式：字母法与数字法。

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

rwX	含义
r	read 表示可读取，对于一个目录，如果没有r权限，那么就意味着不能通过ls查看这个目录的内容。
w	write 表示可写入，对于一个目录，如果没有w权限，那么就意味着不能在目录下创建新的文件。
x	excute 表示可执行，对于一个目录，如果没有x权限，那么就意味着不能通过cd进入这个目录。

```
python@ubuntu:~/Desktop/test$ ls
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ touch test.txt
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ ls -lh
总用量 0
-rw-rw-r-- 1 python python 0 12月 19 13:44 test.txt
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ chmod u+x test.txt
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ ls -lh
总用量 0
-rwxrw-r-- 1 python python 0 12月 19 13:44 test.txt
python@ubuntu:~/Desktop/test$

python@ubuntu:~/Desktop/test$ ls -lh
总用量 0
-rwxrw-r-- 1 python python 0 12月 19 13:44 test.txt
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ chmod g-w test.txt
python@ubuntu:~/Desktop/test$ ls -lh
总用量 0
-rwxr--r-- 1 python python 0 12月 19 13:44 test.txt
python@ubuntu:~/Desktop/test$

python@ubuntu:~/Desktop/test$ ls -lh
总用量 0
-rwxr--r-- 1 python python 0 12月 19 13:44 test.txt
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ chmod o+w test.txt
python@ubuntu:~/Desktop/test$ ls -lh
总用量 0
-rwxr--rwx 1 python python 0 12月 19 13:44 test.txt
python@ubuntu:~/Desktop/test$
```

文件所有者添加执行的权限

同组者减去写的权限

其他人添加写的权限

如果需要同时进行设定拥有者、同组者以及其他人的权限，参考如下：

```
python@ubuntu:~/ftp/share$ ls -lh
总用量 104K
----- 1 python python 138 8月 10 21:19 1.py
-rw----- 1 python python 378 8月 10 20:24 hy.txt
-rw----- 1 python python 75K 8月 10 20:23 Snip20160810_30.png
-rw-rw-r-- 1 python python 0 8月 10 20:20 test.py
-rw----- 1 python python 20K 8月 10 21:03 Thumbs.db
python@ubuntu:~/ftp/share$ chmod u=rw,g=x,o=r 1.py
python@ubuntu:~/ftp/share$ ls -lh
总用量 104K
-rw--xr-- 1 python python 138 8月 10 21:19 1.py
-rw----- 1 python python 378 8月 10 20:24 hy.txt
-rw----- 1 python python 75K 8月 10 20:23 Snip20160810_30.png
-rw-rw-r-- 1 python python 0 8月 10 20:20 test.py
-rw----- 1 python python 20K 8月 10 21:03 Thumbs.db
python@ubuntu:~/ftp/share$
```

```
python@ubuntu:~/ftp/share$ ls -lh
总用量 104K
-rw--xr-- 1 python python 138 8月 10 21:19 1.py
-rw----- 1 python python 378 8月 10 20:24 hy.txt
-rw----- 1 python python 75K 8月 10 20:23 Snip20160810_30.png
-rw-rw-r-- 1 python python 0 8月 10 20:20 test.py
-rw----- 1 python python 20K 8月 10 21:03 Thumbs.db
python@ubuntu:~/ftp/share$ chmod u=,g=,o= hy.txt
python@ubuntu:~/ftp/share$ ls -lh
总用量 104K
-rw--xr-- 1 python python 138 8月 10 21:19 1.py
----- 1 python python 378 8月 10 20:24 hy.txt
-rw----- 1 python python 75K 8月 10 20:23 Snip20160810_30.png
-rw-rw-r-- 1 python python 0 8月 10 20:20 test.py
-rw----- 1 python python 20K 8月 10 21:03 Thumbs.db
python@ubuntu:~/ftp/share$
```

数字法：“rwX”这些权限也可以用数字来代替

字母	说明
r	读取权限，数字代号为“4”
w	写入权限，数字代号为“2”

x	执行权限，数字代号为 "1"
-	不具任何权限，数字代号为 "0"

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

chmod 751 file:

- 文件所有者：读、写、执行权限
- 同组用户：读、执行的权限
- 其它用户：执行的权限

```
python@ubuntu:~/Desktop/test$ ls -lh
总用量 0
-rwxr--r-- 1 python python 0 12月 19 13:44 test.txt
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ chmod 777 test.txt
python@ubuntu:~/Desktop/test$ ls -lh
总用量 0
-rwxrwxrwx 1 python python 0 12月 19 13:44 test.txt
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ chmod 761 test.txt
python@ubuntu:~/Desktop/test$ ls -lh
总用量 0
-rwxr--x 1 python python 0 12月 19 13:44 test.txt
python@ubuntu:~/Desktop/test$
```

注意：如果想递归所有目录加上相同权限，需要加上参数“-R”。如：chmod 777 test/ -R
递归 test 目录下所有文件加 777 权限

<15>修改文件所有者：chown

```
python@ubuntu:~/Desktop/test$ ls -lh
总用量 0
-rwxr--x 1 python python 0 12月 19 13:44 test.txt
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ chown dong4716138 test.txt
chown: 正在更改'test.txt'的所有者: 不允许的操作
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ sudo chown dong4716138 test.txt
python@ubuntu:~/Desktop/test$ ls -lh
总用量 0
-rwxr--x 1 dong4716138 python 0 12月 19 13:44 test.txt
python@ubuntu:~/Desktop/test$
```

<16>修改文件所属组：chgrp

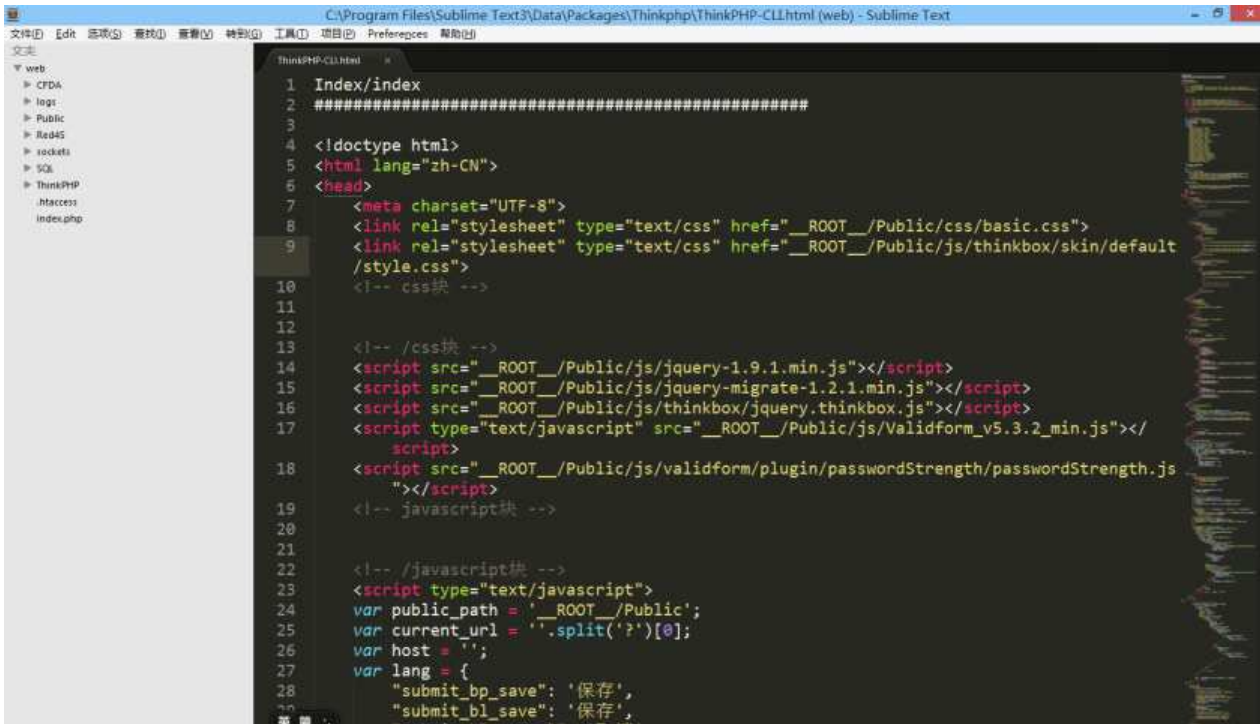
```
python@ubuntu:~/Desktop/test$ ls -lh
总用量 0
-rwxrwx---x 1 dong4716138 python 0 12月 19 13:44 test.txt
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$
python@ubuntu:~/Desktop/test$ sudo chgrp dong4716138 test.txt
python@ubuntu:~/Desktop/test$ ls -lh
总用量 0
-rwxrwx---x 1 dong4716138 dong4716138 0 12月 19 13:44 test.txt
python@ubuntu:~/Desktop/test$
```


gedit编辑器

gedit是一个Linux环境下的文本编辑器，类似windows下的写字板程序，在不需要特别复杂的编程环境下，作为基本的文本编辑器比较合适。



sublime编辑器



Sublime Text 是一个代码编辑器（Sublime Text 2是收费软件，但可以无限期试用）

Sublime Text是由程序员Jon Skinner于2008年1月份所开发出来，它最初被设计为一个具有丰富扩展功能的Vim。

Sublime Text具有漂亮的用户界面和强大的功能，例如代码缩略图，Python的插件，代码段等。

还可自定义键绑定，菜单和工具栏。Sublime Text 的主要功能包括：拼写检查，书签，完整的 Python API，Goto 功能，即时项目切换，多选择，多窗口等等。

Sublime Text 是一个跨平台的编辑器，同时支持Windows、Linux、Mac OS X等操作系统。

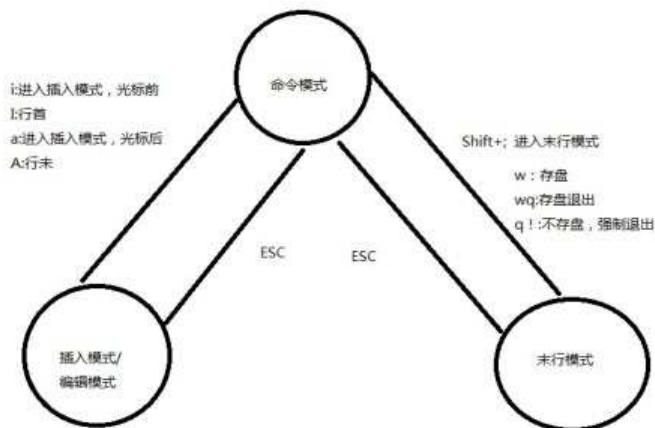
编辑器之神-vim

vi简介

vi是“Visual interface”的简称，它在Linux上的地位就仿佛Edit程序在DOS上一样。它可以执行输出、删除、查找、替换、块操作等众多文本操作，而且用户可以根据自己的需要对其进行定制。Vi不是一个排版程序，它不象Word或WPS那样可以对字体、格式、段落等其他属性进行编排，它只是一个文本编辑程序。vi没有菜单，只有命令，且命令繁多。

Vi有三种基本工作模式：

- + 命令模式
- + 文本输入模式
- + 末行模式。



命令行模式

任何时候，不管用户处于何种模式，只要按一下ESC键，即可使Vi进入命令模式；我们在shell环境(提示符为\$)下输入启动Vi命令，进入编辑器时，也是处于该模式下。在该模式下，用户可以输入各种合法的Vi命令，用于管理自己的文档。此时从键盘上输入的任何字符都被当做编辑命令来解释，若输入的字符是合法的Vi命令，则Vi在接受用户命令之后完成相应的动作。但需注意的是，所输入的命令并不在屏幕上显示出来。若输入的字符不是Vi的合法命令，Vi会响铃报警。

文本输入模式

在命令模式下输入插入命令i、附加命令a、打开命令o、修改命令c、取代命令r或替换命令s都可以进入文本输入模式。在该模式下，用户输入的任何字符都被Vi当做文件内容保存起来，并将其显示在屏幕上。在文本输入过程中，若想回到命令模式下，按键ESC即可。

末行模式

末行模式也称ex转义模式。在命令模式下，用户按“:”键即可进入末行模式下，此时Vi会在显示窗口的最后一行(通常也是屏幕的最后一行)显示一个“:”作为末行模式的提示符，等待用户输入命令。多数文件管理命令都是在此模式下执行的(如把编辑缓冲区的内容写到文件中)。末行命令执行完后，Vi自动回到命令模式。例如：

```
:sp newfile
```

则分出一个窗口编辑newfile文件。如果要从命令模式转换到编辑模式，可以键入命令a或者i；如果需要从文本模式返回，则按Esc键即可。在命令模式下输入“:”即可切换到末行模式，然后输入命令。

vim基础操作

vim是从vi发展出来的一个文本编辑器。代码补完、编译及错误跳转等方便编程的功能特别丰富

进入插入模式:

```
i: 插入光标前一个字符  
I: 插入行首  
a: 插入光标后一个字符  
A: 插入行末  
o: 向下新开一行, 插入行首  
O: 向上新开一行, 插入行首
```

进入命令模式:

ESC:从插入模式或末行模式进入命令模式

移动光标:

h: 左移

j: 下移

k: 上移

l: 右移

M: 光标移动到中间行

L: 光标移动到屏幕最后一行行首

G: 移动到指定行, 行号 -G

w: 向后一次移动一个字

b: 向前一次移动一个字

{: 按段移动, 上移

}: 按段移动, 下移

Ctrl-d: 向下翻半屏

Ctrl-u: 向上翻半屏

Ctrl-f: 向下翻一屏

Ctrl-b: 向上翻一屏

gg: 光标移动文件开头

G: 光标移动到文件末尾

删除命令:

x: 删除光标后一个字符, 相当于 Del

X: 删除光标前一个字符, 相当于 Backspace

dd: 删除光标所在行, n dd 删除指定的行数 D: 删除光标后本行所有内容, 包含光标所在字符

d0: 删除光标前本行所有内容, 不包含光标所在字符

dw: 删除光标开始位置的字符, 包含光标所在字符

撤销命令:

u: 一步一步撤销

Ctrl-r: 反撤销

重复命令:

.: 重复上一次操作的命令

文本行移动:

>>: 文本行右移

<<: 文本行左移

复制粘贴:

yy: 复制当前行, n yy 复制 n 行

p: 在光标所在位置向下新开辟一行, 粘贴

可视模式:

v: 按字符移动, 选中文本

V: 按行移动, 选中文本可视模式可以配合 d, y, >>, << 实现对文本块的删除, 复制, 左右移动

替换操作:

r: 替换当前字符

R: 替换当前行光标后的字符

查找命令:

/: str查找

n: 下一个

N: 上一个

替换命令：

把abc全部替换成123

```
末行模式下，将当前文件中的所有abc替换成123  
:%s/abc/123/g
```

```
末行模式下，将第一行至第10行之间的abc替换成123  
:1, 10s/abc/123/g
```

vim里执行 shell 下命令：

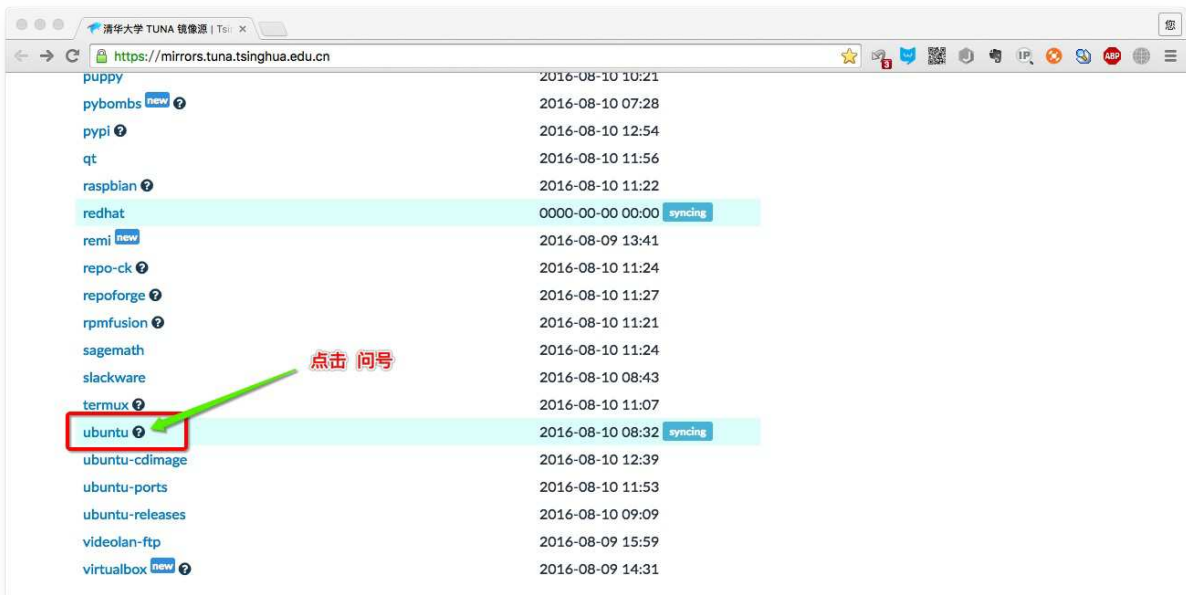
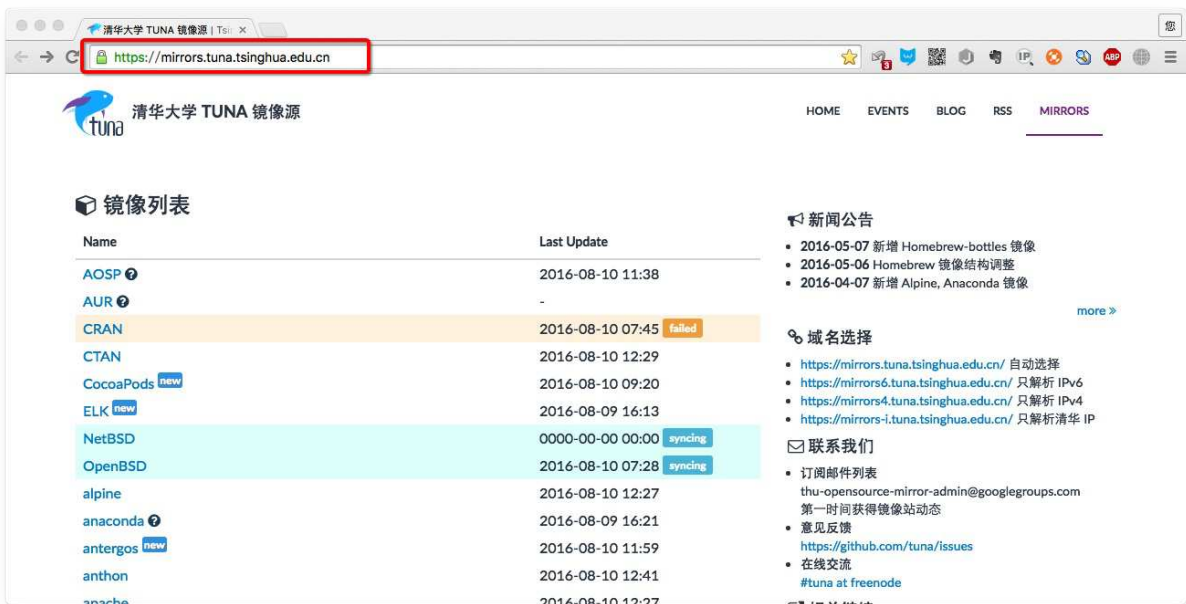
```
末行模式里输入!,后面跟命令
```

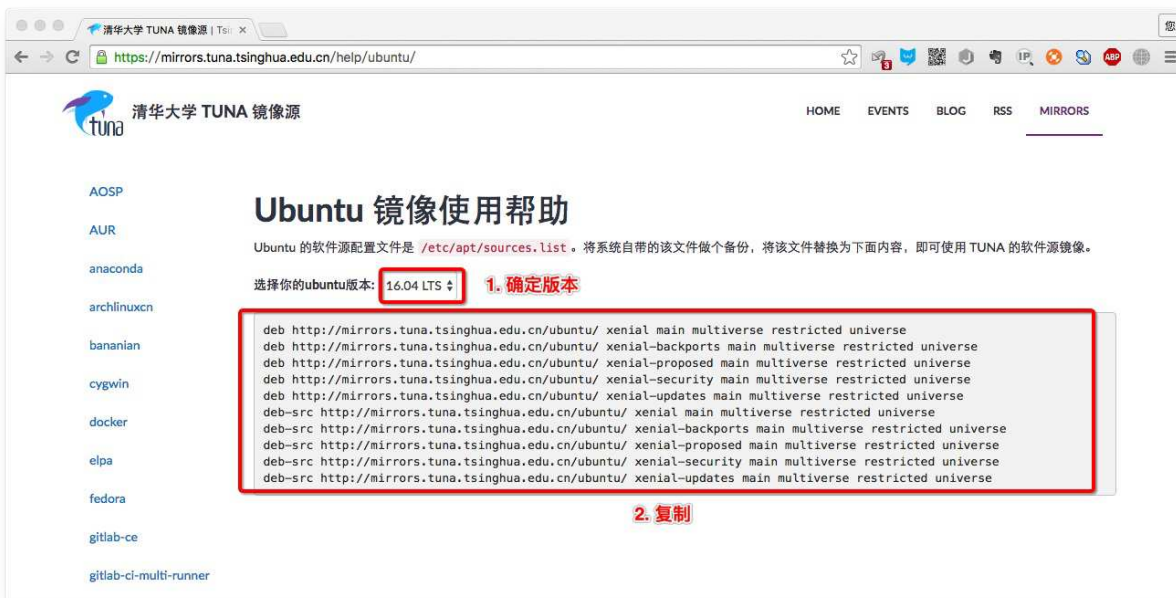
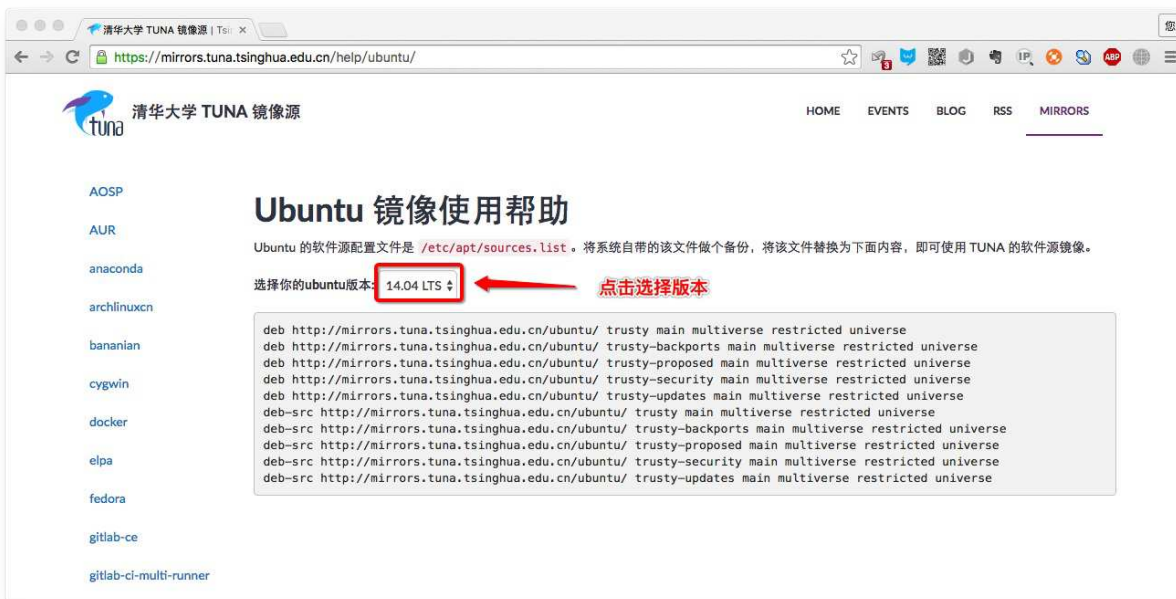

ubuntu软件安装与卸载

更新Ubuntu软件下载地址

1. 寻找国内镜像源

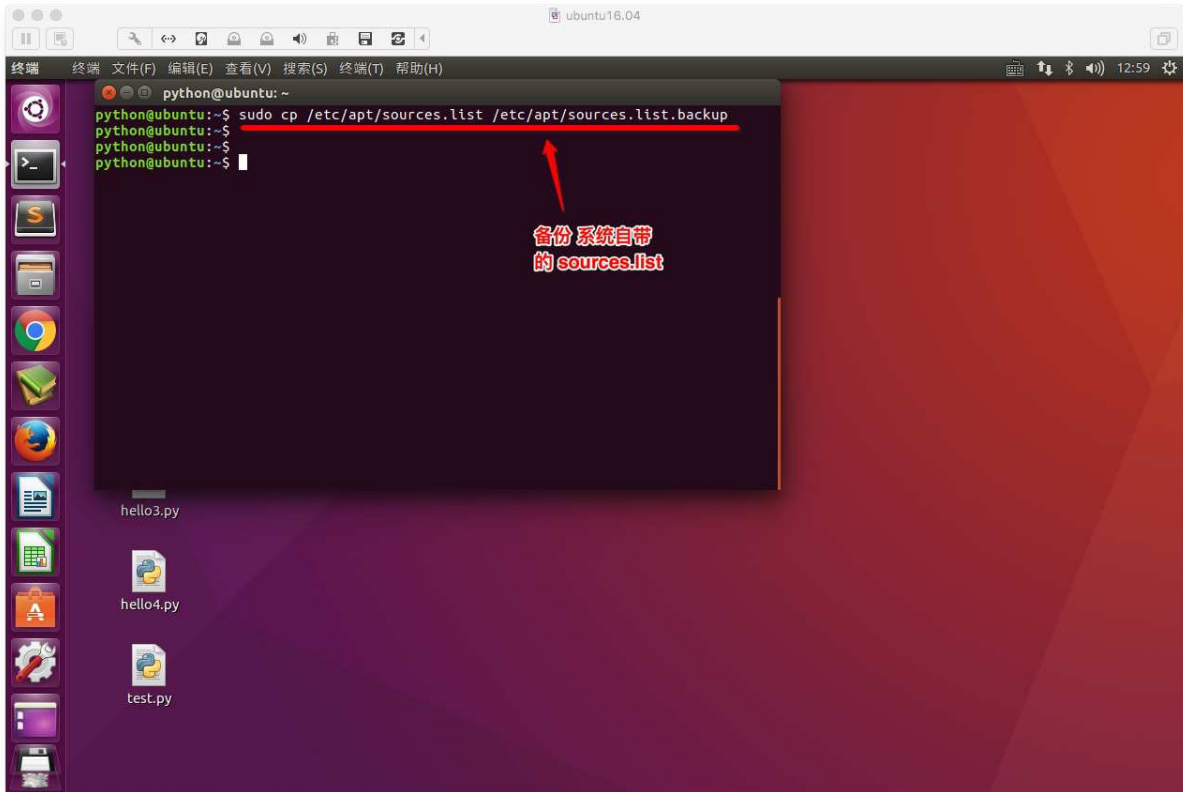
所谓的镜像源：可以理解为提供下载软件的地方，比如Android手机上可以下载软件的91手机助手；iOS手机上可以下载软件的AppStore



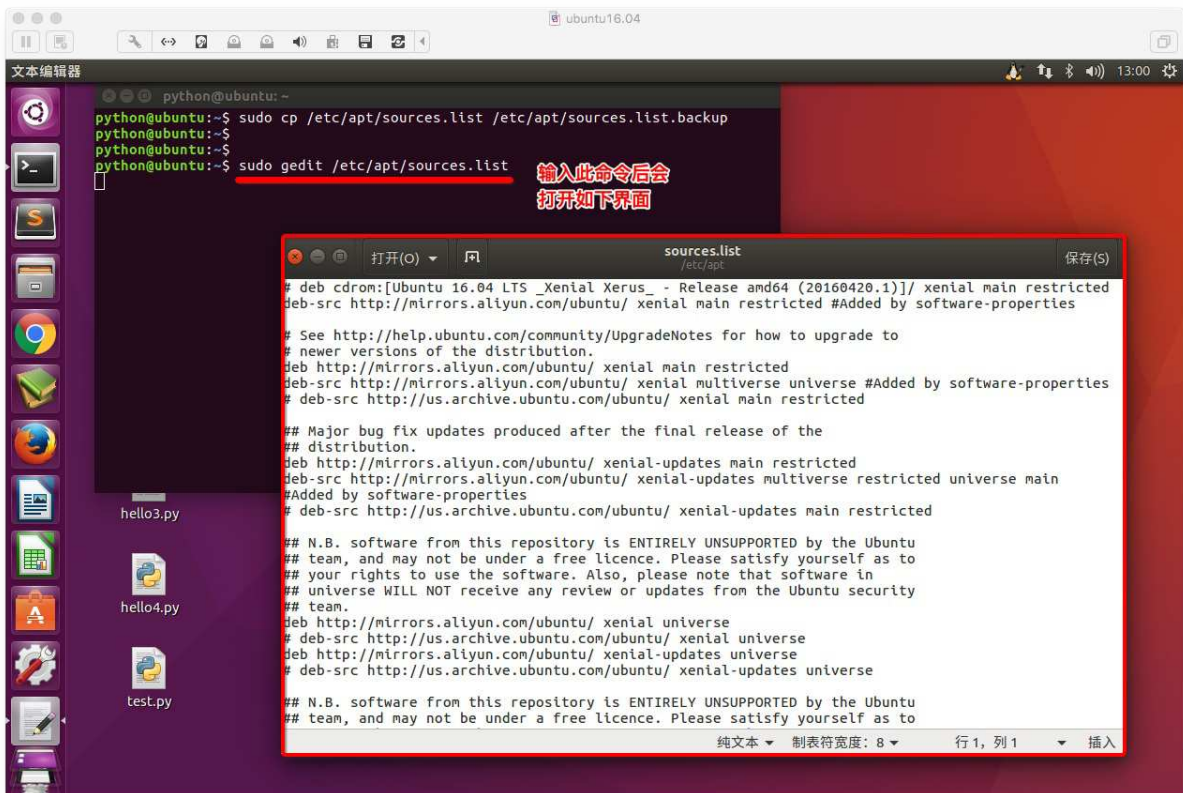


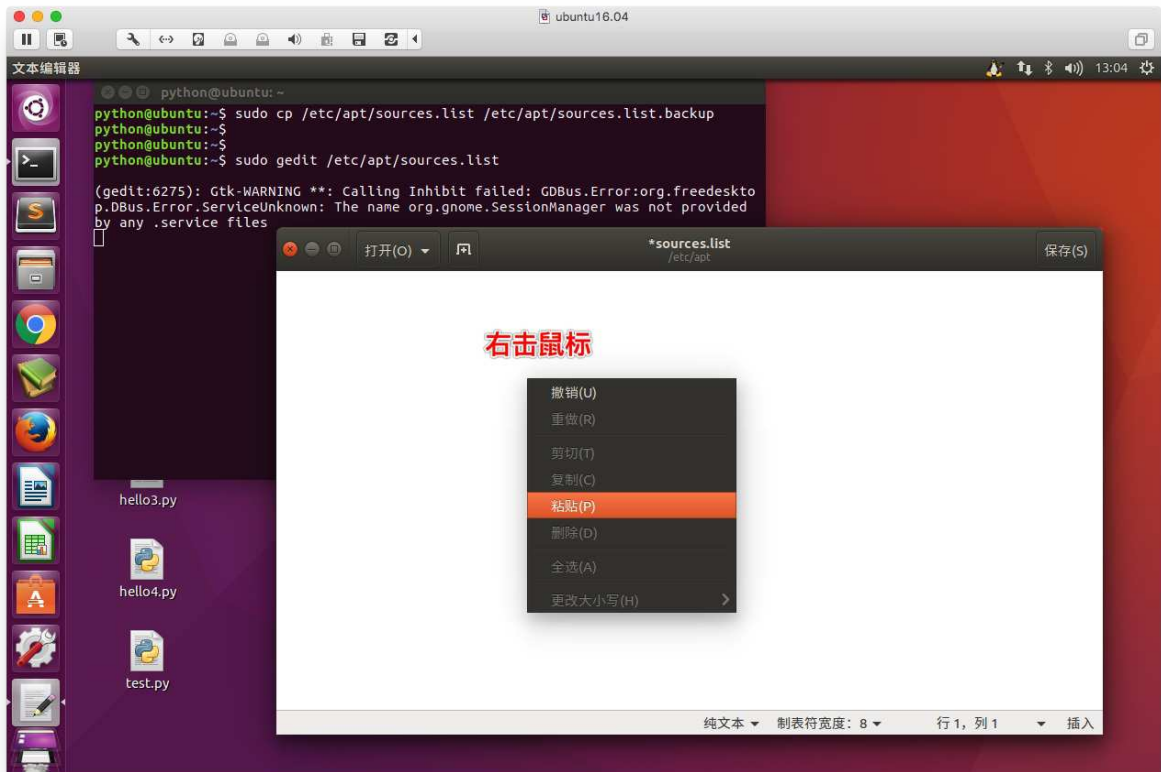
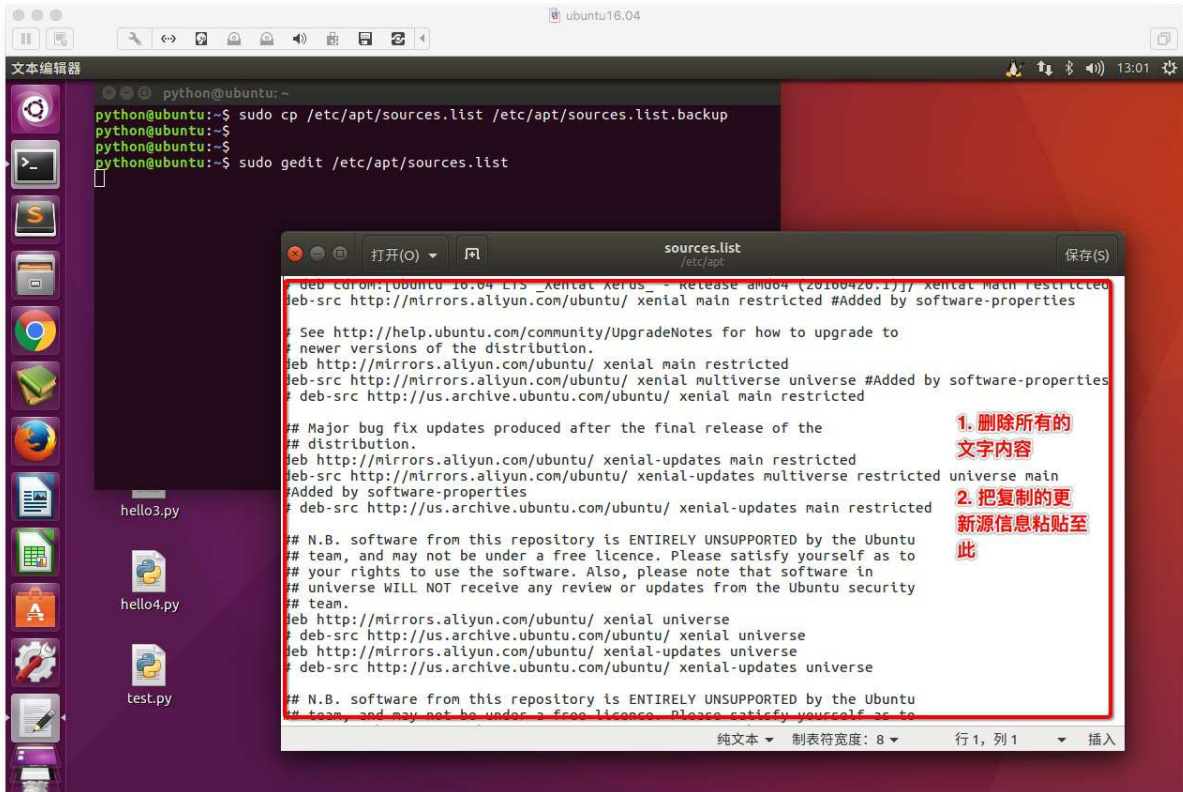
2. 备份Ubuntu默认的源地址

```
sudo cp /etc/apt/sources.list /etc/apt/sources.list.backup
```



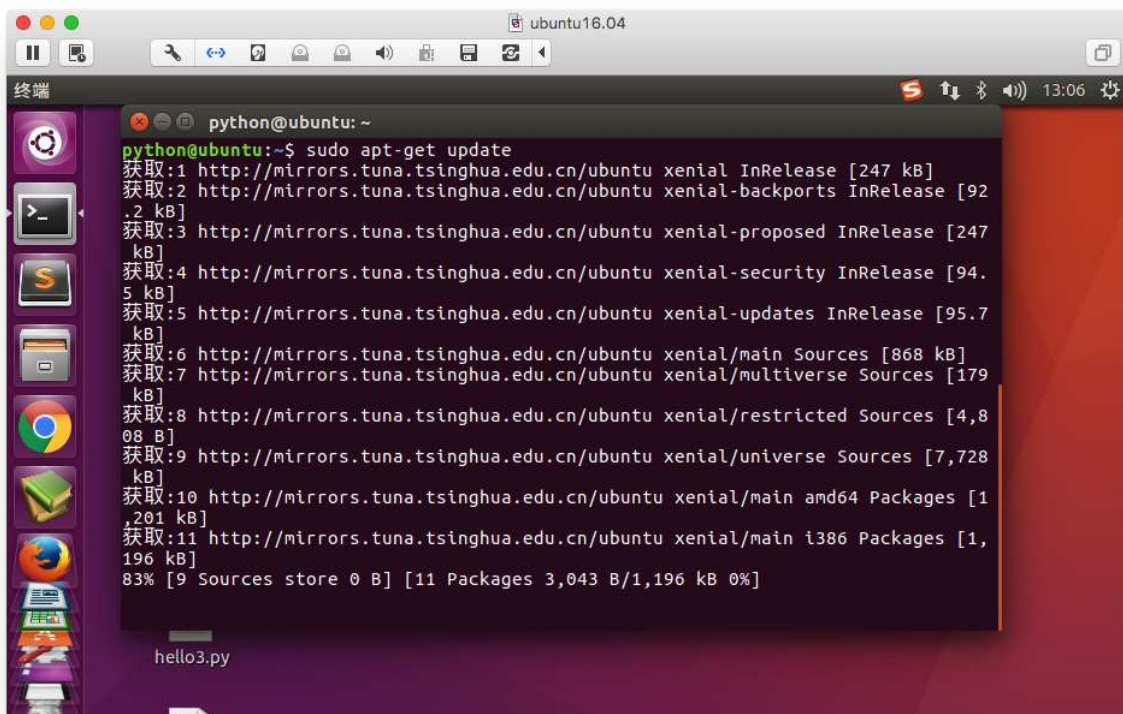
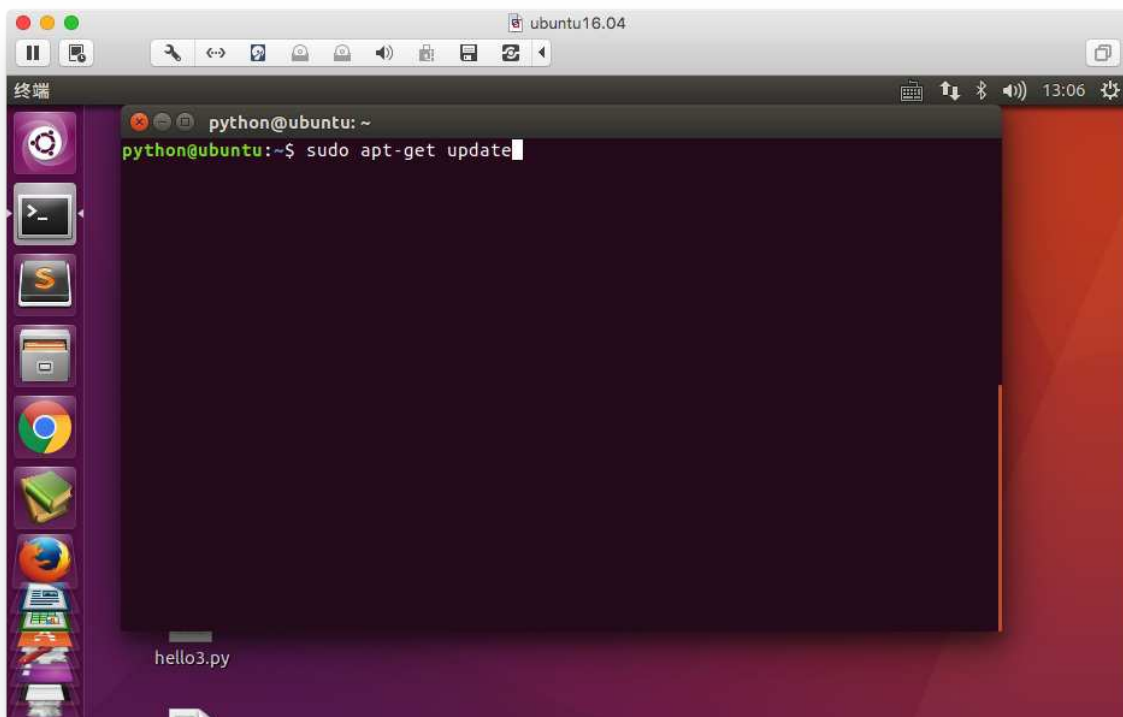
3. 更新源服务器列表

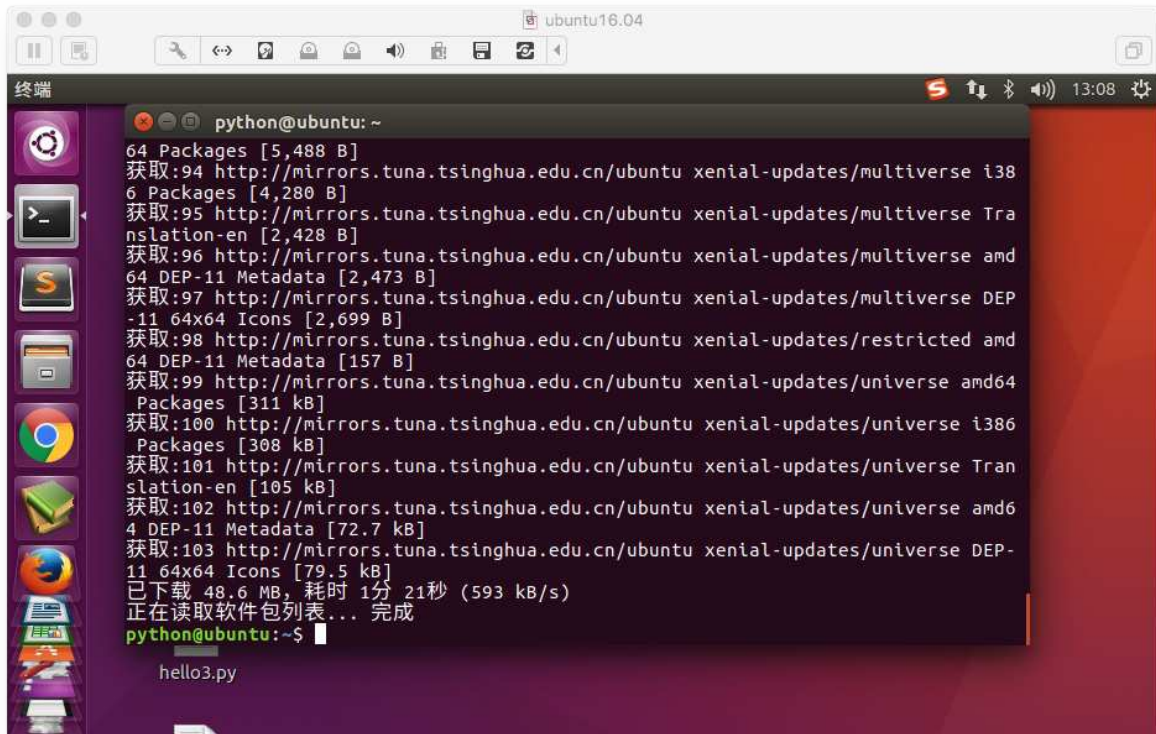




4. 更新源

做完此步骤之后，就可以进行apt-get install 下载了





Ubuntu软件操作的相关命令

`sudo apt-get update` 更新源

`sudo apt-get install package` 安装包

`sudo apt-get remove package` 删除包

`sudo apt-cache search package` 搜索软件包

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

`sudo apt-get install package --reinstall` 重新安装包

`sudo apt-get -f install` 修复安装

`sudo apt-get remove package --purge` 删除包，包括配置文件等

`sudo apt-get build-dep package` 安装相关的编译环境

`sudo apt-get upgrade` 更新已安装的包

`sudo apt-get dist-upgrade` 升级系统

`sudo apt-cache depends package` 了解使用该包依赖那些包

`sudo apt-cache rdepends package` 查看该包被哪些包依赖

`sudo apt-get source package` 下载该包的源代码

`sudo apt-get clean && sudo apt-get autoclean` 清理无用的包

`sudo apt-get check` 检查是否有损坏的依赖

Linux常用服务器构建-ftp服务器

ftp服务器

FTP 是File Transfer Protocol（文件传输协议）的英文简称，而中文简称为“文传协议”。用于Internet上的控制文件的双向传输。

同时，它也是一个应用程序（Application）。基于不同的操作系统有不同的FTP应用程序，而所有这些应用程序都遵守同一种协议以传输文件。

在FTP的使用当中，用户经常遇到两个概念：“下载”（Download）和“上传”（Upload）。

“下载”文件就是从远程主机拷贝文件至自己的计算机上；

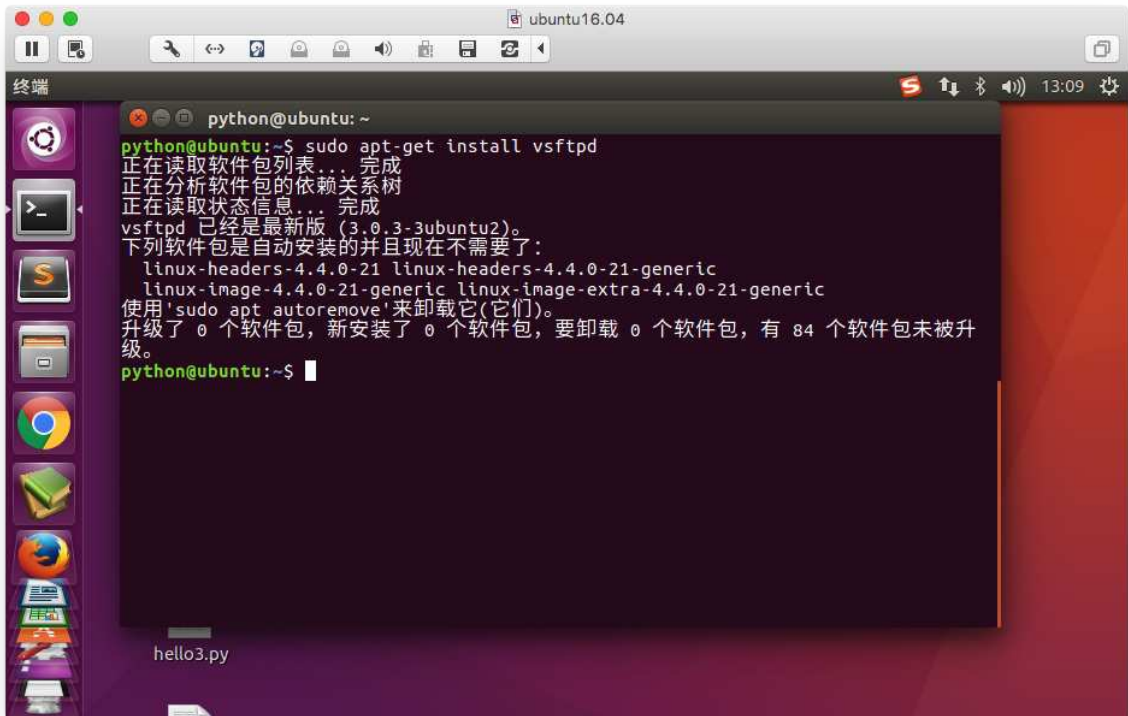
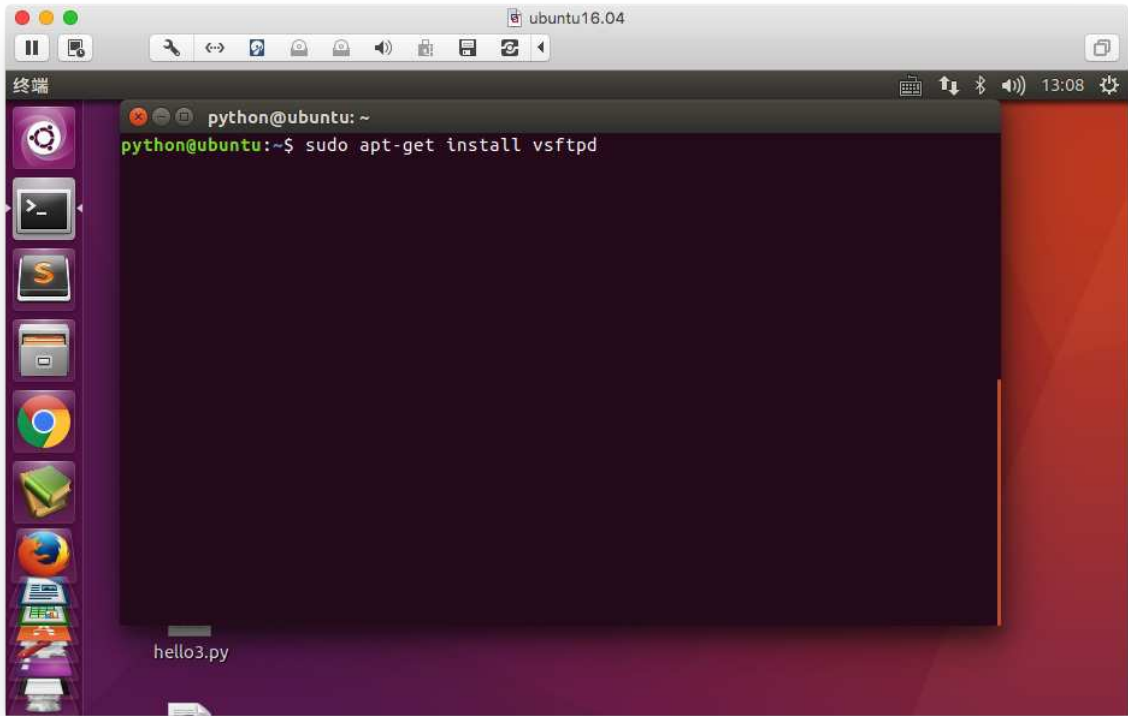
“上传”文件就是将文件从自己的计算机中拷贝至远程主机上。用Internet语言来说，用户可以通过客户机程序向（从）远程主机上传（下载）文件。



FTP架构图

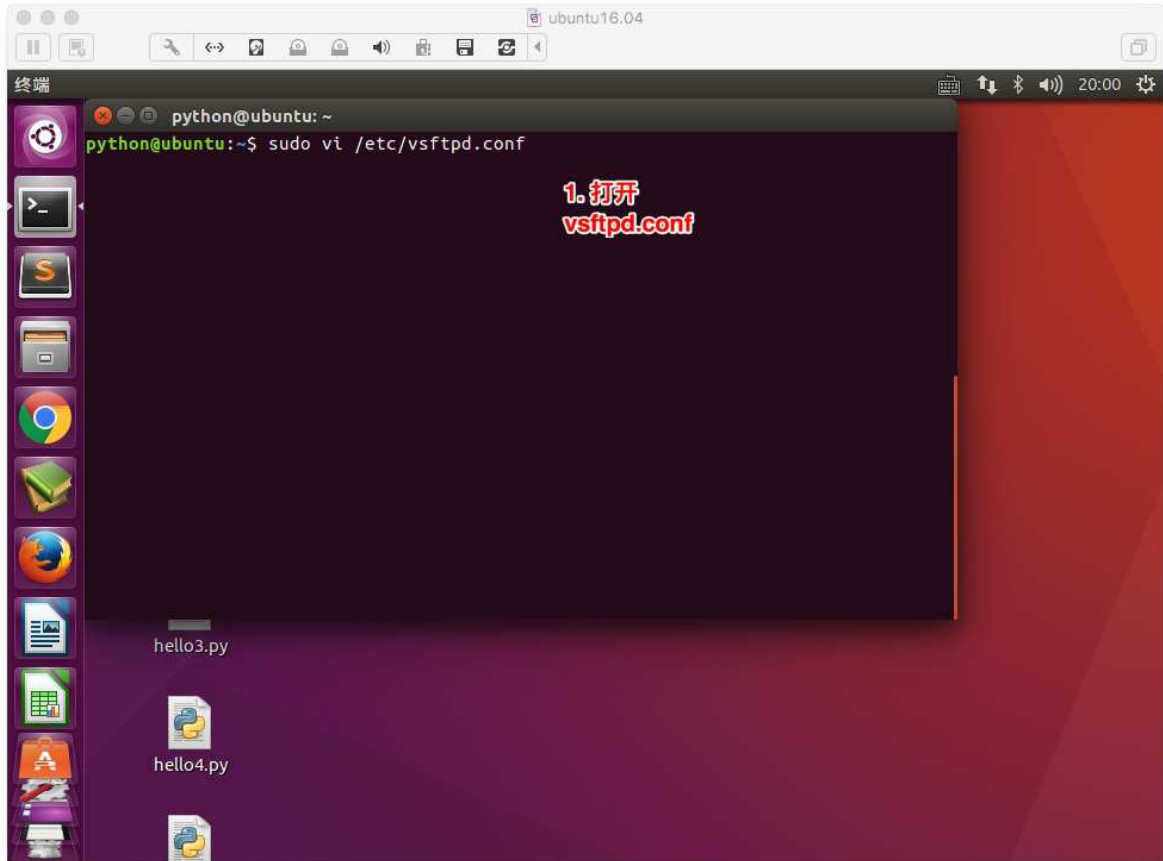
1.安装vsftpd服务器

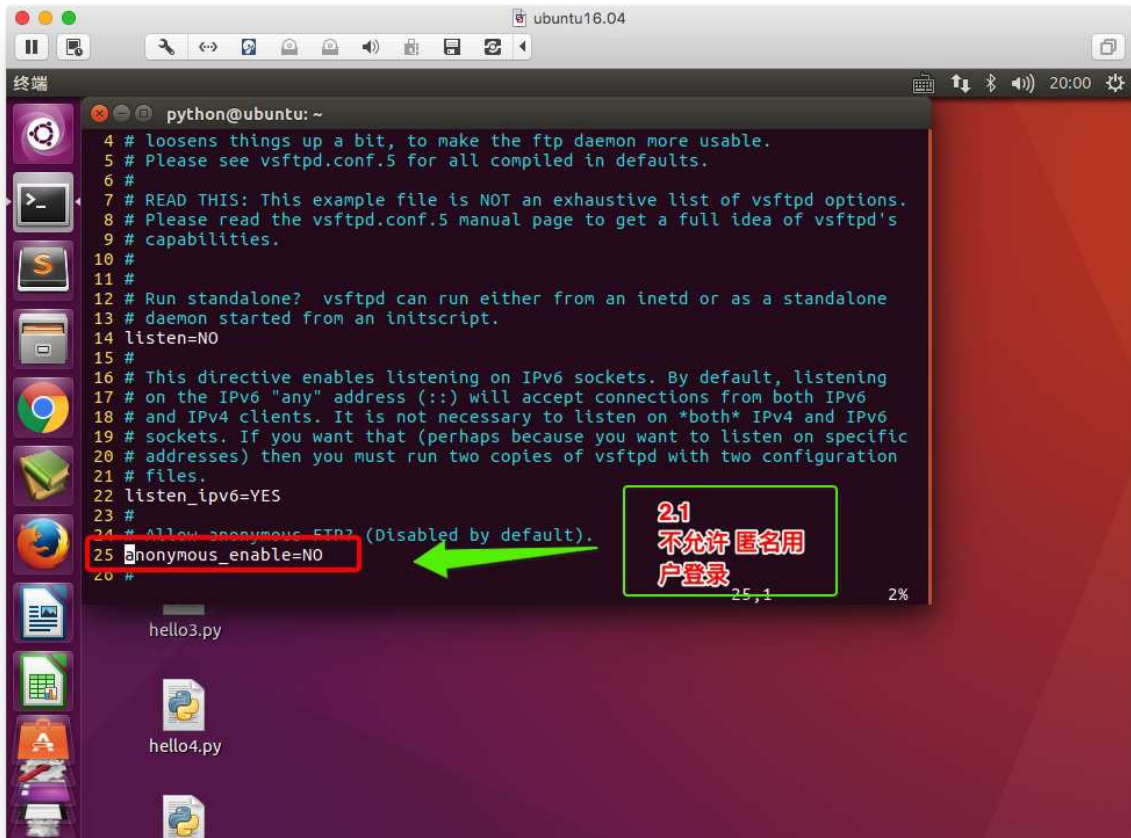
```
sudo apt-get install vsftpd
```

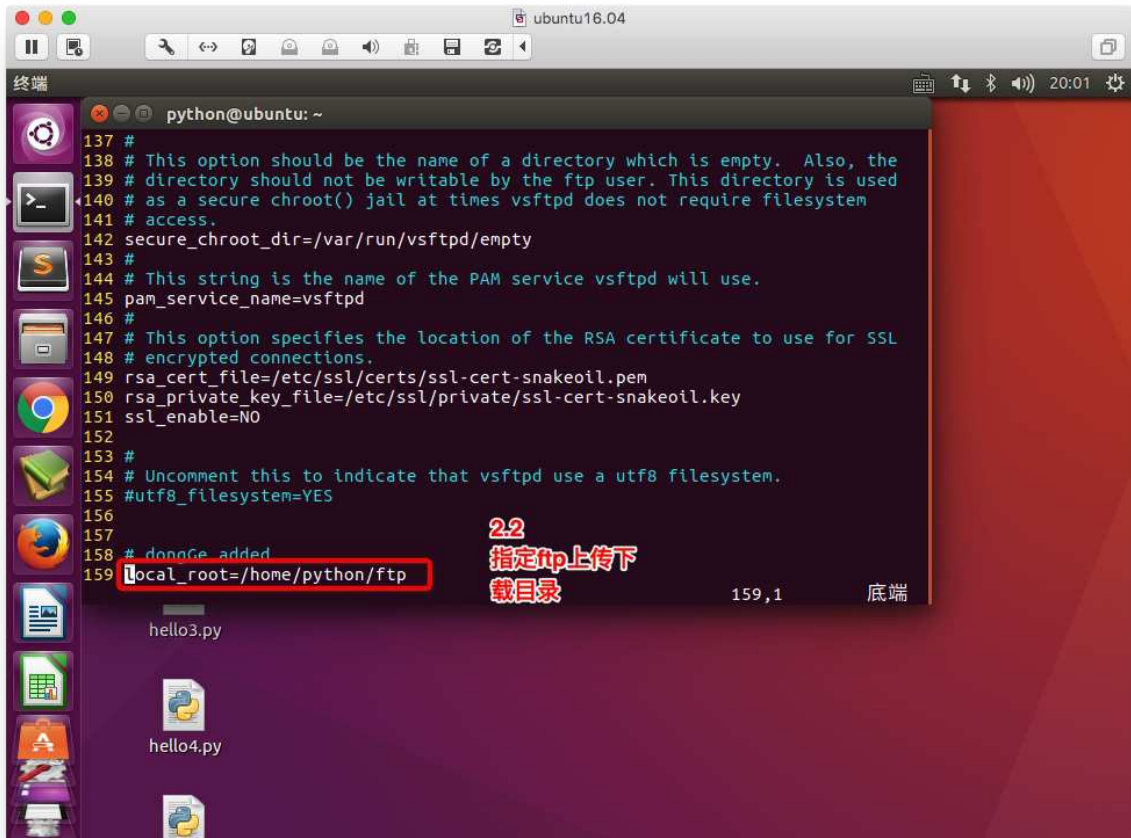



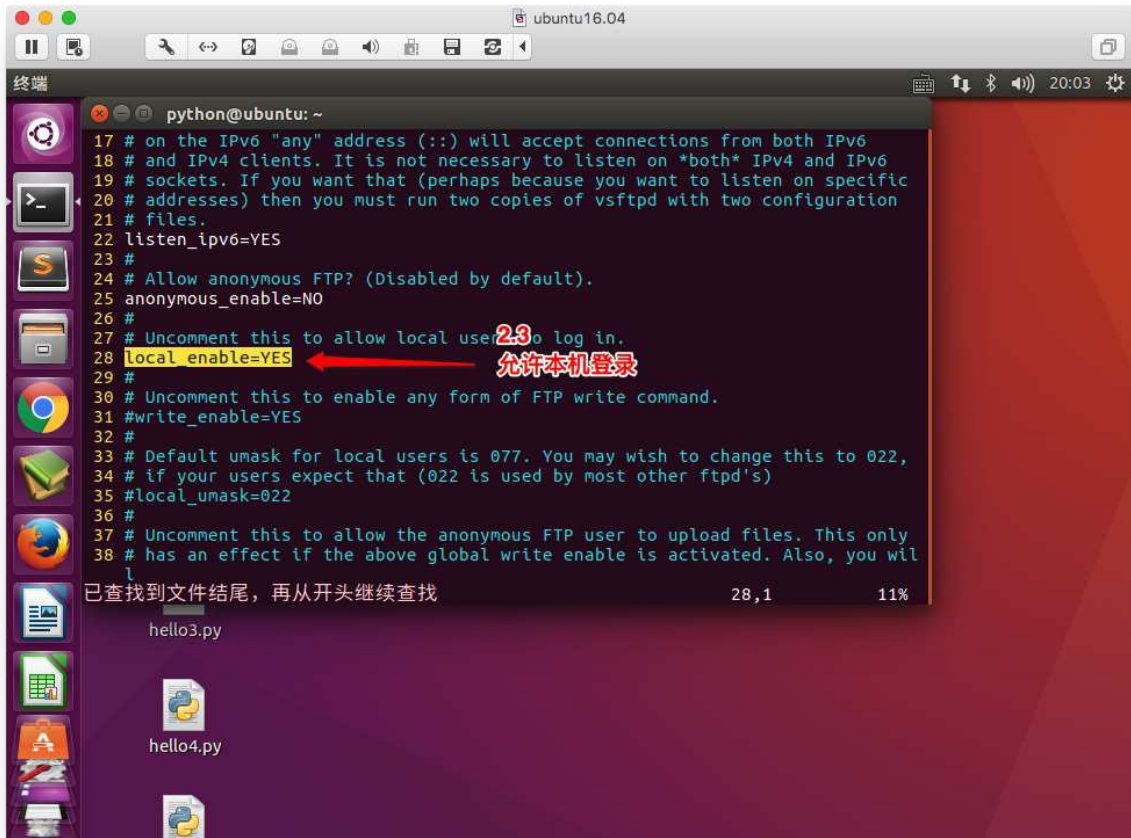
2.配置vsftpd.conf文件

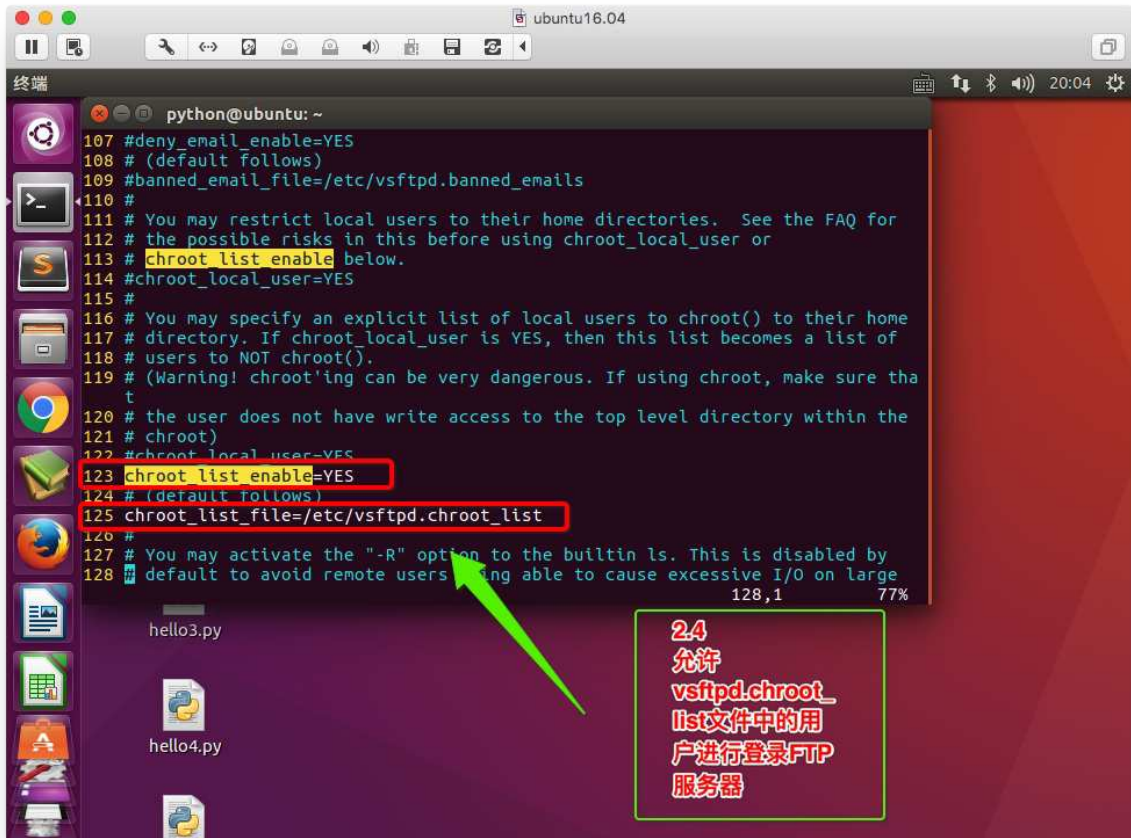
```
sudo vi /etc/vsftpd.conf
```

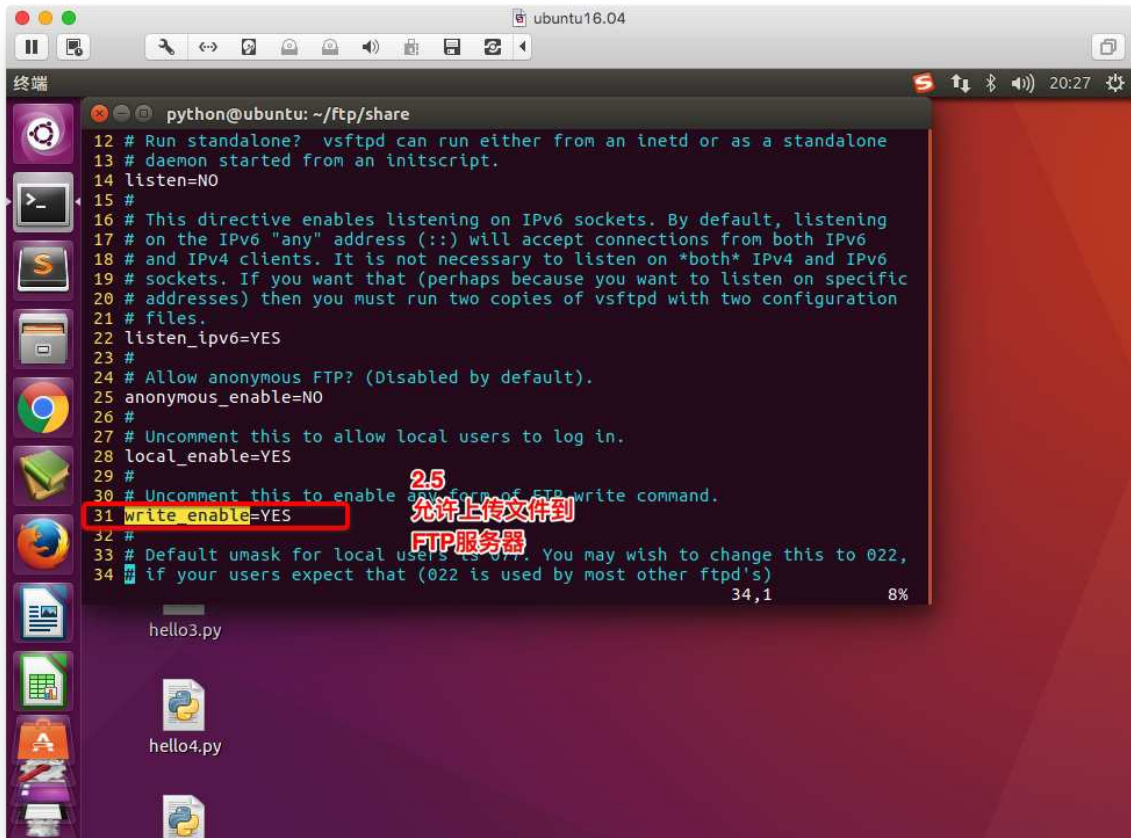


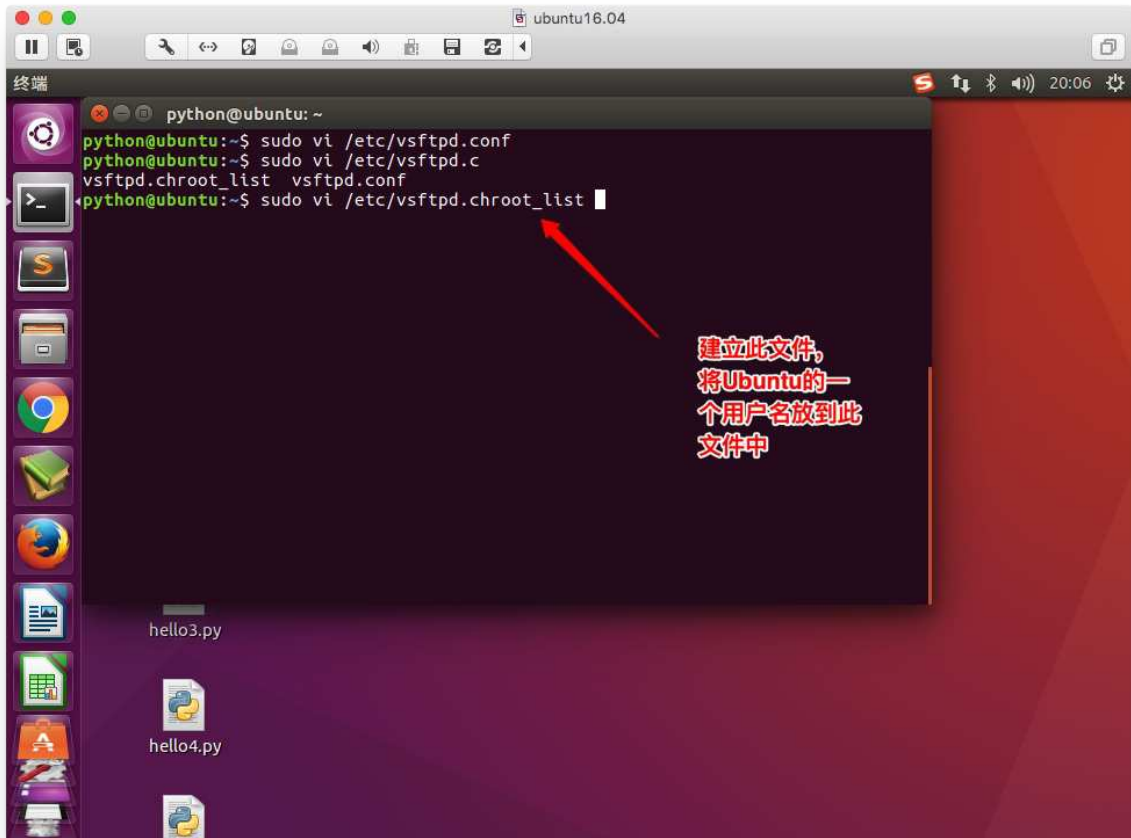


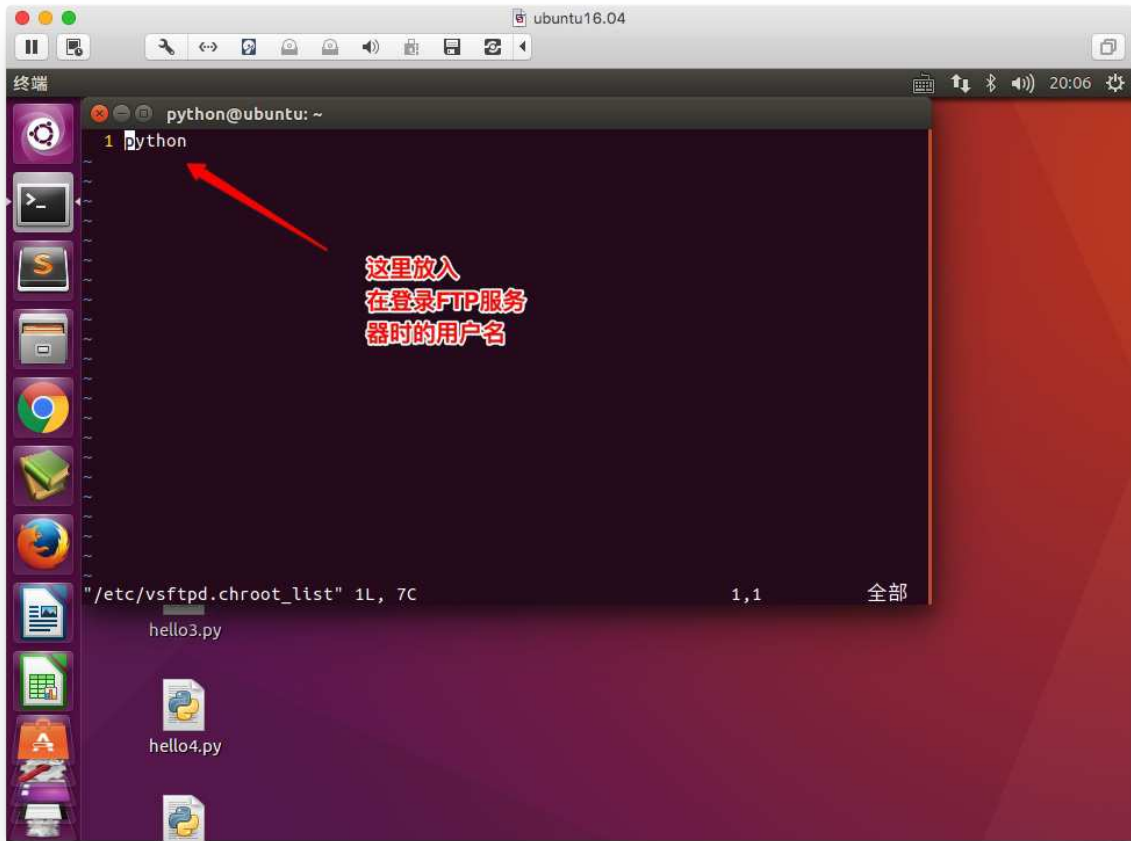


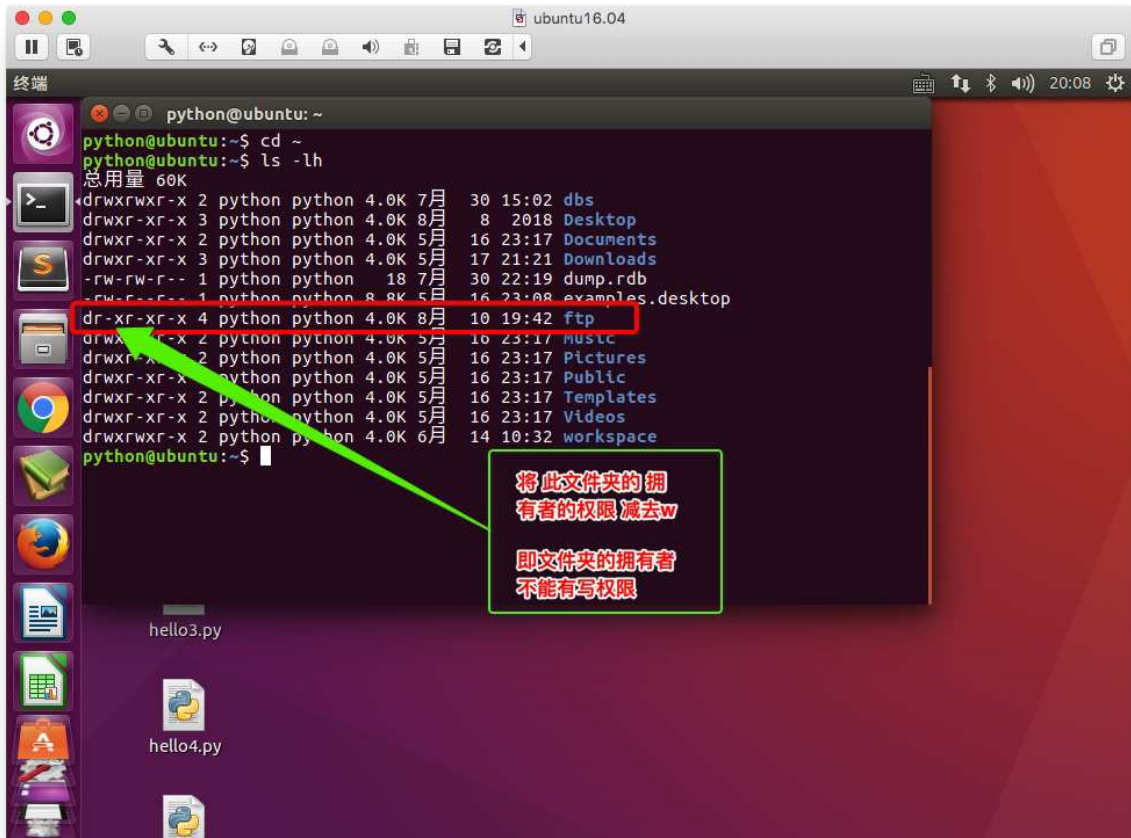


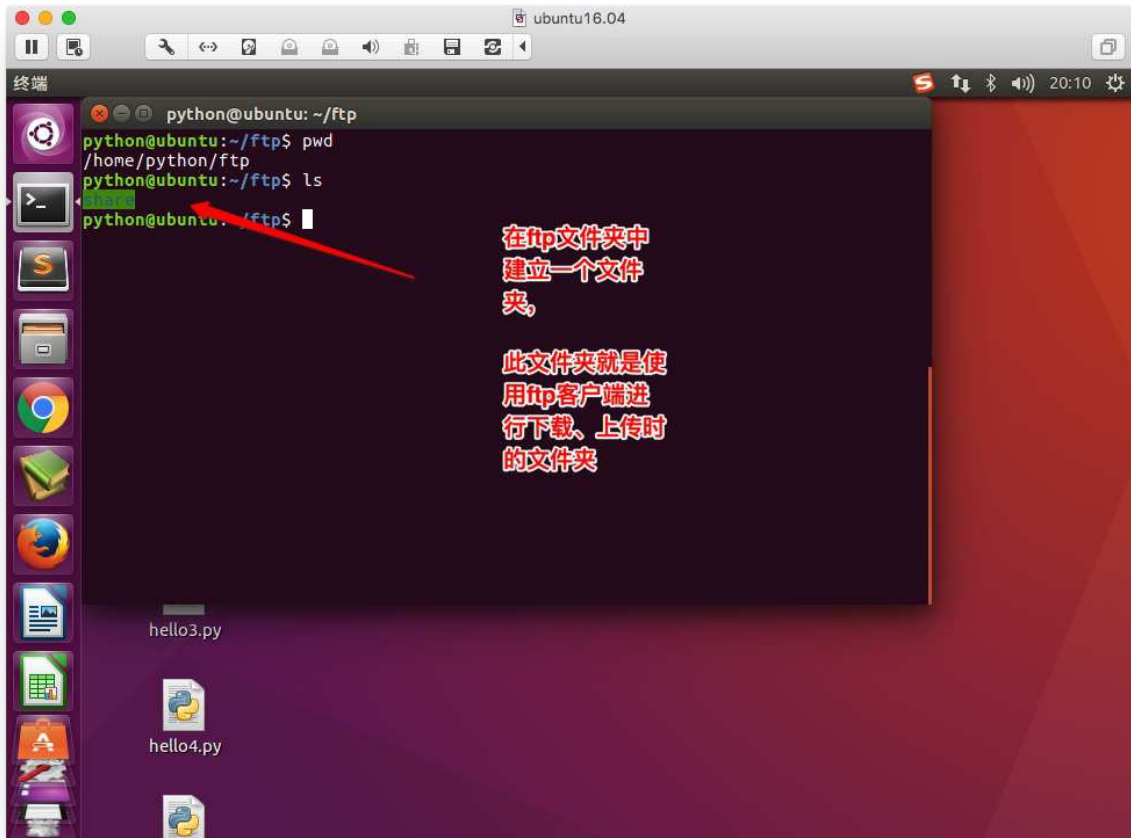


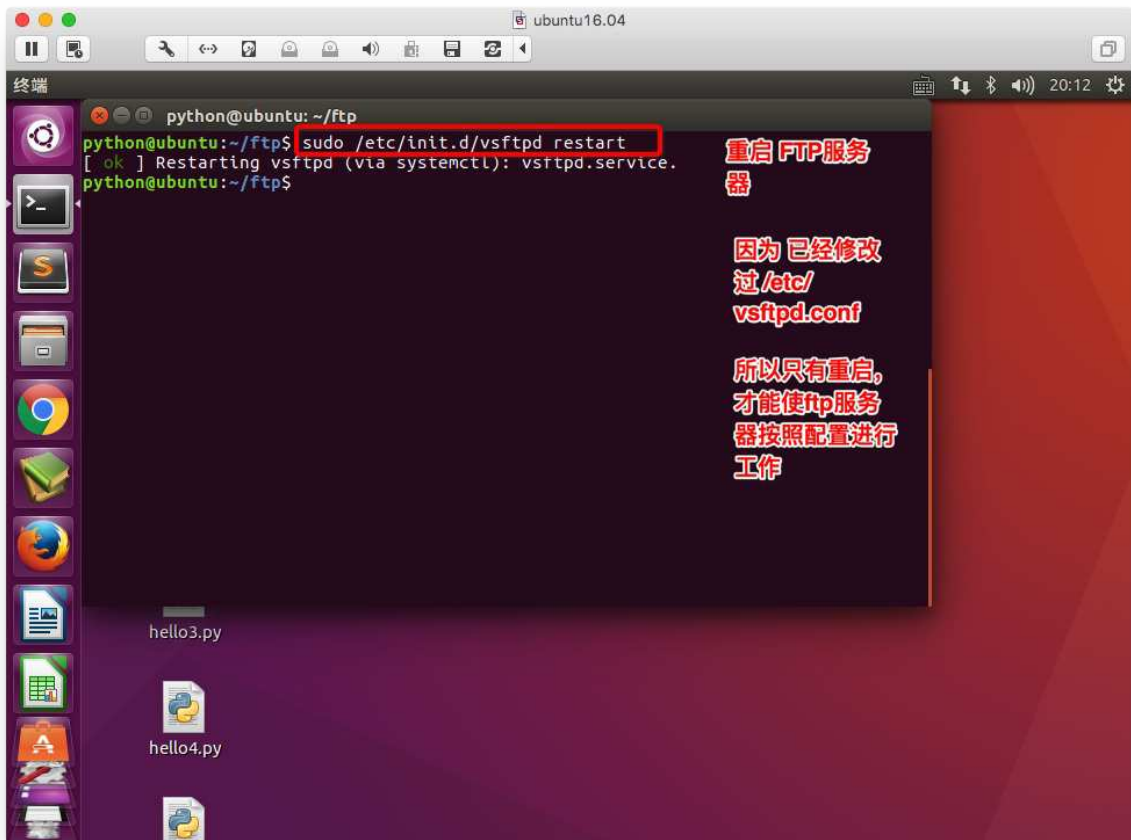












3.测试上传功能，登陆ftp服务器

```
ftp IP
```



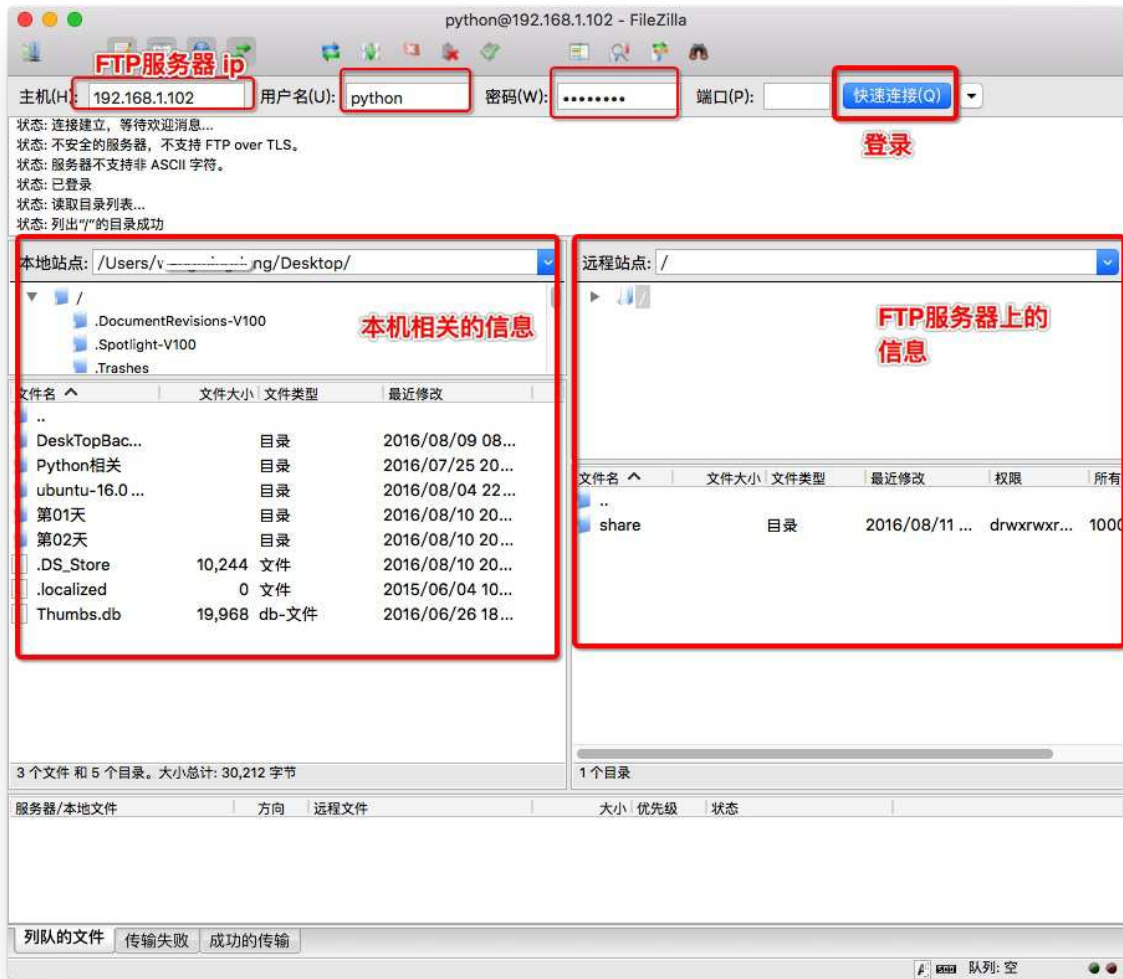
4.上传命令，可以把文件上传到ftp服务器

```
put somefile
```

5.下载命令，可以把ftp服务器上的文件下载到本地

```
get somefile
```

图形界面的ftp客户端(filezilla)



Linux常用服务器构建-ssh和scp

1.ssh

<1>ssh介绍

SSH为Secure Shell的缩写，由 IETF 的网络工作小组（Network Working Group）所制定；SSH 为建立在应用层和传输层基础上的安全协议。

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。常用于远程登录，以及用户之间进行资料拷贝。

利用SSH协议可以有效防止远程管理过程中的信息泄露问题。SSH最初是 UNIX 系统上的一个程序，后来又迅速扩展到其他操作平台。SSH 在正确使用时可弥补网络中的漏洞。SSH 客户端适用于多种平台。几乎所有 UNIX 平台—包括 HP-UX、Linux、AIX、Solaris、Digital UNIX、Irix，以及其他平台，都可运行SSH。

使用SSH服务，需要安装相应的服务器和客户端。客户端和服务器的关系：如果，A机器想被B机器远程控制，那么，A机器需要安装SSH服务器，B机器需要安装SSH客户端。

<2>安装ssh

A.安装ssh服务器

```
sudo apt-get install openssh-server
```

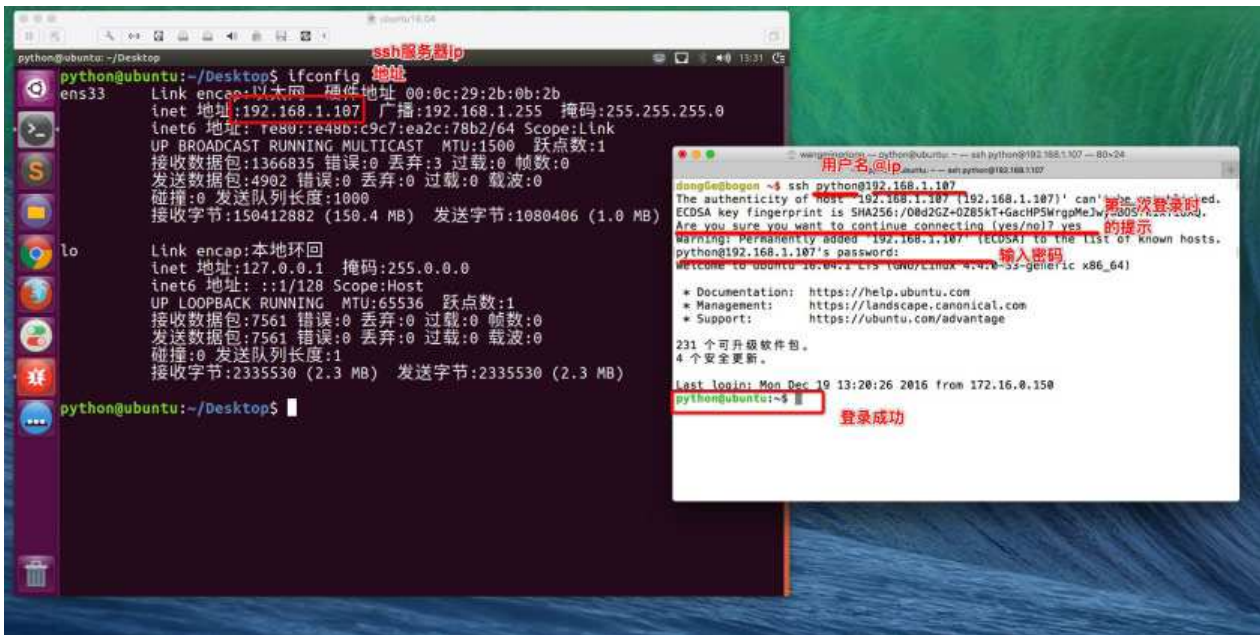
B.远程登陆

```
ssh 用户名@IP
```

使用ssh访问，如访问出现错误。可查看是否有该文件 `~/.ssh/known_ssh` 尝试删除该文件解决。

<3>使用ssh连接服务器

SSH 告知用户，这个主机不能识别，这时键入"yes"，SSH 就会将相关信息，写入"~/.ssh/know_hosts"中，再次访问，就不会有这些信息了。然后输入完口令，就可以登录到主机了。



2.scp

远程拷贝文件,scp -r 的常用方法:

1.使用该命令的前提条件要求目标主机已经成功安装openssh-server

如没有安装使用 `sudo apt-get install openssh-server` 来安装

2.使用格式:

`scp -r 目标用户名@目标主机IP地址: /目标文件的绝对路径 /保存到本机的绝对/相对路径`

举例:

`scp -r itcast@192.168.1.100:/home/itcast/qq_dir/ ./mytest/lisi`

在后续会提示输入"yes"此时，只能输"yes"而不能简单输入"Y"

拷贝单个文件可以不加 -r参数，拷贝目录必须要加。

本地文件复制到远程:

```
scp FileName RemoteUserName@RemoteHostIp:RemoteFile
scp FileName RemoteHostIp:RemoteFolder
```



```
scp FileName RemoteHostIp:RemoteFile
```

本地目录复制到远程:

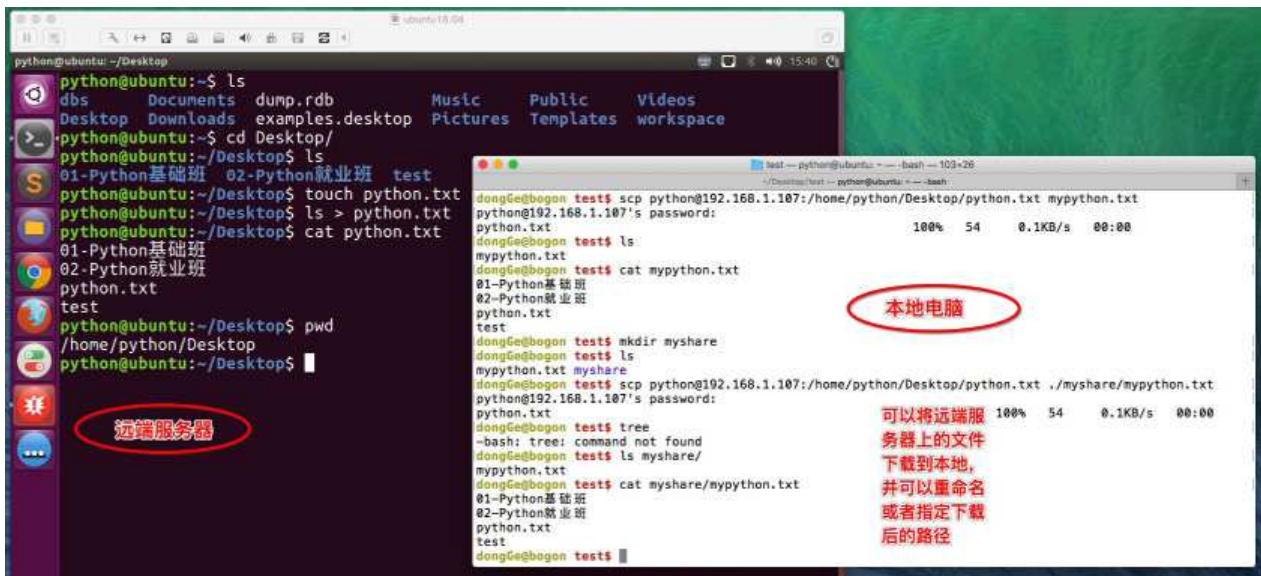
```
scp -r FolderName RemoteUserName@RemoteHostIp:RemoteFolder  
scp -r FolderName RemoteHostIp:RemoteFolder
```

远程文件复制到本地:

```
scp RemoteUserName@RemoteHostIp:RemoteFile FileName  
scp RemoteHostIp:RemoteFolder FileName  
scp RemoteHostIp:RemoteFile FileName
```

远程目录复制到本地:

```
scp -r RemoteUserName@RemoteHostIp:RemoteFolder FolderName  
scp -r RemoteHostIp:RemoteFolder FolderName
```



Linux常用服务器构建-samba

1. 介绍

Samba是在Linux和UNIX系统上实现SMB协议的一个免费软件，能够完成在windows、mac操作系统下访问linux系统下的共享文件

2. 安装

使用apt命令安装samba

```
python@ubuntu:~$ sudo apt-get install samba samba-common
[sudo] python 的密码:
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
将会同时安装下列软件：
  attr python-crypto python-dnspython python-ldb python-samba python-tdb s
amba-common-bin
  samba-dsdb-modules samba-vfs-modules tdb-tools
建议安装：
  python-crypto-dbg python-crypto-doc bind9 bind9utils ctdb ldb-tools ntp
smldap-tools winbind
  heimdal-clients
下列【新】软件包将被安装：
  attr python-crypto python-dnspython python-ldb python-samba python-tdb s
amba samba-common
  samba-common-bin samba-dsdb-modules samba-vfs-modules tdb-tools
升级了 0 个软件包，新安装了 12 个软件包，要卸载 0 个软件包，有 224 个软件
包未被升级。
需要下载 3,436 kB 的归档。
解压缩后会消耗 25.7 MB 的额外空间。
您希望继续执行吗？ [Y/n] y
获取:1 http://mirrors.aliyun.com/ubuntu xenial/main amd64 python-dnspython
  all 1.12.0-1 [85.2 kB]
获取:2 http://mirrors.aliyun.com/ubuntu xenial/main amd64 python-crypto am
d64 2.6.1-6build1 [245 kB]
```

3. 配置

3.1 创建存放共享文件的路径

在home路径下操作：

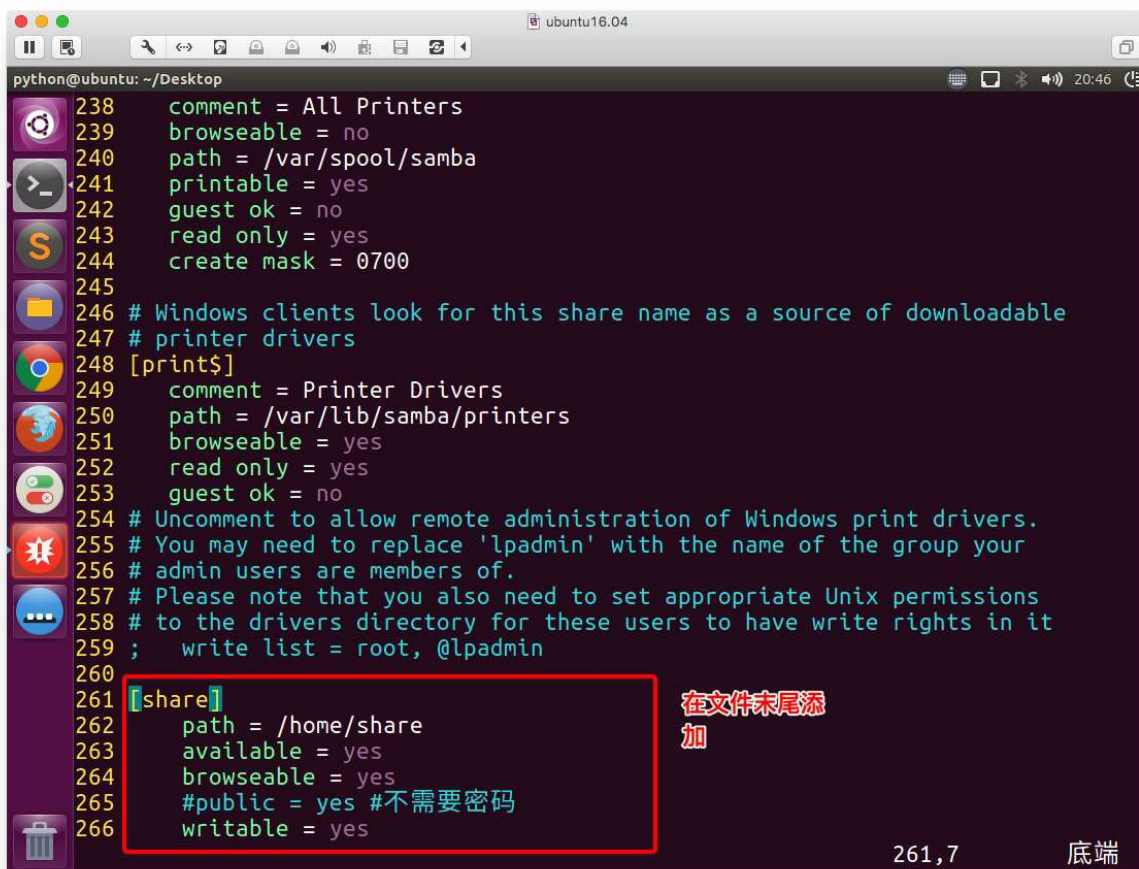
```
python@ubuntu:~/home$ sudo mkdir share  
python@ubuntu:~/home$
```

修改其权限：

```
root@ubuntu:~/home# chmod 777 share
```

修改samba的配置文件：

```
root@ubuntu:~/home# vi /etc/samba/smb.conf
```



```
python@ubuntu: ~/Desktop  
238 comment = All Printers  
239 browseable = no  
240 path = /var/spool/samba  
241 printable = yes  
242 guest ok = no  
243 read only = yes  
244 create mask = 0700  
245  
246 # Windows clients look for this share name as a source of downloadable  
247 # printer drivers  
248 [print$]  
249 comment = Printer Drivers  
250 path = /var/lib/samba/printers  
251 browseable = yes  
252 read only = yes  
253 guest ok = no  
254 # Uncomment to allow remote administration of Windows print drivers.  
255 # You may need to replace 'lpadmin' with the name of the group your  
256 # admin users are members of.  
257 # Please note that you also need to set appropriate Unix permissions  
258 # to the drivers directory for these users to have write rights in it  
259 ; write list = root, @lpadmin  
260  
261 [share]  
262 path = /home/share  
263 available = yes  
264 browseable = yes  
265 #public = yes #不需要密码  
266 writable = yes
```

在文件末尾添加

261,7 底端

3.2 创建samba账户

```
root@ubuntu:~/home/share# touch /etc/samba/smbpasswd
```

```
root@ubuntu:/home/share# smbpasswd -a python
New SMB password:
Retype new SMB password:
root@ubuntu:/home/share#
```

4 重启samba

当对配置进行了更新，需要重启samba软件后才可生效

重启命令如下：

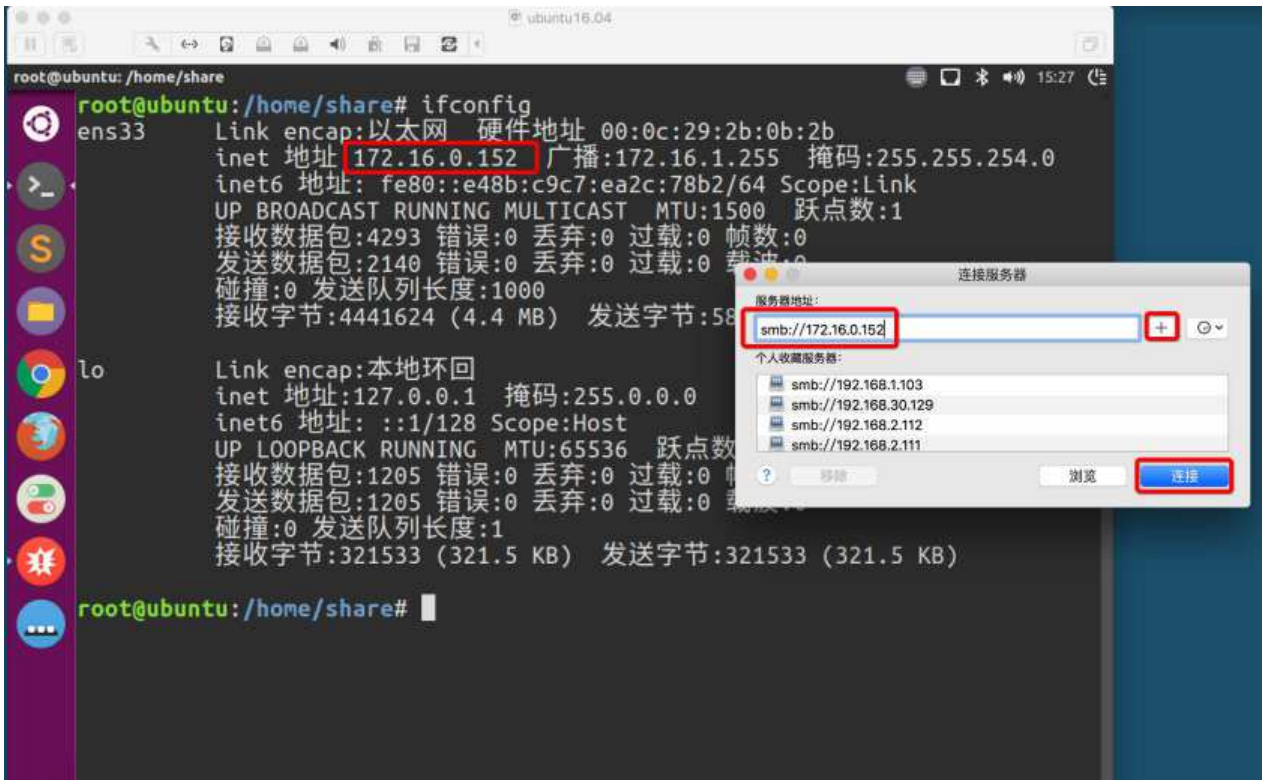
```
root@ubuntu:/home# /etc/init.d/smbd restart
```

重启启动

5. 访问共享文件

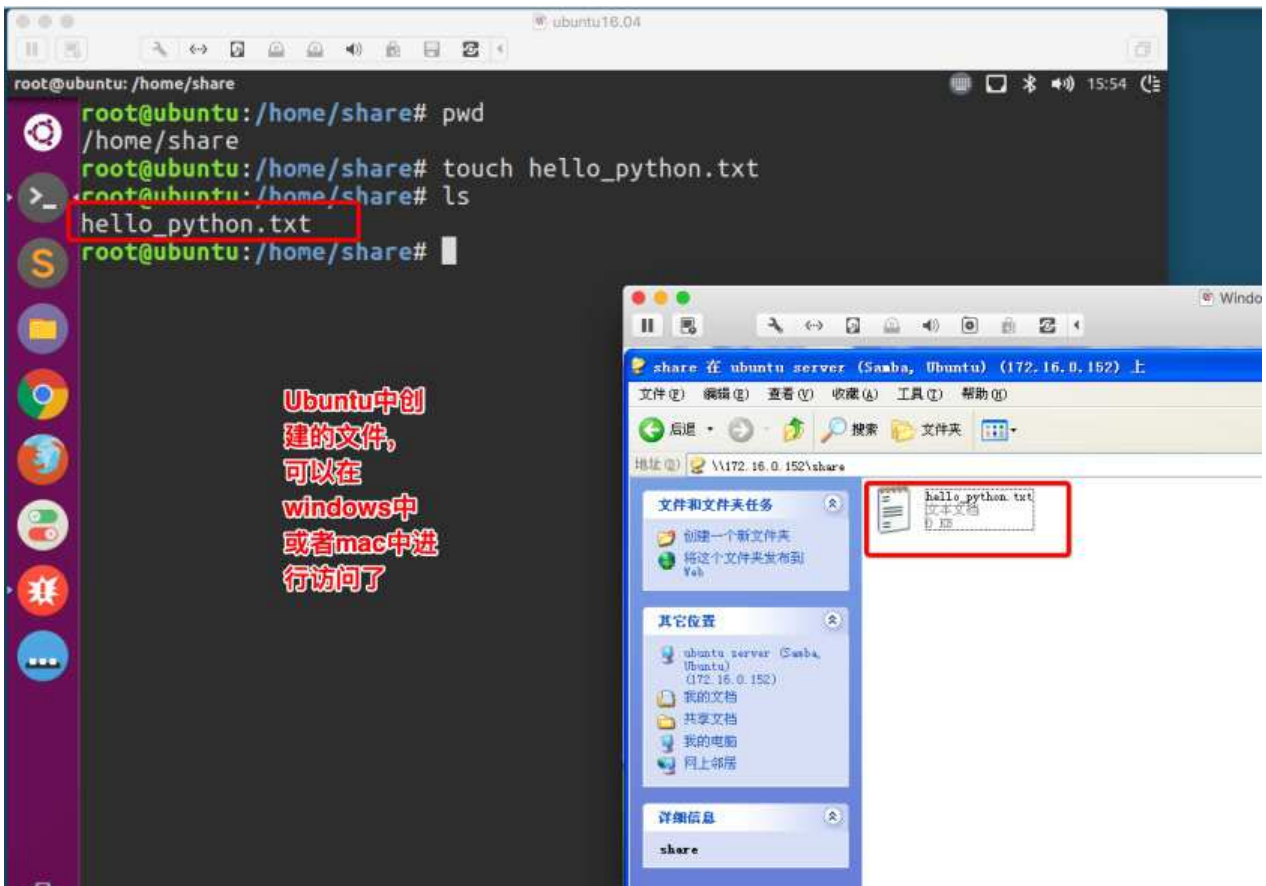
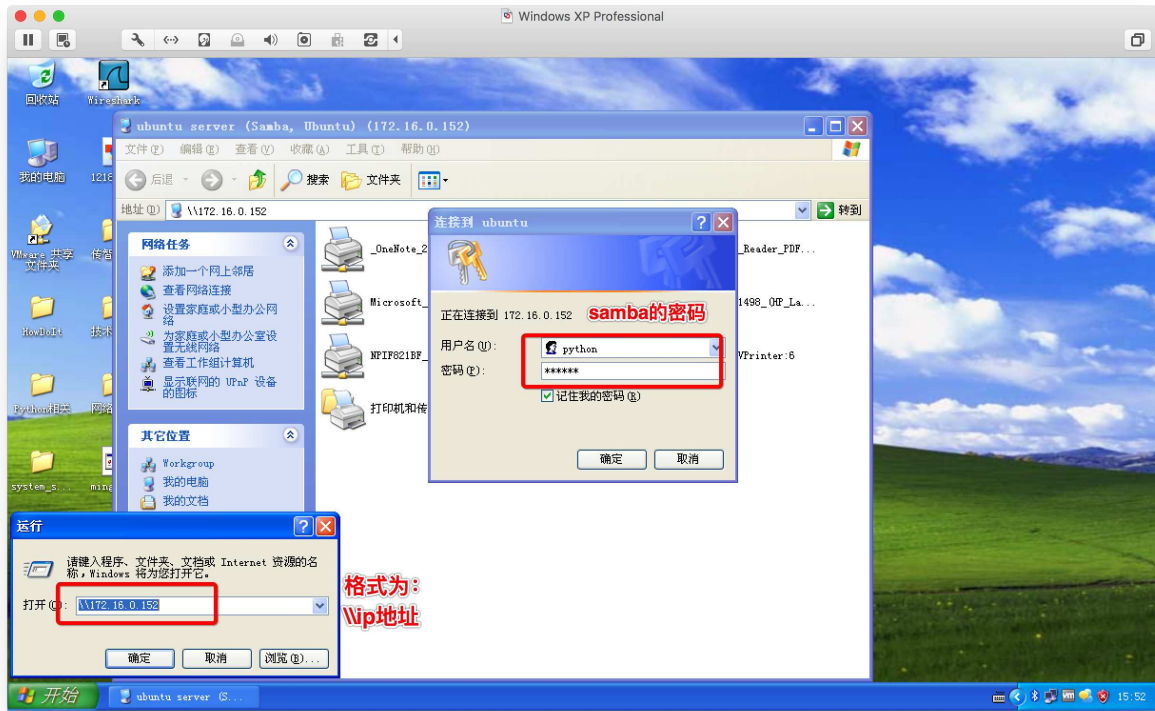
5.1 mac下访问方式







5.2 windows下访问方式



附录-vim分屏操作

vim分屏操作

分屏操作:

```
sp: 上下分屏,后可跟文件名
```

```
vsp: 左右分屏,后可跟文件名
```

```
Ctrl+w+w: 在多个窗口切换
```

启动分屏:

1.使用大写O参数进行垂直分屏

```
$ vim -On file1 file2 ...
```

2.使用小写o参数进行水平分屏

```
$ vim -on file1 file2 ...
```

注: n是数字, 表示分屏的数量,n要大于等于文件个数

关闭分屏

1.关闭当前窗口

```
ctrl+w c
```

2.关闭当前窗口, 如果只剩最后一个, 则退出vim

```
ctrl+w q
```

编辑中分屏

1.上下分割当前打开的文件


```
ctrl+w s
```

2.上下分割，并打开一个新的文件

```
:sp filename
```

3.左右分割当前打开的文件

```
ctrl+w v
```

4.左右分割，并打开一个新的文件

```
:vsp filename
```

分屏编辑中光标的移动

vi中的光标键是h,j,k,l,要在各个屏之间切换，只需要先按一下ctrl+w

1.把光标移动到上边的屏

```
ctrl+w k
```

2.把光标移动到下边的屏

```
ctrl+w j
```

3.把光标移动到右边的屏

```
ctrl+w l
```

4.把光标移动到左边的屏

```
ctrl+w h
```

5.把光标移动到下一个的屏

```
ctrl+w w
```

移动分屏

1.向上移动

```
ctrl+w K
```

2.向下移动

```
ctrl+w J
```

3.向右移动

```
ctrl+w L
```

4.向左移动

```
ctrl+w H
```

屏幕尺寸

1.增加高度

```
ctrl+w +
```

2.减少高度

```
ctrl+w -
```

3.让所有屏的高度一致

```
ctrl+w =
```

4.左加宽度

```
ctrl+w >
```

5.右加宽度

```
ctrl+w <
```

6.右增加n宽 (如: n=30)

```
ctrl+w n <
```

vim打造IDE

简洁版IDE

C+p: 生成tags

C+] : 跳转到函数定义

C+t : 从函数定义返回

C+o : 在左侧打开文件列表

F4 : 在右侧打开函数列表

C+n : 补齐函数, 向下翻

vimrc是vim的配置文件, 可以修改两个位置

```
1. /etc/vim/vimrc
```

```
2. ~/.vimrc
```

~/.vimrc优先级高