# Shell和vi

## 什么是shell

在计算机科学中，Shell俗称壳，用来区别于Kernel（核），是指“提供使用者使用界面”的软件（命令解析器）。它类似于[DOS](http://baike.baidu.com/view/365.htm" \t "_blank)下的command和后来的cmd.exe。它接收用户命令，然后调用相应的应用程序。

## shell分类

1、图形界面shell：通过提供友好的可视化界面，调用相应应用程序，如windows系列操作系统，Linux系统上的图形化应用程序GNOME、KDE等。

2、命令行shell：通过键盘输入特定命令的方式，调用相应的应用程序，如windows系统的cmd.exe、Windows PowerShell，Linux系统的Bourne shell ( sh)、Bourne Again shell ( bash)等。

## 认识bash这个shell

在window系统下使用bash，需要一个软件，这个软件模拟集成了bash大部分命令。

各个 shell 的功能都差不多， Linux 默认使用 bash ，所以我们主要学习bash的使用。

**1、bash命令格式**

命令 [ options] [参数]

查看帮助：命令 --help

**2、bash常见命令**

pwd ---(Print Working Directory) 查看当前目录

cd .. 退出当前目录

cd ---(Change Directory) 切换目录，如 cd /etc

ls --- (List) 查看当前目录下内容，如 ls -al

mkdir--- (Make Directory) 创建目录，如 mkdir blog

touch 创建文件，如 touch index.html

cat 查看文件全部内容，如 cat index.html

less 查看文件，如more /etc/passwd、less /etc/passwd

rm (remove) 删除文件，如 rm index.html、rm -rf blog

rmdir (Remove Directory) 删除文件夹，只能删除空文件夹，不常用

mv (move) 移动文件或重命名，如 mv index.html 🡪 . /demo/index.html

cp (copy) 复制文件，cp index.html 🡪 . /demo/index.html

head 查看文件前几行，如 head -5 index.html

tail 查看文件后几行 –n –f，如 tail index.html、tail -f -n 5 index.html

tab 自动补全，连按两次会将所有匹配内容显示出来

history 查看操作历史

> 和 >>重定向，如echo hello world! > README.md，>覆盖， >>追加

wget 下载，如wget https://nodejs.org/dist/v4.4.0/node-v4.4.0.tar.gz

tar 解压缩，如tar zxvf node-v4.4.0.tar.gz

curl 网络请求，如 curl http://www.baidu.com

whoami 查看当前用户

| 管道符可以将多个命令连接使用，上一次（命令）的执行结果当成下一次（命令）的参数。

grep 匹配内容，一般结合管道符使用 grep s index.txt --s为正则表达式，匹配文件中的内容

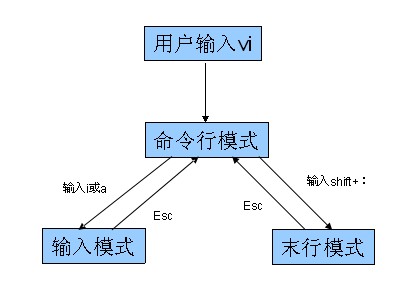
## vi编辑器

如同Windows下的记事本，vi编辑器是Linux下的标配，通过它我们可以创建、编辑文件。它是一个随系统一起安装的文本编辑软件。

**1、三种模式**

vi编辑器提供了3种模式，分别是命令模式、插入模式、底行模式，每种模式下用户所能进行的操作是不一样的。

3种模式的切换如下图所示：



通过上图我们发现，输入模式是不能直接切换到末行模式的，必须要先切回到命令模式（按ESC键）

**2、使用vi编辑器**

a) 打开/创建文件， vi 文件路径

b) 底行模式 :w保存，:w filenme另存为

c) 底行模式 :q退出

d) 底行模式 :wq保存并退出

e) 底行模式 :e! 撤销更改，返回到上一次保存的状态

f) 底行模式 :q! 不保存强制退出

g) 底行模式 :set nu 设置行号

h) 命令模式 ZZ（大写）保存并退出

i) 命令模式 u辙销操作，可多次使用

j) 命令模式 dd删除当前行

k) 命令模式 yy复制当前行

l) 命令模式 p 粘贴内容

m) 命令模式 ctrl+f向前翻页

n) 命令模式 ctrl+b向后翻页

o) 命令模式 i进入编辑模式，当前光标处插入

p) 命令模式 a进入编辑模式，当前光标后插入

q) 命令模式 A进入编辑模式，光标移动到行尾

r) 命令模式 o进入编辑模式，当前行下面插入新行

s) 命令模式 O进入编辑模式，当前行上面插入新行

当我们处在编辑模式的情况下，和我们在Windows编辑器的使用相似。

## SSH

SSH是一种网络协议，用于计算机之间的加密登录。

SSH只是一种协议，存在多种实现，既有商业实现，也有开源实现。本教程针对的是[OpenSSH](http://www.openssh.com/" \t "_blank)，它是自由软件，应用非常广泛。

如果要在Windows系统中使用SSH，会用到另一种软件[PuTTY](http://www.chiark.greenend.org.uk/~sgtatham/putty" \t "_blank)，我们后面用到的Git客户也集成了SSH

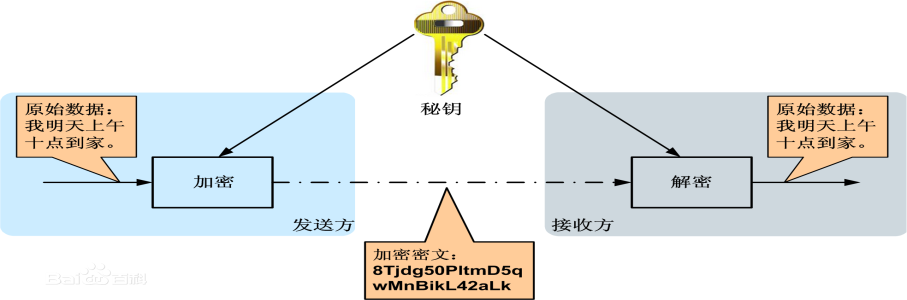
格式：ssh user@host

user 代表真实存在的用户

host代表要登录的远程计算机

常见有两种加密技术，分别是对称性加密和非对称性加密，SSH属于后者。

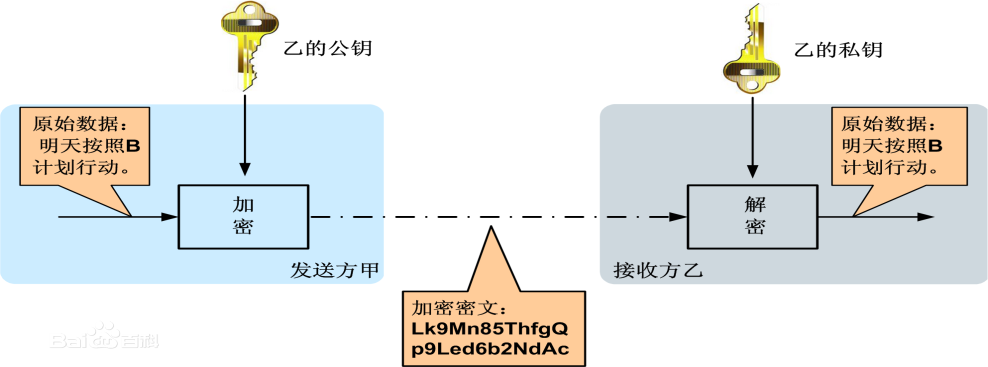
对称加密[算法](http://baike.baidu.com/view/7420.htm" \t "_blank)在加密和解密时使用的是同一个密钥；



而[非对称加密算法](http://baike.baidu.com/view/1490349.htm" \t "_blank)需要两个[密钥](http://baike.baidu.com/view/934.htm" \t "_blank)来进行加密和解密，这两个秘钥分别是[公开密钥](http://baike.baidu.com/view/1145160.htm" \t "_blank)（public key，简称公钥）和私有密钥（private key，简称私钥）。

**工作原理**

公钥和私钥是成对出现，可以通过ssh-keygen -t rsa来创建，既可以通过密钥来加密数据，也可以通过私钥来加密数据，如果是以公钥进行的数据加密，只能与之相对应的私钥才可以解密，相反如果以私钥进行的数据加密，则只能与之对应的公钥才可以将数据进行解密，这样就可以提高信息传递的安全性。



**免密码登录**

我们可以将本地机器上的公钥保存到特定的远程计算机上，这样当我们再次登录访问这台远程计算机时就可以实现免密码登录了。

1、ssh-keygen -t rsa 会创建公钥和密钥（默认在用户目录/.ssh目录下）

2、ssh-copy-id user@host 添加到对应远程主机的用户目录/.ssh目录下

3、也可以登录远程主机，进入到用户目录/.ssh目录下手动创建authorized\_keys文件，并将自已的公钥粘入该文件。

这部分具体实现细节，参照我的演示有个印象就可以了。

# 版本控制

## 关于版本控制

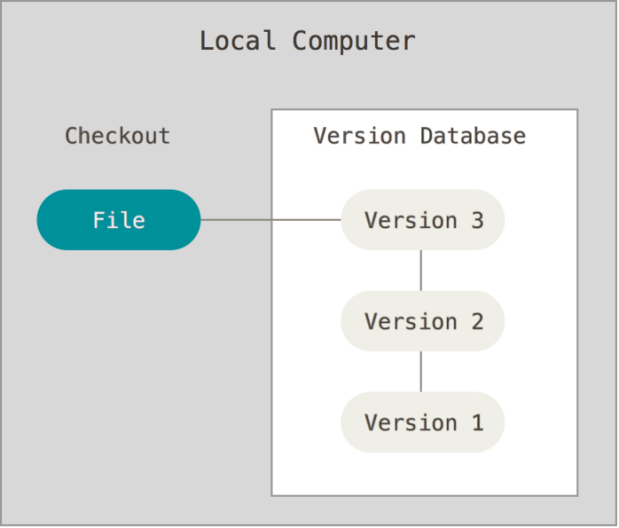
版本控制（Version Control Systems）是一种记录一个或若干文件内容变化，以便将来查阅特定版本修订情况的系统。这个系统可以自动帮我们备份文件的每一次更改，并且可以非常方便的恢复到任意的备份（版本）状态。

举例：我们通常都是手动的重命名一个文件进行备份的，index.html改成index1.html或者index.html.bak等形式，然后这种方式对于单个文件我们还能够管理，但是对于整个项目而言，就会成为噩梦了！！！我们不得不借助于软件来实现。

实现版本控制的软件有很多种类，大致可以分为本地版本控制系统、集中式版本控制系统、分布式版本控制系统。

## 本地版本控制系统

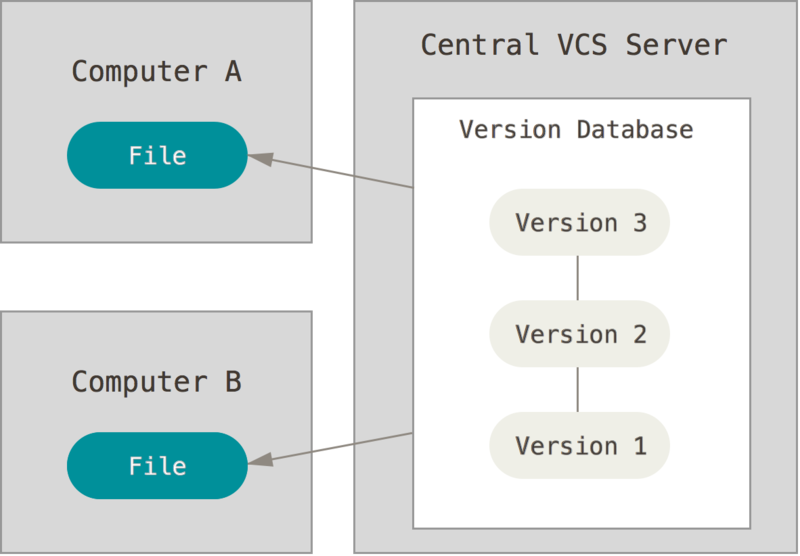
借助软件我们可以记录下文件的每一次修改，如下图所示，文件被修改后，记录下了3个版本，这样我们通过版本控制系统（软件）便可以非常方便的恢复到任意版本。



这种类型的版本控制系统，功能比较单一，比如很难实现多人协同开发，所以现在几乎很少使用了。

## 集中式版本控制系统

实际开发环境，一个项目通常是由多人协作共同完成的，如何让在不同终端上的开发者协同工作成了亟待解决的问题，集中式版本控制系统便应运而生了。它通过单一的集中管理的服务器，保存所有文件的修订版本，协同工作的开发者都通过客户端连到这台服务器，取出最新的文件或者提交更新。其代表为SVN，如下图所示。

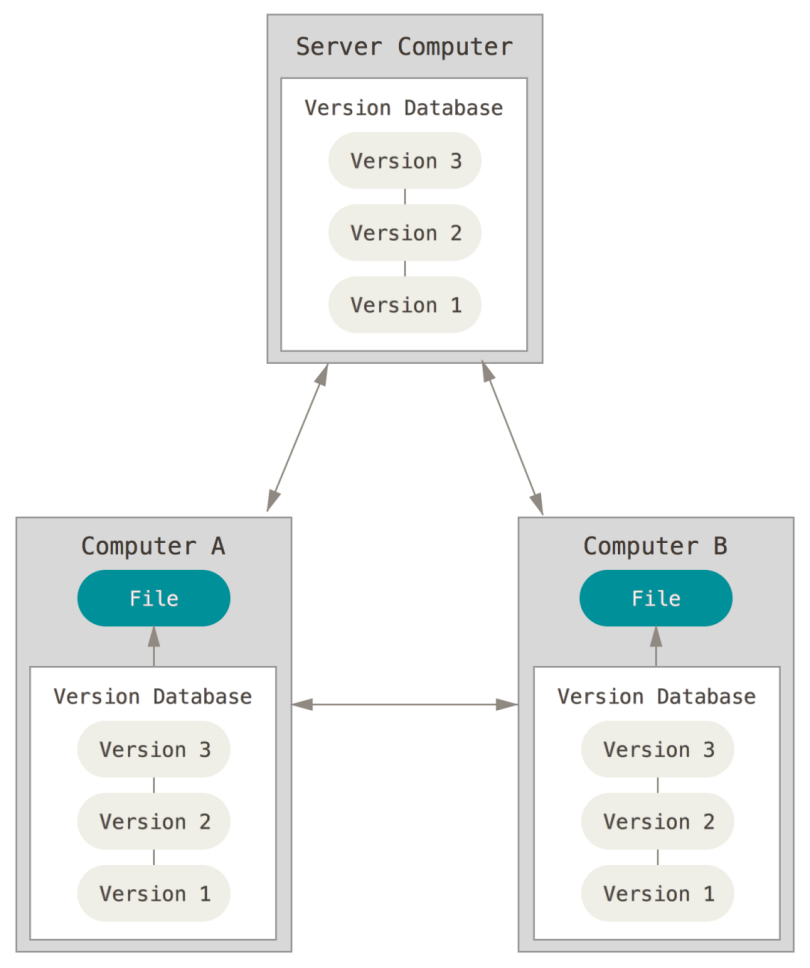


这种方式很好解决了多人协同开发的问题，但是也有一个弊端，如果集中管理的服务器出现故障，将会导致数据（版本）丢失的风险，另外协同开发者从集中服务器中更新数据时，严重依赖网络，如果网络不佳，也给开发带来诸多不便。

## 分布式版本控制系统

分布式版本控制系统，则不需要中央服务器，每个协同开发者都拥有一个完整的版本库，这么一来，任何协同开发者用的服务器发生故障，事后都可以用其它协同开发者本地仓库恢复。

由于版本库在本地计算机，也便不再受网络影响了。如果要将本地的修改，推送给其它协同开发者，还需要一台共享服务器，所有开发者通过这台共享服务器同步和更新数据。如下图所示。



分布式版本控制系统弥补了前面两种版本控制系统的缺陷，成为了版本控制的首选方案。其代表就是Git。