

*Linux in 3 days*

第2版



# Linux 快速掌握

自学 it 网®

燕十八 著

<http://www.zixue.it>

---

第一章 虚拟机及 Linux 安装.....	3
1.1 虚拟机介绍.....	3
1.2 创建虚拟机.....	3
1.3 装操作系统.....	5
第二章 安装与开关机.....	6
第三章: Linux 的基本命令.....	7
第四篇 文件的常用命令.....	9
4.1 文件文件查看.....	9
4.2 文件打包.....	10
4.3 文件查找.....	11
第五章 Linux 的系统命令.....	14
第六章 文件挂载.....	15
第七章 vi 编辑器.....	16
第八章 网络配置.....	19
第九章 远程管理.....	22
第十章 linux 的用户管理.....	24
第十一章 权限管理.....	26
第十二章 rpm 软件管理包.....	28
第十二章 yum 软件管理器.....	32
第十三章 软件编译.....	35
第十四章 编译 lnmp.....	37
第十五章 shell 编程(选学内容).....	43
第十六章 定时任务.....	50
第十七章 数据库定期备份实例.....	51

# 第一章 虚拟机及 Linux 安装

## 1.1 虚拟机介绍

通过软件模拟一台物理机. 常用的 vmware,virtualbox.

<http://pan.baidu.com/s/1xMex9>


## 1.2 创建虚拟机



### 虚拟电脑名称和系统类型

为即将新建的虚拟电脑输入一个名称，并指定虚拟电脑上即将安装的操作系统类型。

每个虚拟电脑都要有一个唯一的名称来标识，用来区分该虚拟电脑的硬件配置和上面的系统、文件和数据。

名称 (a)	<input type="text" value="test"/> 创建一台适合64位linux的裸机, 并没有操作系统	
系统类型 (T)		
操作系统 (S):	<input type="text" value="Linux"/>	
版本 (V):	<input type="text" value="Red Hat (64 bit)"/>	



## 内存

根据自己的电脑内存调整大小, 1/4以下即可

指定虚拟电脑可用内存大小, 单位为: MB .

建议分配的内存大小是 512 MB.

内存大小 (M)



## 虚拟硬盘

在你的物理硬盘上, 创建一个文件, 来模拟虚拟机的硬盘, 所以, 挑剩余空间大一点的盘来安装虚拟机

Select a virtual hard disk to be used as the boot hard disk of the virtual machine. You can either create a new hard disk or select an existing one from the drop-down list or by pressing corresponding button (to invoke file-open window).

If you need a more complicated hard disk setup, you can also skip this step and attach hard disks later using the VM Settings dialog.

The recommended size of the boot hard disk is 8.00 GB.

☒ Boot Hard Disk

☒ 创建新的虚拟硬盘 (C)

☐ 使用现有的虚拟硬盘 (U)

centos6.4-64bit.vdi (Normal, 8.00 GB)





创建完毕后,多出一台虚拟机



## 1.3 装操作系统

到 [http://isoredirect.centos.org/centos/6/isos/x86\\_64/](http://isoredirect.centos.org/centos/6/isos/x86_64/) 下载 最新 ISO 镜像  
(32 位操作系统请下载 32 位 iso),  
把 iso 模拟成虚拟机的光盘

## 第二章 安装与开关机

### Linux 安装与开关机

1:linux 安装有哪两种分区方式?

答:手动分区和自动分区

2:Linux 安装过程中,若选择手动分区时,必需配置的 2 个分区分别是什么?

答:根分区(/)与交换分区.swap[注 1:内存大不要 swap 也可以]

3:命令行如何关机?

服务器一般不关机,虚拟机也不用关,点击"休眠",保持其运行状态.

在终端输入 shutdown 或 halt 或 poweroff

命令行如何重启 Linux?

在终端输入 reboot

Linux 根用户的用户名是什么?

root

4: 图形界面下如何进入字符界面?

右键->终端->init 3

或编辑/etc/inittab,让启动级别为 3

注意事项:安装虚拟机时候,一定要把杀毒软件全部关闭

## 第三章: Linux 的基本命令

### 1:关于位置的探讨

在 linux 下,有两种定位方式,绝对定位,与相对定位.

绝对定位是从根目录(/)开始算,一层一层的定位.

相对定位是以当前所在的目录开始算,通过../表示上一层目录,./表示当前目录.

绝对定位举例:

如 `cd /usr/local/bin`,此时将进入到 `/usr/local/bin` 下.

相对定位举例:

接上一步,在已经进入到`/usr/local/bin`的情况下.

如何到`/usr/local/share` 文件夹呢?

可以这样, `cd ../share`

绝对定位与相对定位还可以结合起来使用:

如: `cd /usr/local/bin/../../` 回车之后,将进入 `/usr/local`

文件/目录操作 : 创建,查看,移动,改名,删除,复制  
用户/组管理: 创建组/用户,删除组/用户  
权限管理: 查看/修改权限  
进程管理: 查进程,杀进程  
打包解压: .gz .bz这样的压缩文件操作  
软件安装: yum安装 编译安装  
编辑器: vim  
网络配置

实战达标:  
安装Linux,配置上网,配置lnmp环境  
(nginx+mysql+php)

提升要求:  
shell 定时任务,定期备份

### 2:目录/文件的进入(查看),创建,复制,移动,改名与删除

进入目录:

用法 `cd 目录路径`

代码示例:

```
cd /var/tmp
```

```
cd ../
```

```
cd /var/tmp/../../ (结合上面讲的"绝对路径与相对路径的结合用法",思考,此时进入到哪儿了?)
```

创建 1 个或多个目录 `mkdir`

用法: `mkdir dir1 [dir2] [dir3] [dirn]`

代码示例:

```
cd ~ (先进入到自己的家目录)
```

```
mkdir dir1
```

```
ls 查看结果?
```

```
mkdir dir2
```

```
ls 查看结果?
```

```
mkdir dir3 dir4
```

```
ls 查看结果?
```

```
mkdir dir{5,6,7}
```

```
mkdir -p /d1/d2/d3 创建级联目录
```

`ls` 列出当前目录的文件

---

`ls /path` 列出/path 下的文件

`ls /path -l` 详细列出 path 的文件

移动命令(或移动文件夹或者文件) `mv`

用法 `mv` 源文件夹/文件 目标文件夹/文件

示例: `mv ./dir1 ./dir2`

效果:ls 一下,看看 此时还能找到 `dir1` 吗,再 `ls dir2` 一下看看.

复制 `copy`, 命令 `cp`

`cp` 源文件 目标文件

`cp` 源目录 目标目录 `-R` [递归 `copy` 整个目录]

如何改名? 比如先 `touch` 一个 `a.txt`, 改名成 `a.txt.bak`

还是用 `mv` 命令.

在 `mv` 时, `mv` 源文件名 新文件名

`rmdir` 删除空文件夹 (如果文件夹非空,不能用这个命令)

用法 `rmdir` 空文件夹名称

示例 `rmdir ./dir2/dir1`

再 `ls ./dir2` 看一看,`dir1` 还有没有?

`touch` 创建一个空文件.

`Touch` 文件名称

示例: `touch tmp.txt`

`rm` 删除一个文件或文件夹

用法 `rm` 文件名称

`rm -rf` 文件夹名称 (`-r` 表示循环迭代的意思,这样他碰到子目录就进去删,`-f` 是强制的意思)

示例: `rm tmp.txt`

`rm -rf ./dir1`

`pwd` 显示当前所在位置

示例: `pwd` (不用加参数,将会显示你当前所在的位置)

`ls` 显示当前目录的文件夹及文件

用法 `ls -a/-l` (参数可选)

`-a` 表示把隐藏文件也显示出来

`-l` 表示显示详细信息 创建日期/大小/权限/属主等都显示.



ln 链接命令(相当于 windows 下面的快捷方式)

用法 ln -s 源文件夹或源文件 目标链接名称

举例

```
ln -s /var/tmp ./linkvar
```

```
cd ./linkvar
```

pwd 一下,看看自己是在哪儿?

## 第四篇 文件的常用命令

### 4.1 文本文件查看

> 重定向(覆盖原文件)

>> 重定向(追加原文件)

查看 more,less,head,tail,cat,以及 grep

more 查看文件的内容

用法 more filename

less 作用/用法同 more

[more 最后一屏时自动退出,less 不自动退出,按 q 退出,也可以 ctrl+b 往前翻,ctrl+f 往后翻]

head 查看文件的前几行

用法 head -3 filename (不一定是-3 哦,只是举个例子,查看前 3 行)

tail 查看文件的后几行

用法 tail -5 filename (也不一定是-5,只是举个例子,查看后 5 行)

cat 把文件的内容连接起来打印到终端或者用 > 覆盖到另一个文件

用法 cat 文件 1 文件 2 ...文件 n (直接把 n 个文件的内容连接起来输出在屏幕上)

cat 文件 1 文件 2 ...文件 n > 文件 n+1(把 n 个文件连接成一个新文件)

示例 echo aa > aa.txt

echo bb > bb.txt

cat aa.txt bb.txt > cc.txt

more cc.txt (看一下结果)

grep 匹配文件中的行

grep 判断字 文件

grep adm /etc/passwd ,会把/etc/passwd 文件中含有 adm 的行打印出来

---

more a.txt|grep china, 把本来要输出 a.txt 的内容,交给 grep 再匹配,匹配含有 china 的行.

## 4.2 文件打包压缩及解压

.tar

解包: tar xvf FileName.tar

打包: tar cvf FileName.tar DirName

(注: tar 是打包, 不是压缩!)

注:如果不想看打包的具体过程,可以用 tar cf filename.tar dirname;

-----  
.gz

解压 1: gunzip FileName.gz

解压 2: gzip -d FileName.gz

压缩: gzip FileName

压缩且不删除源文件: gzip -c FileName > FileName.gz

.tar.gz

解压: tar zxvf FileName.tar.gz

压缩: tar zcvf FileName.tar.gz DirName

-----  
.bz2

解压 1: bzip2 -d FileName.bz2

解压 2: bunzip2 FileName.bz2

压缩: bzip2 -z FileName

.tar.bz2

解压: tar jxvf FileName.tar.bz2

压缩: tar jcvf FileName.tar.bz2 DirName

-----  
.bz

解压 1: bzip2 -d FileName.bz

解压 2: bunzip2 FileName.bz

压缩: 未知

.tar.bz

解压: tar jxvf FileName.tar.bz

压缩: 未知

-----  
.Z

解压: uncompress FileName.Z

压缩: compress FileName

.tar.Z

解压: tar Zxvf FileName.tar.Z

压缩: tar Zcvf FileName.tar.Z DirName

-----  
.tgz

解压: tar zxvf FileName.tgz

压缩: 未知

.tar.tgz

解压: tar zxvf FileName.tar.tgz

压缩: tar zcvf FileName.tar.tgz FileName

-----  
.zip

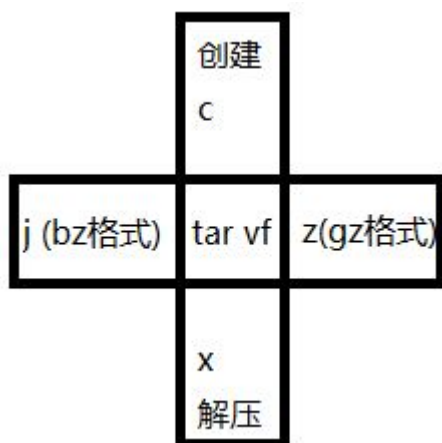
解压: unzip FileName.zip

压缩: zip FileName.zip DirName

-----  
.rar

解压: rar a FileName.rar

压缩: rar e FileName.rar



rar 请到: <http://www.rarsoft.com/download.htm> 下载!

解压后请将 rar\_static 拷贝到/usr/bin 目录 (其他由\$PATH 环境变量指定的目录也可以)

## 4.3 文件查找

```
find / -name httpd.conf
```

这个命令语法看起来很容易就明白了, 就是直接在 find 后面写上 -name, 表明要求系统按照文件名查找, 最后写上 httpd.conf 这个目标文件名即可。稍等一会系统会在计算机屏幕上显示出查找结果列表:

```
etc/httpd/conf/httpd.conf
```

这就是 httpd.conf 这个文件在 Linux 系统中的完整路径。查找成功。

如果输入以上查找命令后系统并没有显示出结果, 那么不要以为系统没有执行 find/ -name httpd.conf 命令, 而可能是你的系统中没有安装 Apache 服务器, 这时只要你安装了 Apache Web 服务器, 然后再使用 find / -name httpd.conf 就能找到这个配置文件了。

无错误查找技巧：

在 Linux 系统中普通用户使用“find”命令来查询这些文件目录是，往往会出现"Permissiondenied."（禁止访问）字样。系统将无法查询到你想要的文件。为了避免这样的错误，我们可是使用转移错误提示的方法尝试着查找文件，输入

```
find / -name access_log 2>/dev/null
```

根据部分文件名查找方法：

这个方法和在 WINDOWS 中查找已知的文件名方法是一样的。不过在 Linux 中根据部分文件名查找文件的方法要比在 WINDOWS 中的同类查找方法要强大得多。例如我们知道某个文件包含有 srm 这 3 个字母，那么要找到系统中所有包含有这 3 个字母的文件是可以实现的，输入：

```
find /etc -name '*srm*'
```

这个命令表明了 Linux 系统将在/etc 整个目录中查找所有的包含有 srm 这 3 个字母的文件，比如 absrmyz, tibc.srm 等等符合条件的文件都能显示出来。如果你还知道这个文件是由 srm 这 3 个字母打头的，那么我们还可以省略最前面的星号，命令如下：

```
find/etc -name 'srm*'
```

这是只有像 srmyz 这样的文件才被查找出来，象 absrmyz 或者 absrm 这样的文件都不符合要求，不被显示，这样查找文件的效率和可靠性就大大增强了。

根据文件的特征查询方法：

```
find / -amin -10 # 查找在系统中最后 10 分钟访问的文件
find / -atime -2 # 查找在系统中最后 48 小时访问的文件
find / -empty # 查找在系统中为空的文件或者文件夹
find / -group cat # 查找在系统中属于 groupcat 的文件
find / -mmin -5 # 查找在系统中最后 5 分钟里修改过的文件
find / -mtime -1 #查找在系统中最后 24 小时里修改过的文件
find / -nouser #查找在系统中属于作废用户的文件
find / -user fred #查找在系统中属于 FRED 这个用户的文件
```

下面的列表就是对 find 命令所可以指定文件的特征进行查找的部分条件。在这里并没有列举所有的查找条件，参考有关 Linux 有关书籍可以知道所有 find 命令的查找函数。

-amin n

查找系统中最后 N 分钟访问的文件

-atime n

---

查找系统中最后  $n*24$  小时访问的文件

**-cmin n**

查找系统中最后 N 分钟被改变状态的文件

**-ctime n**

查找系统中最后  $n*24$  小时被改变状态的文件

**-empty**

查找系统中空白的文件，或空白的文件目录，或目录中没有子目录的文件夹

**-false**

查找系统中总是错误的文件

**-fstype type**

查找系统中存在于指定文件系统的文件，例如：ext2 .

**-gid n**

查找系统中文件数字组 ID 为 n 的文件

**-group gname**

查找系统中文件属于 gnam 文件组，并且指定组和 ID 的文件

技巧: find 配合 grep 来查询含有某个关键词的文件

```
find /www -name "*.php" |xargs grep 'mysql_connect'
```

## 第五章 Linux 的系统命令

whoami 或者 who am I, 显示当前登陆者的用户名称

用法 whoami 或 who am I

示例 先用 whoami 看看自己是谁

再 su -登陆 root 账号, 再运行 who am i 看看自己谁?

who 是显示谁在线

su - 切换用户

用法 su 用户名 或者 su - 用户名

示例 su - root

问: su 后面加-和不加-有什么区别?

答:加 -后,不仅切换到其他用户,而且环境变量这些什么的都切换成其他用户的.

而不加-的话,仅仅使用这个用户的权限,而不使用其他.

**free**

Free 显示内存状态

用法 free -m -s (-m 表示内存的状态用 M 为单位来表示, -s 表示刷新的时间间隔,秒为单位)

示例: free -m -s 3

top (类似于 windows 下面的资源管理器)

ps 列出进程

用法 ps -aux

管道操作

\$xx|strtoupper ,竖线前面的内容,作为竖线后的命令的输入参数

kill 进程号, root身份运行  
pkill 进程名, 注意,杀所有 \*进程名\* 的进程

## 第六章 文件挂载

大家知道,linux 下,几乎把一切都看成文件(甚至内存都看成文件来管理),那么,当我们一个新设备(比如 USB,光盘)连上电脑后,我们也要想办法把此设备挂载到一个文件夹下,当成文件来管理.这就用到文件挂载的概念.

其实,你在装 linux 系统时,自己如果给硬盘分过区,那么分过的这些区,其实就是不同的硬盘空间挂载在不同的文件夹下

mount 命令,直接输入,可以看到目前的硬盘分区和所挂载点.

mount 命令用法

mount -t 文件系统类型 -o 选项参数 device(设备) dir(挂载点)

-t : 如 fat32,ext3,iso9660,ntfs 等,但这个选项通常不必指定,系统往往能自动认出来.

-o : option,选项的意思,可以是-ro(只读) -rw(读写),-loop(把文件看成一分区)

示例:

当我们在命令行模式下,插入光盘后,要把光盘挂到某一个具体的文件夹下,才能访问.

1:su - 登录要账户

2:创建一个空文件夹 如/mnt/cd

3: mount /dev/cdrom /mnt/cd

4: cd /mnt/cd 就可以看到光盘的内容了.

umount 卸载

用法 umount dir(挂载点) 或者 umount dev(所挂载的设备)

示例 umount /mnt/cd

(再 ls /mnt/cd 看看,还有东西吗?)

## 第七章 vi 编辑器

(注:我们实际使用的一般是 vim,即 vi 的加强版,有代码高亮等实用功能)

### 最重要的概念 :编辑模式,命令模式,尾行模式

当用 vi 打开一个文件后,首先停留在命令模式里.

#### 跟着我做:

vi tmp.txt (不用担心,如果没有此文件会自动创建)

此时,你狂按空格键,发现什么也没打入进去,因为此时还在命令模式.

如何进入编辑模式呢?

你可以按 i,a,o,s 三个键,

i,进入编辑模式,且光标位置不变.

a,且光标往后移 1

o,且光标跳下去一行

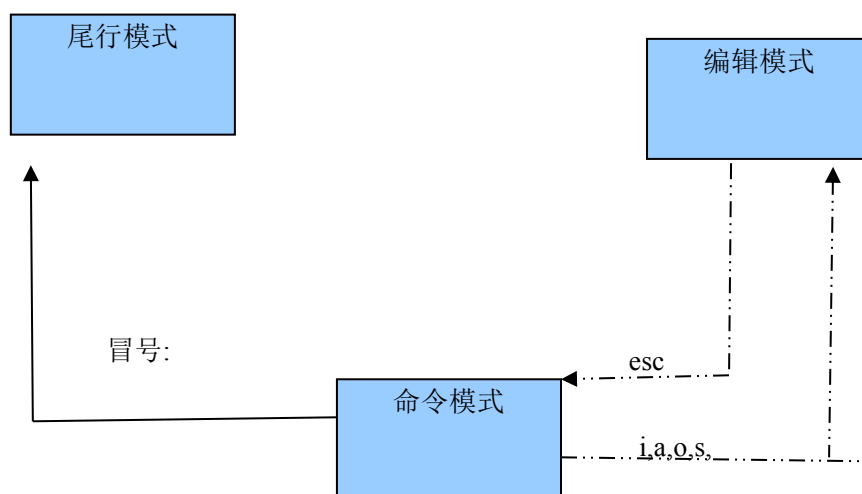
s,删除当前一个字符

在编辑模式里,我们可以自由输入,是不是在命令模式里,就完全无法更改文件的内容呢?

不是的,对于有些小的更改,其实在命令模式里,会更方便.

(如 x,p,y,dd,.)等命令,具体看我发给大家的 vim.htm 那个网页

#### 三种之间如何切换?





## 命令模式下的光标快捷移动键

### 字符级移动

- h 左移一个字符
- l 右移一个字符

### 单词级移动

- w 移到下个单词首
- e 移到本单词尾
- b 移到本单词首

### 行级移动

- \$ 移到行尾
- 0 移到行首
- j 下移一行
- k 上移一行

### 段级移动

- { 上移一段
- } 下移一段

### 屏级移动

- H 移到本屏幕第一行
- L 移到本屏幕最后一行

### 文章级移动

- G 移到文章末尾
- 1G 移到文章开头

### 命令模式下的快捷删除

- 1:d+光标快捷移动键
- 2:x 删除当前字符
- 3:dd 删除一行

### 命令模式下的复制

- 1:v+光标快捷移动键+y
- 2:yy 复制一行
- 3:yny 复制 n 行,n 为数字

### 命令模式下的粘贴命令

p

---

有效的命令组合

xp 交换两个字符

ddp 交换两行

其他命令

. 重复上一次命令

u 撤消

J 合并两行

尾行模式,则相对简单一些

w 保存,

q 退出

wq 保存+退出

!表示强制

如 q!,更改了之后,不保存,强制退出.

## 第八章 网络配置

如果是在虚拟机下,需要配置虚拟机的网卡,如下:



桥接模式: 虚拟网卡和物理网卡同等地位,获得和物理在同一局域网的 IP

界面名称: 虚拟网卡通过哪张物理卡上网,自己的物理用的哪张网卡,就选哪张.

接入网线: 一定要选.

[注意: 如果是真实服务器,不必经过这些]

网络配置:

1: 可以通过 `setup` 命令,基本图形界面来配置.

如果有此命令,会看到和 window 下类似的图形界面 .

2: 如果没有,则编辑网络相关的配置文件:

对于 redhat,centos, 网络的主要配置文件有如下几项:

2.1) `/etc/resolv.conf` 指定 DNS 服务器用的(如果是自动获取 IP,可以不设定)

示例内容:

```
nameserver 208.67.222.222
```

```
nameserver 8.8.8.8
```

2.2) `/etc/sysconfig/network-scripts/ifcfg-eth0`

注意: 红色 0,是变量, 不同的网卡,有不同的值.

比如 3 块网卡,分别对应 `ifcfg-eth0` , `eth1`,`eth2`

对于网卡常用的字段如下:

```
TYPE=Ethernet      #网卡类型
DEVICE=eth0        #网卡接口名称
ONBOOT=yes         #系统启动时是否自动加载
BOOTPROTO=static   #启用地址协议 --static:静态 IP --dhcp 自动获取
IPADDR=192.168.1.11 #网卡 IP 地址
NETMASK=255.255.255.0 #网卡网络地址
GATEWAY=192.168.1.1 #网卡网关地址
HWADDR=00:0C:29:13:5D:74 #网卡设备 MAC 地址[一般不能改]
BROADCAST=192.168.1.255 #网卡广播地址 [可不写]
```

Static 配置示例, 打开 ifcfg-eth0 看到如下内容:

```
DEVICE=eth0
HWADDR=08:00:27:FA:46:DB
GATEWAY=192.168.1.1
TYPE=Ethernet
UUID=6d96f3cb-1ed0-4dcd-a996-1c4d6f6ad3bc
ONBOOT=yes
NM_CONTROLLED=no
BOOTPROTO=dhcp
```

修改为:

```
DEVICE=eth0
HWADDR=08:00:27:FA:46:DB
GATEWAY=192.168.1.1
NETMASK=255.255.255.0
TYPE=Ethernet
UUID=6d96f3cb-1ed0-4dcd-a996-1c4d6f6ad3bc
ONBOOT=yes
NM_CONTROLLED=no
BOOTPROTO=static
IPADDR=192.168.1.108
```

对于一块网卡,一般配置 3 项,其他的默认即可.

Ipaddr

Gateway

Netmask

根据自己的网络情况做修改

修改后: service network restart 重启网络

Dhcp 自动获取示例:

```
DEVICE=eth0
HWADDR=08:00:27:FA:46:DB
GATEWAY=192.168.1.1
NETMASK=255.255.255.0
TYPE=Ethernet
UUID=6d96f3cb-1ed0-4dcd-a996-1c4d6f6ad3bc
ONBOOT=yes
NM_CONTROLLED=no
BOOTPROTO=dhcp
```

```
[root@bogon network-scripts]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 08:00:27:FA:46:DB
          inet addr:192.168.1.184  Bcast:192.168.1.255  Mask:255.255.255.0
```

## 网络配置时常用的命令

service network stop/start/restart 停/启/重启网络

ifdown eth0 关闭 eth0 网卡

ifup eth0 激活 eth0 网卡

ifconfig 查看网卡信息

## 第九章 远程管理

### 远程连接协议: Ssh 协议



ftp:21

Ssh:22

Telnet:23

http:80

https:443

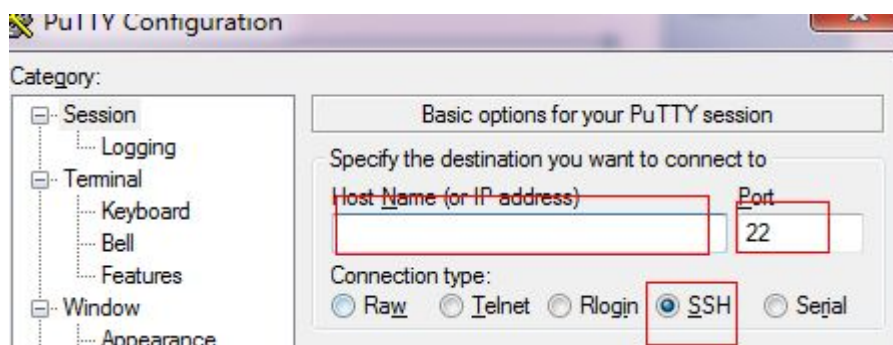
Mysql:3306

远程连接工具: putty, “ssh secure shell client”,securecr xshell

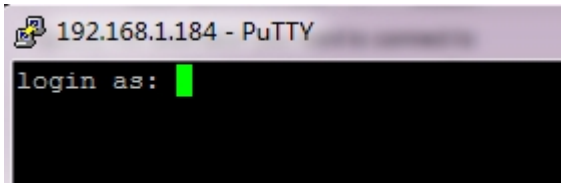
注 1):putty 中文版中后门,从官方下载英文原版

注 2) ssh 协议一般用来执行操作,也可以上传文件,和具体的工具有关,“ssh secure shell clien”可以上传文件

### 以 putty 为例



填写 IP 或域名后,看到如下界面



输入用户名和密码即可登陆

## 第十章 linux 的用户管理

### 知识点:

增加组(groupadd)

修改组(groupmod)

删除组(groupdel)

增加用户(useradd)

修改用户(usermod)

删除用户(userdel)

groupadd 增长一个用户组

用法 groupadd [-g 组 id] 组名称

示例 groupadd -g 502 javaf (指定组 id=502)

groupadd javae (不指定组 id,由系统自动生成)

注意: 以上两个命令并不一定会运行成功,因为可能已经存在 502 或 javaf,javae 组  
这个时候你可以通过 tail /etc/group 来查看组的信息.

如果已经存在导致不能正常加入,请删除之.

groupdel 删除一个组

用法 groupdel 组名

示例 groupdel javae

groupmod 修改一个组

用法 groupmod [-g 组 id] [-n 新组名] 组名

示例 groupmod -g 509 javaf

groupmod -n javax javaf

useradd 增加一个用户

用法 useradd [-g 组名] [-d 家目录]

示例 useradd -g javaf -d /var/javaf03/ javaf03

usermod 修改一个用户

用法 usermod [-g 组名] [-d 家目录] [-l 新用户名] 用户名

示例 usermod -g 502 -d /home/javaf03 javaf03

userdel 删除一个用户

用法 userdel 用户名



---

示例 userdel javaf03

userdel mysql -r // -r 代表连用户的相关文件彻底删除

passwd 修改密码

注意,我们刚刚用 useradd 增长一个新用户之后,还不能立即使用,得给新用户设一个密码才可以使用.

用法: passwd username

然后两次输入相同密码,就可以使用了.

注:

用户信息在/etc/passwd 文件里存储

用户密码在/etc/shadow 文件里加密存储

组信息在 /etc/group 文件里存储

## 第十一章 权限管理

用户	权限	权限的数值表示法
u(表示文件的属主)	r (读权限)	4
g(与文件属主一组的用户)	w(写权限)	2
o(既非 u,又不同组的其他人)	x(执行权限)	1

每个文件或者文件夹,都有 3 个权限

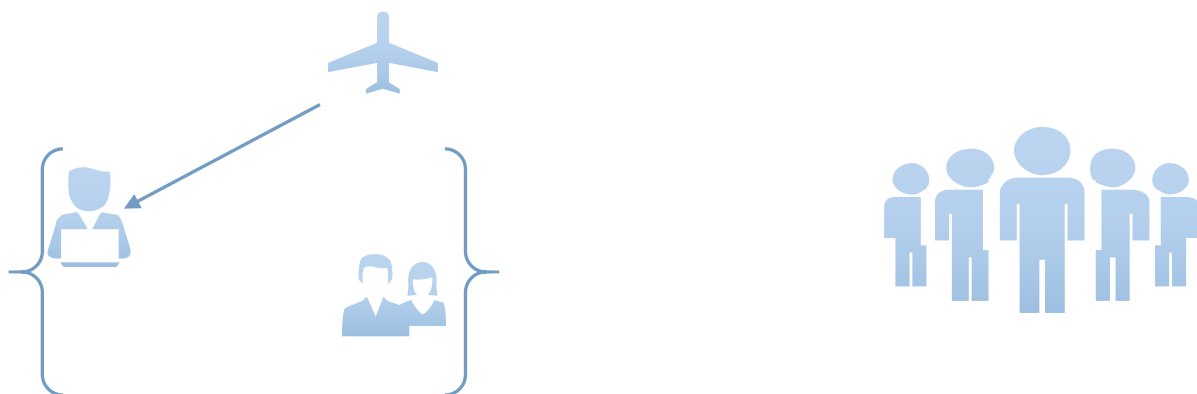
r(read 读)	w (write,写)	x (excute,执行)
1/0	1/0	1/0
4	2	1

一个用户,对于一个文件,有 3 种可能的身份

lisi: 是飞机的主人 user

{liwu,liyuan} 和飞机的主人同组 ,group

{wangwu,zhaoliu},和此飞机没有任何关系, other



每个文件或者文件夹,都有 3 个权限,分别是对 u 的权限,对 g 的权限,对 o 的权限

例如:-rwxrw-r-- for.sh

我们可以知道, 对于 for.sh 文件,

主人有读写及执行权,

对于与主人同组的人,有读写权,

而其他用户,则只有读权限.

思考,如果用数字来表示此文件的权限应该为多少?

答: 764

## 修改权限的方法

修改权限有两种方法,一种是相对法,一种是绝对法.

相对法,是指在原有权限基础上,针对不同的角色加/减 某一个权限

绝对法,则不需考虑之前的角色及对应权限,直接指定最终的各角色权限

做如下试验

## 绝对法修改权限

特点:不管之前的权限是什么样儿的,直接把 3 个对象(u,g,o)的权限改成指定权限

如 `chmod 755 for.sh`

## 相对法修改权限

特点:在知道当前权限的基础上,针对 3 种角色(u,g,o)中的某一个角色,增加或减小某种权限(r,w,x)

如 `chmod o+x for.sh`

修改一个文件的主人

`chown` (change owner 改变主人)

`chown` 新主人 某文件 ,

修改一个文件的组

`chgrp`(change group)

`chgrp` 新组 某文件

## 第十二章 rpm 软件管理包

### RPM 概述

RPM 是一个开放的软件包管理系统，最初的全称是 Red Hat Package Manager。它工作于 Red Hat Linux 以及其它 Linux 系统，成为了 Linux 中公认的软件包管理标准。红帽子软件公司鼓励其他厂商来了解 RPM 并在自己的产品中使用它。

RPM 的发布基于 GPL 协议。随着 RPM 在各种发行版本的广泛使用，如今 RPM 的全称是 RPM Package Manager。RPM 由 RPM 社区负责维护，其官方网站：<http://www.rpm.org>

### 为什么使用 RPM

使用 RPM 最大的好处在于它提供快速之安装，减少编译安装之烦琐困扰。对于最终用户来说，RPM 所提供的众多功能使维护系统要比以往容易的多。安装、卸载和升级 RPM 软件包均只需一条命令即可完成，所有烦琐的细节问题无须您费心。RPM 维护一个所有已安装的软件包和文件的数据库，可以让您进行功能强大的软件包查询和验证工作。

在软件包升级过程中，RPM 会对配置文件进行特别处理，因此您绝对不会丢失以往的定制信息——这对于直接使用.tar.gz 文件是不可能的。对于程序员，RPM 可以让您连同软件的源代码打包成源代码和二进制软件包供最终用户使用。这个过程十分简单，整个过程由一个主文件和可能的补丁程序组成。RPM 在软件的新版本发布时，这种“原始”源代码，补丁程序和软件生成指令的清晰描述简化了软件包的维护工作。

### RPM 的功能

简言之，RPM 具有如下五大功能：

- \* 安装——将软件从包中解出来，并且安装到硬盘。
- \* 卸载——将软件从硬盘清除。
- \* 升级——替换软件的旧版本。
- \* 查询——查询软件包的信息。
- \* 验证——检验系统中的软件与包中软件的区别。

### RPM 包的名称格式

RPM 包的名称有其特有的格式，如某软件的 RPM 包名称由如下部分组成：

name-version.type.rpm

其中：

- \* name 为软件的名称
- \* version 为软件的版本号
- \* type 为包的类型
- \* i[386]86: 表示在 Intel x86 计算机平台上编译的
- \* sparc: 表示在 sparc 计算机平台上编译的
- \* alpha: 表示在 alpha 计算机平台上编译的
- \* src: 表示软件源代码
- \* rpm 为文件扩展名

例如：

bind-9.2.1-16.i386.rpm: 是 bind (9.2.1-16) 的 Intel 386 平台编译版本包

bind-9.2.1-16.sparc.rpm: 是 bind (9.2.1-16) 的 sparc 平台编译版本包

bind-9.2.1-16.alpha.rpm: 是 bind (9.2.1-16) 的 alpha 平台编译版本包

bind-9.2.1-16.src.rpm: 是 bind (9.2.1-16) 的源代码版本包

因此，看到一个 RPM 包的文件名之后就可以获得该软件包的大致信息了。

rpm 命令的使用

rpm 命令简介

在 CentOS 中升级和安装系统通常使用 yum 命令，因为它可以良好的解决包的依赖性问题，即自动安装/处理依赖的其他软件包。但是 rpm 命令还是在某些情况下用得上，比如，查询包信息，安装或卸载一个不在 CentOS 软件库中的.rpm 包等。

常见的用法：

命令      说明

- |                           |                        |
|---------------------------|------------------------|
| rpm -i <.rpm file name>   | 安装指定的 .rpm 文件          |
| rpm -U <.rpm file name>   | 用指定的.rpm 文件升级同名包       |
| rpm -e <package-name>     | 删除指定的软件包               |
| rpm -q <package-name>     | 查询指定的软件包在系统中是否安装       |
| rpm -qa                   | 查询系统中安装的所有 RPM 软件包     |
| rpm -qf </path/to/file>   | 查询系统中指定文件所属的软件包        |
| rpm -qi <package-name>    | 查询一个已安装软件包的描述信息        |
| rpm -ql <package-name>    | 查询一个已安装软件包里所包含的文件      |
| rpm -qc <package-name>    | 查看一个已安装软件包的配置文件位置      |
| rpm -qpi <.rpm file name> | 查询一个未安装的 RPM 文件的描述信息   |
| rpm -qpl <.rpm file name> | 查询一个未安装的 RPM 文件里所包含的文件 |
| rpm -qpc <.rpm file name> | 查看一个未安装的 RPM 文件的配置文件位置 |
| rpm -qpd <.rpm file name> | 查看一个未安装的 RPM 文件的文档安装位置 |
| rpm -qpR <.rpm file name> | 查询一个未安装的 RPM 文件的最低依赖要求 |
| rpm -V <package-name>     | 校验指定的软件包               |
| rpm -V </path/to/file>    | 校验包含指定文件的软件包           |
| rpm -Vp <.rpm file name>  | 校验指定的未安装的 RPM 文件       |

`rpm -Va` 校验所有已安装的软件包

`rpm --rebuilddb` 重新创建系统的 RPM 数据库，用于不能安装和查询的情况

`rpm --import <key file>` 导入指定的签名文件

`rpm -Kv --nosignature <.rpm file name>` 检查指定的 RPM 文件是否已损坏或被恶意篡改（验证包的 MD5 校验和）

`rpm -K <.rpm file name>` 检查指定 RPM 文件的 GnuPG 签名

`rpm -qd <package-name>` 查看一个已安装软件包的文档安装位置

`rpm -qR <package-name>` 查询一个已安装软件包的最低依赖要求

\* 在安装/升级时，还可以使用 `-vh` 参数，其中：`v` 表示在安装过程中将显示较详细的信息；`h` 表示显示水平进度条

\* 在使用 `rpm -qa` 命令时，还可以使用 `|more` 或 `|grep` 进行过滤

\* 所有的 `<.rpm file name>` 既可以是本地文件，也可以是远程文件

\* 校验软件包将检查软件包中的所有文件是否与系统中所安装的一致性。包括校验码文件大小，存取权限和属主属性都

\* 将根据数据库进行校验。该操作可在用户安装了新程序以后怀疑某些文件遭到破坏时使用。

## rpm 命令使用举例

安装软件包:

1、安装本地软件包

```
# rpm -ivh /media/CentOS_5.5_Final/CentOS/vsftpd-2.0.5-16.el5_4.1.i386.rpm
#注:红色部分代表将要安装的软件的路径,根据自己的实际情况来,不要照抄
Preparing... ##### [100%]
 1:vsftpd ##### [100%]
```

## 升级软件包:

1、从本地文件升级软件包:

```
# rpm -Uvh vsftpd-2.0.5-16.el5_4.1.i386.rpm
Preparing... #
```

## 卸载软件包:

```
# rpm -e vsftpd
```

## 查询软件包:

```
# 查询 vsftpd 软件包在系统中是否安装
$ rpm -q vsftpd
# 查询系统中已安装的 vsftpd 软件包的描述信息
$ rpm -qi vsftpd
# 查询系统中已安装的 vsftpd 软件包里所包含的文件
$ rpm -ql vsftpd
# 查询系统中文件 /etc/vsftpd/vsftpd.conf 所属的软件包
$ rpm -qf /etc/vsftpd/vsftpd.conf
# 查询 vsftpd-2.0.5-16.el5_4.1.i386.rpm 包文件中的信息
$ rpm -qp vsftpd-2.0.5-16.el5_4.1.i386.rpm
# 查询系统中已经安装的所有包含名字 vsftpd 的软件包
$ rpm -qa | grep vsftpd
# 查询 httpd 包的最低依赖要求
$ rpm -qR httpd
```

## 验证软件包:

```
// 验证 vsftpd 软件包
# rpm -V vsftpd
// 验证包含文件 /etc/passwd 的软件包
# rpm -Vf /etc/passwd
// 验证 vsftpd-2.0.5-16.el5_4.1.i386.rpm 包文件
# rpm -Vp vsftpd-2.0.5-16.el5_4.1.i386.rpm
// 验证所有已安装的软件包
# rpm -Va
```

如果校验一切正常，将没有输出，反之则输出不一致结果，格式为：

xxxxxxxx 文件名

字段 1 由八个字符组成，每个字符指明该文件与 RPM 数据库中一致或不一致的地方，单个点 (.) 说明没有异常，具体含义如下：

- \* 5 — 校验和
- \* S — 文件大小
- \* L — 符合连接
- \* T — 文件修改时间
- \* D — 设备
- \* U — 用户
- \* G — 组
- \* M — 文件模式
- \* ? — 文件不可读

## 第十二章 yum 软件管理器

yum 是一个用于管理 rpm 包的后台程序，用 python 写成，可以非常方便的解决 rpm 的依赖关系。在建立好 yum 服务器后，yum 客户端可以通过 http、ftp 方式获得软件包，并使用方便的命令直接管理、更新所有的 rpm 包，甚至包括 kernel 的更新。它也可以理解为红旗环境下的 apt 管理工具。

### 一、列举包文件

列出资源库中所有可以安装或更新的 rpm 包

```
# yum list
```

列出资源库中特定的可以安装或更新以及已经安装的 rpm 包

```
# yum list perl           //列出名为 perl 的包
```

```
# yum list perl*         //列出 perl 开头的包
```

列出资源库中所有可以更新的 rpm 包

```
# yum list updates
```

列出已经安装的所有的 rpm 包

```
# yum list installed
```

列出已经安装的但是不包含在资源库中的 rpm 包

```
# yum list extras
```

注:extras 是 repos.d 中定义的资源列表名称

### 二、列举资源信息

列出资源库中所有可以安装或更新的 rpm 包的信息

```
# yum info
```

列出资源库中特定的可以安装或更新以及已经安装的 rpm 包的信息

```
# yum info perl          //列出 perl 包信息
```



```
# yum info perl*           //列出 perl 开头的包的信息
```

列出资源库中所有可以更新的 rpm 包的信息

```
# yum info updates
```

列出已经安装的所有的 rpm 包的信息

```
# yum info installed
```

列出已经安装的但是不包含在资源库中的 rpm 包的信息

```
# yum info extras
```

### 三、搜索

搜索匹配特定字符的 rpm 包

```
# yum search perl          //在包名称、包描述等中搜索
```

搜索有包含特定文件名的 rpm 包

```
# yum provides realplay
```

### 四、管理包

安装 rpm 包

```
# yum install perl        //安装 perl 包
```

```
# yum install perl*       //安装 perl 开头的包
```

删除 rpm 包,包括与该包有倚赖性的包

```
# yum remove perl*        //会删除 perl-* 所有包
```

### 五、更新

检查可更新的 rpm 包

```
# yum check-update
```

更新所有的 rpm 包

```
# yum update
```

更新指定的 rpm 包,如更新 kernel 和 kernel source

```
# yum update kernel kernel-source
```

大规模的版本升级,与 yum update 不同的是,连旧的淘汰的包也升级

```
# yum upgrade
```

六、清空缓存

清除暂存中 rpm 包文件

```
# yum clean packages
```

清除暂存中 rpm 头文件

```
# yum clean headers
```

清除暂存中旧的 rpm 头文件

```
# yum clean oldheaders
```

清除暂存中旧的 rpm 头文件和包文件

```
# yum clean
```

或

```
# yum clean all
```

**示例:** yum 安装 gcc 编译环境,为编译 lnmp 做准备

```
yum install gcc automake autoconf libtool gcc-c++
```

## 第十三章 软件编译

软件编译就是把 源代码(如 c,c++)编译成 2 进制  
注意要下载源代码

以 memcached 为例,来编译,到 [memcached.org](http://memcached.org) 下载源码.  
下载到/usr/local/src 下

编译软件分为 3 步---

1: `configure --prefix=/安装/路径`

如果还有其他选项,./configure --help 来查看

2: `make` 编译 [生成 2 进制]

3: `make install` [把生成的 2 进制复制到 `prefix` 指定的安装路径里]

其中 2,3 两步,可以合写为 `make && make install`

---

以 memcached 为例

```
./configure prefix=/usr/local/memcached
```

```
configure: error: libevent is required. You can get it from http://www.monkey.org/~provos/libevent/
```

If it's already installed, specify its path using `--with-libevent=/dir/`

这是因为----memcached 需要 libevent

你既可以下载 libevent 的源码,先编译 libevent,也可以 yum libevent

以 yum 为例:

```
# yum install libevent    安装后问题仍然存在
```

对于库,不仅要装库本身,往往还要装库的源码 `xx-devel`

```
# yum install libevent-devel
```

```
# ./configure prefix=/usr/local/memcached
```

```
# make
```

```
#mak install
```

## 第十四章 编译 Inmp

Linux+nginx+mysql+php

### 14.1 编译 nginx

1: 下载 <http://nginx.org/en/download.html>

选择 stable 版本下载

2: 解压

```
# tar xzf nginx.xxxx.tar.gz
```

3: 配置

```
#!/configure --prefix=/usr/local/nginx
```

如果提示缺少 pcre 库,

则从 <http://www.pcre.org/>

假设解压在 /usr/local/src/pcre-source

假设安装在 /usr/local//pcre

4: 再次配置

1.6.X 版本,要求指定 pcre 的源码目录,即:

```
#!/configure --prefix=/usr/local/nginx \
```

```
--with-pcre=/usr/local/src/pcre-source
```

之前的版本,指定 pcre 的安装目录,即:

```
#!/configure --prefix=/usr/local/nginx \
```

```
--with-pcre=/usr/local/pcre
```

5: make && make install

6: 启动 nginx

```
./sbin/nginx
```

7: 启动时,极易出现端口被占的错误,如下

```
[root@bogon nginx]# ./sbin/nginx
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
```

这是因为 80 端口已被 apache 或其他 webserver 占据,

可以用 ps 命令来查找可以进程,然后杀掉.

ps -9 进程名

如 ps -9 httpd

或 kill 进程号

8: 连接,从局域网内连接 nginx,  
如果连接不能,先 ping 测试,2 台机器之间网络是否通,  
再在服务器上 telnet localhost 80  
如果 2 者都能,但外界连不上 80 端口,则是防火墙的原因.

service iptables stop

再次连接,出现以下界面,则安装成功

## Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](http://nginx.org).  
Commercial support is available at [nginx.com](http://nginx.com).

*Thank you for using nginx.*

### 14.5 编译安装 PHP

#yum install gd zlib zlib-devel openssl openssl-devel libxml2 libxml2-devel libjpeg libjpeg-devel libpng libpng-devel  
解压,cd 到 php-5.x

```
# ./configure --prefix=/usr/local/php \  
--with-gd \  
--enable-gd-native-ttf \  
--enable-gd-jis-conv \  
--enable-mysqlnd \  
--with-mysql=mysqlnd \  
--with-pdo-mysql=mysqlnd \  
--with-openssl \  
--enable-mbstring \  
--enable-fpm
```

针对PHP7,应如下配置:

```
./configure --prefix=/usr/local/php \  
--with-gd \  
--enable-gd-native-ttf \  
--enable-gd-jis-conv \  
--enable-mysqlnd \  
--with-pdo-mysql=mysqlnd \  
--with-openssl \  
--enable-mbstring \  
--enable-fpm
```

```
--with-pdo-mysql=mysqlnd \ 一定要加!  
--with-pdo-mysql=mysqlnd \ 一定要加!  
--with-pdo-mysql=mysqlnd \ 一定要加!
```

--enable-fpm 是让 PHP 作为独立的进程来运行(默认占据 9000 端口)

如果是和 apache 配合,PHP 一般是作为 apache 的模块来使用

```
# cd /usr/local/php
# cp etc/php-fpm.conf.default etc/php-fpm.conf
#cp /usr/local/src/php-5.5.13/php.ini-development ./lib/php.ini
#./sbin/php-fpm
```

#### 14.6 整合 nginx 和 PHP

Vim /path/to/nginx.conf

```
location ~ /\.php$ { 碰到.php结尾的,用此段指令来处理
    root             html;
    fastcgi_pass      127.0.0.1:9000; 把PHP文件转交给127的9000端口
    fastcgi_index     index.php;
    fastcgi_param     SCRIPT_FILENAME  /scripts$fastcgi_script_name;
    include           fastcgi_params; 上句是说明到哪儿去找xx.php
}
```

根据实际情况修改,例:

```
location ~ /\.php$ {
    root             html;
    fastcgi_pass      127.0.0.1:9000;
    fastcgi_index     index.php;
    fastcgi_param     SCRIPT_FILENAME  /usr/local/nginx/html$fastcgi_script_name;
    include           fastcgi_params;
}
```

也可以如下方式:

```
location ~ /\.php$ {
    root             html;
    fastcgi_pass      127.0.0.1:9000;
    fastcgi_index     index.php;
    fastcgi_param     SCRIPT_FILENAME  $DOCUMENT_ROOT$fastcgi_script_name;
    include           fastcgi_params;
}
```

让 nginx 的最新配置文件生效

```
# ./sbin/nginx -s reload
```

再次请求 xx.php,看到如下类似效果,即整合成功

PHP Version 5.5.13



<b>System</b>	Linux bogon 2.6.32-358.el6.x86_64 #1 SMP Fri Feb 22 00:31:26 UTC 2013 x86_64
<b>Build Date</b>	Jun 10 2014 00:26:59
<b>Configure Command</b>	'./configure' '--prefix=/usr/local/php' '--with-gd' '--enable-gd-native-ttf' '--enable-gd-jis-conv' '--with-mysql=mysqlnd' '--enable-mysqlnd' '--with-pdo-mysql=mysqlnd' '--enable-fpm'



## 14.2 编译 MySQL

<http://ftp.nchu.edu.tw/Unix/Database/MySQL/Downloads/MySQL-5.5/>

mysql-5.5.30-linux2.6-x86\_64.tar.gz

MySQL 的安装稍复杂一些(主要是编译后的配置及初始化),大家注意,碰到开源软件

1:官网的安装介绍

2: 下载源码后,一般有 README/INSTALL

3: ./configure --help

我们可以下载 2 进制版本来安装:

官方示例:

```
shell> groupadd mysql
shell> useradd -r -g mysql mysql
shell> cd /usr/local
shell> tar zxvf /path/to/mysql-VERSION-OS.tar.gz
shell> ln -s full-path-to-mysql-VERSION-OS mysql
shell> cd mysql
shell> chown -R mysql .
shell> chgrp -R mysql .
shell> scripts/mysql_install_db --user=mysql # 安装初始化数据
shell> chown -R root .
shell> chown -R mysql data
```

具体安装流程:

```
# groupadd mysql
[root@bogon mysql5.5]# useradd -g mysql mysql
[root@bogon mysql5.5]# cd /usr/local/mysql5.5/
#chown -R mysql .
# chgrp -R mysql .
```

```
# ./scripts/mysql_install_db --user=mysql
```

如果提示如下错误:

```
/bin/mysqld: error while loading shared libraries: libaio.so.1: cannot open shared object file: No such file or directory
```

则# yum install libaio.so.1 libaio

然后再次执行

```
# chown -R root .
# chown -R mysql data
# mkdir /var/run/mysqld
# chown mysql /var/run/mysqld
```

```
# chgrp mysql /var/run/mysqld
# ./bin/mysqld_safe --user=mysql &
```

### 14.3 mysql 连接

Mysqld 安装后,连接经常出现找不到 sock 的情况

ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)

```
[root@bogon mysql5.5]# ./bin/mysql -uroot
ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)
[root@bogon mysql5.5]# ps aux|grep mysql
root      22517  0.0  0.3 106192 1516 pts/1    S   23:23   0:00 /bin/sh ./bin/mysqld_safe --user=mysql
mysql     22669  0.1  9.6 619472 48440 pts/1    Sl  23:23   0:00 /usr/local/mysql5.5/bin/mysqld
--basedir=/usr/local/mysql5.5 --datadir=/var/lib/mysql --plugin-dir=/usr/local/mysql5.5/lib/plugin --user=mysql --log-error=/var/log/mysqld.log --pid-file=/var/run/mysqld/mysqld.pid --socket=/var/lib/mysql/mysql.sock
root      22695  0.0  0.1 103244   860 pts/1    S+  23:28   0:00 grep mysql
```

我们用 2 个办法来解决

1: 建立软件链接

```
#ln /var/lib/mysql/mysql.sock /tmp/mysql.sock
```

2: 查看 mysql --help

```
[root@bogon mysql5.5]# ./bin/mysql --help|grep sock
--protocol=name The protocol to use for connection (tcp, socket, pipe,
-S, --socket=name The socket file to use for connection.
                    The buffer size for TCP/IP and socket communication.
socket              (No default value)
```

Mysql -S /path/to/mysql.sock

### 14.4 mysql 修改密码

Mysql 用户的密码,存储在一个系统库里的---mysql

```
mysql> select Host,User,Password from user;
+-----+-----+-----+
| Host      | User  | Password |
+-----+-----+-----+
| localhost | root  |          |
| bogon     | root  |          |
| 127.0.0.1 | root  |          |
| ::1       | root  |          |
| localhost | root  |          |
| bogon     | root  |          |
+-----+-----+-----+
```

注意: mysql 用户权限检测,检测 Host,User,Password

```
mysql> update user set Password=password('123456') where Host='localhost' and User='root';
```

```
mysql> delete from user where Password="";
```

```
mysql> flush privileges;
```

## 第十五章 shell 编程(选学内容)

### 15.1 入门

什么是 shell?

首先打开一个终端,在终端里能做的操作,  
如 `cd,cp` 等  
以及在终端里执行的命令,如 `date,echo` 等,  
都能同样的写进.sh 脚本里面去,

此时,你可以把脚本文件理解成一个”命令包”,运行一个文件相当于运行了里面的多个操作和命令.  
(参照 windows 下的.bat 文件来理解)

那么让我们来做一个最简单的.sh 文件

文件名: `simple.sh`

代码示例

```
#!/bin/bash
cd /var/tmp/
pwd
```

第 2 步,加上执行权限, `chmod u+x simple.sh`

(脚本编辑完成后,需要加”可执行权限”这一步,在下面的脚本示例中,就不再重复写了,请大家自己加)

第 3 步 执行, `./simple.sh`

看看打印什么?(和在终端直接执行 `cd /var/tmp , pwd` 两个命令的效果是一样的)

### 二: 进阶

既然是 shell 编程,那么必然少不了变量,表达式,及控制结构这些.

分别讲解如下:

**变量有三种:**

- 1:自定义变量
- 2:系统变量
- 3:命令返回值变量

我们先来试一下自定义变量

代码示例

```
#!/bin/bash
var1=hello
var2=bash
```

```
#注意,=号两边都不要有空格  
echo $var1 $var2
```

此时,我们就声明了一个简单的变量.

**注意:**声明变量时,不要加\$,且'='号前后不要加空格,  
而在引用变量时,则要加上\$,如 `echo $var1`

再来试一下 [系统变量](#)

文件名: sys.sh

代码示例

```
#!/bin/bash  
echo $HOME  
echo $USER
```

运行 sys.sh 查看结果?

对于\$HOME,\$USER,我们并没有定义这两个变量,但为什么可以打印出结果呢?  
这是因为\$HOME,\$USER 是系统变量,系统已经定义好了,不用我们再定义.

最后来试一下 [命令返回值变量](#)

脚本: com.sh

代码示例

```
#!/bin/bash  
var1=`date +%y%m%d`  
echo $var1  
date +%y%m%d
```

运行 ./com.sh, 看看打印结果? 分析一下两行的结果各是怎样打印出来的?

**思考:**命令的输出内容如何赋到变量里?

答:有时候,我们需要把命令的运行结果捕捉,并赋给某变量,则可以用反引号 `命令` 来实现.  
如 `pos=`pwd``,则 `pwd` 命令的结果将赋给 `pos` 变量.

## 三:提高

会定义变量,会使用系统变量,会捕捉命令的返回值,还不够,  
有了变量,还要有表达式和控制结构,才能叫”程序”

表达式有”命令表达式”,”数学表达式”,”字符串表达式”,”文件判断表达式”

控制结构有 [if/ case/for/while](#)

先来看一下 [命令表达式](#)和 [if/else 控制结构](#)

命令表达式 是指 如果一个命令执行成功,则返回 true,如果执行失败,则执行 false

if/else 的语法格式如下:

```
if 表达式
then
语句 1
语句 2
....
else
语句 one
语句 two
.....
fi
```

结合命令表达式,和 if 控制结构,我们创建一个脚本:com\_exp.sh

作用:帮我们创建一个目录 dir1, 如果创建成功,则显示 ok;如果失败,则显示 fail.

代码示例:

```
#!/bin/bash
if mkdir dir1
then
echo ok
else
echo fail
fi
```

先保证当前目录下没有 dir1 目录,然后运行 2 次此脚本,看看分别打印什么结果,并思考为什么?

## 数学比较表达式

数学比较表达式的作用: 比较两数的大小关系

数学表达式是这样的结构[ \$var1 -gt/-lt/-eq/-ge/-le/-ne \$var2 ]

注意:中括号两端各有 1 空格

比较运算符的含义见下:

```
-gt >
-lt <
-ge >=
-le <=
-eq =
-ne !=
```

创建脚本 math.sh

作用:判断两个变量的大小,并打印结果.

代码示例:

```
#!/bin/bash
var1=8
```

```
var2=12
if [ $var1 -gt $var2 ]
then
echo $var1 more than $var2
elif [ $var1 -eq $var2 ]
then
echo $var1 equal $var2
else
echo $var1 less than $var2
fi
```

数学加减运算

```
#!/bin/bash
var1=8
var2=12
var3=$(( $var1 + $var2 ))
echo $var3
```

## 字符串判断表达式

字符串表达式的作用: 用于判断两字符串是否相同

格式: [ \$str1 != str2 ]

注意: 中括号两端各有 1 空格

代码示例

```
#!/bin/bash
var1=hello
var2=world
if [ $var1 = $var2 ]
then
echo $var1 is $var2
else
echo $var1 is not $var2
```

## 文件判断表达式

文件表达式作用: 判断文件是否存在, 是否可读/可写/可执行, 是否是目录或普通文件等, 以及用来比较两个文件的新旧顺序.

格式 1: [ -d/-f/-e/-r/-w/-x filename ]

分别判断

- d 文件是否存在且是目录
- f 文件是否存在且是文件
- e 是否存在
- r 是否可读
- w 是否可写
- x 是否可执行

格式 2: [ file1 -nt/-ot file2 ]

分别判断

-nt 检查 file1 是否为 file2 新

-ot 检查 file1 是否为 file2 旧

创建脚本: file.sh

作用:判断当前目录下有没有 dir2 目录

代码示例:

```
#!/bin/bash
if [ -d ./dir2 ]
then
ls dir2
else
echo dir2 is not exists
fi
```

控制结构:

if/else 结构(上面有很多例子,跳过)

for 结构 (for 循环有两种)

1: bash 风格的 for 循环

格式如下:

```
for 变量名 in 值 1 值 2 值 3 .... 值 n
do
语句 1
语句 2
....
done
```

创建脚本 bash\_for.sh

作用: 打印 A B C D E

代码示例:

```
#!/bin/bash
for i in A B C D E
do
echo $i
done
```

2:C 风格的 for 循环

格式如下:

```
for((变量=初始值;变量<=n;变量++))
do
```

```
    语句 1
    语句 2
    ...
done
```

创建脚本:c\_bash.sh  
作用:计算 1-100 的和  
代码示例:

```
#!/bin/bash
for((i=1;i<=100;i++))
sum=0
do
sum=$((sum + i))
done
echo $sum
```

## case 结构介绍

格式如下:

```
case 变量 in
可能性 1)
    语句 1
    语句 2
;;
可能性 2)
    语句 1
    语句 2
;;
*)
    语句 1
    语句 2
;;
```

创建脚本: case.sh

作用:判断当前登陆者是谁,并根据情况打印不同内容

代码示例

```
#!/bin/bash
case $USER in
shiba)
echo -n "you "
echo -n "are "
echo "shiba";;
shijiu)
echo you are shijiu
*)
echo sorry!
```



## esac

几行语句后面,放两个;;就可以了,而不是每行后面都放两个;;

\*代表以上情况都不是时,类似于 java 中的 default

案例练习:

某个目录下有文件,也有目录,比如, /usr/local/src 目录下,有很多.tar.gz 文件,也有解压出来的很多目录.

写一个 shell 脚本,运行并删除所有目录,保留文件.

## 第十六章 定时任务

定时任务的创建: `crontab -e` 进入任务编辑状态

格式为:

`* * * * *` 命令

分 时 日 月 周

`2 1 * * * aaa` # 代表每天 1:02 分执行 `aaa` 命令

`*/2 * * * * aaa` # 代表每 2 分钟执行 `aaa` 命令

执行结果会被发送到 `root` 的邮件里, 如果不想收到定时任务地信息

`You have new mail in /var/spool/mail/root`

可以把输出重定向

`2 1 * * * aaa > /dev/null 2>&1`

每周日凌晨 3 点执行

`0 3 * * 0 command`

每隔 2 分钟

`*/2 * * * * command`

## 第十七章 数据库定期备份实例

目标: 每天凌晨 3 点,导出.sql,压缩,并按日期存储在/data 下

知识: 定时任务 crontab , mysqldump 导出 , tar 压缩, 按日期创建文件 date

```
#!/bin/bash
/usr/local/mysql/bin/mysqldump -uroot -p123456 -B test > /data/test.sql
cd /data
tar zcf test.sql.tar.gz test.sql
mv test.sql.tar.gz bak/`date -d '-1 day' +%Y%m%d`.tar.gz

old=`date -d '-7 day' +%Y%m%d`
if [ -f /data/bak/$old.tar.gz ]
then
    rm -rf /data/bak/$old.tar.gz
fi
```

如何再把 7 天前的备份删掉 ,提醒:先通过 date 命令得到 7 天的日期,计算出 7 天前对应的文件名再用文件判断表达式,判断文件是否存在,如果存在则删掉.