**Linux笔记**

## Linux基础介绍

### 命令提示符

#：代表管理员用户



$：代表普通用户

### 命令格式

命令 选项 参数

选项：

短选项：-xxx

-ab= -a -b

长选项：--xxxx

参数：命令的作用对象

### 虚拟终端切换

Linux支持多个虚拟终端：

快捷键：Ctrl+alt+F1(F1…F12 实现多个虚拟终端间切换)

### Linux图形界面类型（GUI Graphic User Interface）

Linux的图形用户界面是C/S架构的，它不像Windows将图形用户界面嵌入在内核中。Linux的图形用户界面只是Linux的一个外围软件。

Gnome：C开发

KDE：C++

XFace

### 命令行界面分类（CLI Command-Line Interface）

Bash

Csh

Zsh

Ksh

Tcsh

### 切换用户（switch user）

命令：su 用户名

### 更换密码

命令：passwd（输完passwd后回车,然后再输两次密码）

### 进入伪图形界面

命令：setup

## Linux常用命令

### 获取命令的使用帮助

内部命令：help 命令名（只有内部命令才能使用）

外部命令：命令名 - -help

打开命令使用手册：man 命令名

### 显示当前所在目录

命令：pwd （print working directory ）

选项： -P : 他可以让我们取得正确的目录名称，而不是已链接文件的路径来显示的。（P178）

也就是说加上-P选项，如果你当前在一个连接文件夹内，就会显示链接的实际路径，如果不加则显示当前这个链接文件所在的路径

### 切换目录

命令：cd 路径

注：

1. cd不加任何参数，则回到用户的主目录（家目录）（当前用户的主目录通常用~表示）



1. cd ~用户名 ：在管理员权限下，进入其他用户的家目录（主目录）

如果省略用户名，例如：cd ~，则进入当前用户的家目录。

1. cd -：在前一个目录和当前目录来回切换。

### 显示当前目录下文件列表（ls）

命令：ls （list）

命令选项：

-l：显示完整信息（long），或者以ll代替

a：显示全部文件

d：仅列出目录本身，而不是列出目录内的文件数据

h：将文件的容量已比较容易阅读的方式展现（单位转换）

R：递归显示文件

r：倒序输出

S：以容量大小排序

t：依时间排序



① 在显示的完整信息中第一列代表文件类型，文件类型有下列几种：

-：普通文件

d：目录文件

b：块设备文件

c：字符设备文件

l：符号链接文件（symbolic link file）

p：命令管道文件

s：套接字文件（socket）

② 从第二列开始代表文件权限，每3位一组（一共三组），每一组rwx（读，写，执行）。

③ 代表文件被硬链接的次数

④ 文件的属主（owner）

⑤ 文件的属组（group）

⑥ 文件的大小（字节）

⑦ 时间戳 （最后一次修改文件（内容被修改）的时间）

-h：单位换算（将文件大小单位转换成易读的单位，K、M、G等，该选项通常与“-l”一起使用）



-a：显示以 . 开头的隐藏文件

-d：显示当前目录的自身属性（通常与-l选项同时使用）



-i：显示文件的索引节点号（index node number）



-r：逆序显示文件

-R：递归显示文件

### 查询命令的类型

命令：type 命令名



命令的类型：

内部命令（shell内置）：内部、内建

外部命令：在文件系统中的某个路径下有一个与命令名称相同的可执行文件。外部命令必须配置环境变量才能在任何地方运行。

### 查看系统的环境变量

命令：printenv (print enviroment)或 env

### 查看时间

在Linux中时间分为硬件时间和系统时间，在系统开机后读取硬件时钟后，Linux开始通过软件来实现时间的计算，这也就是系统时钟。

**硬件时钟命令**：clock/hwclock（hardware clock）

**系统时钟命令**：date

按指定格式输入：

硬件时间和系统时间可以单独修改

### 查看日历

cal [月份 年份] （如果省略年份，就是查看当前月日历）



### 获得命令帮助

Linux命令分为内部命令和外部命令

查看内部命令的帮助：

help 命令名



（外部命令不能使用 help 查看帮助）

查看外部命令：

获取简要帮助信息：命令名 --help



查看命令手册（内部命令和外部命令都可以使用）

man 命令名（manual）



命令手册分章节：（man命令后面可以加数字，表示查看手册哪一章节）

1. 用户命令
2. 系统调用（开发人员手册）
3. 库调用
4. 特殊文件（设备文件）
5. 文件格式

### 显示命令摘要

可以查看指定命令在命令手册中出现在第几章

whatis 命令名



### 查看指定名称的进程及进程标识符（pid）

ps aux|grep name

查看指定名称的进程



### 解压命令

tar -xvf 目标文件 -C 解压目录

### 开启指定端口

第一步（开放3306端口命令）：/sbin/iptables -I INPUT -p tcp --dport 3306 -j ACCEPT

第二步（将修改保存到防火墙中）：/etc/rc.d/init.d/iptables save

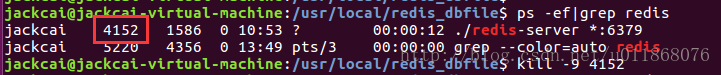
某些时候：

开启端口 iptables -I INPUT 4 -p tcp -m state --state NEW -m tcp --dport 3306 -j ACCEPT

保存至防火墙 service iptables save

### 关闭进程

ps -ef|grp redis 找到redis的当前进程号



kill -9  4152 强制关闭进程

### 查看端口开启状态

**命令：iptables -nvL**

### 内存数据同步命令（P143）

命令：sync （synchronized）

该命令所有用户都能执行，但是管理员执行时会同步所有用户的数据。

### 关机命令（P144）

poweroff

注：在执行关机命令时最好先执行sync命令（将内存中的数据同步到磁盘中）

### 查看系统IP

CentOS6 ifconfig

CentOS7 ip addr

### 切换用户

命令： **su – 用户名（switch user）**

注：1.切换用户时中间一定要打“-”

2.如果想要回到原来的用户模式中，即使用“exit”注销该账户即可

### 注销账户

命令：exit

### 切换终端界面

快捷键：Ctrl+Alt+F1…F6

### 启动图形界面

命令：startx

### 修改文件的所属用户组

（change group的缩写）

命令：chgrp [-R] 组名 目标文件/目录

R选项：进行递归修改，即连同子目录下的所有文件、目录都更新成这个用户组的意思。

### 修改文件的拥有者

change owner

命令：chown [-R] 账户名 目标文件/目录

R选项：进行递归修改，即连同子目录下的所有文件、目录都更新成这个账户的意思。

### 修改文件权限

**方式一：数字类型修改**

命令：chmod [-R] xyz 目标文件/目录

各权限数字对照表：

**r：4**

**w：2**

**x：1**

例如：将文件修改成 rwx r-x ---权限

chmod 750 haha.txt

**方式二：符号类型修改**

**符号类型修改常用于某个权限的修改与授予**

u 用户

g 用户组

o 其他

a 所有

1. **为三类权限授予指定权限**

**chmod u=rwx,go=rx hah.txt**

1. **如果想要删除某个权限呢？**

**例如：删除haha.sh 所有分类下的执行权限**

**chmod a-x haha.sh**

1. **如果要增加某个权限呢？**

**例如：给所属用户haha.sh的执行权限**

**chmod u+x haha.sh**

### 创建目录

命令： mkdir 目录名

选项： -p ： 层级创建目录（默认情况下需要一层一层的创建目录）（parents）

-m ：设置权限（mode）



注意：在Linux中在一个目录中文件和文件夹也不能同名

### 查看Path变量

命令：echo $PATH

### 关闭防火墙

命令：CentOS6：service iptables stop

CentOS7：sudo systemctl stop firewalld.service && sudo systemctl disable firewalld.service

CentOS7关闭防火墙问题：<https://blog.csdn.net/peterxiaoq/article/details/72897231>

### 文件基本操作

#### 复制

命令：cp [-option] 源文件 目标文件

**注意(重中之重)：**

1. **如果目标文件是一个已经存在的文件夹，复制后文件的文件名与原来相同。**

**（例如：cp haha.txt /usr/local，这样就会生成 /usr/local/haha.txt，因为最后一个斜杠后面的文字（local）是真实存在的，所以按照原来的文件名不变进行复制）**

1. **如果目标文件是一个不存在的名称，也就是最后有一级虚拟路径，那么复制后的文件名就是制定虚拟路径的名**

**（例如：cp haha.txt /usr/local/hello.txt（在/usr/local下没有hello.txt），这样会将haha.txt复制过来并重命名为hello.txt）**

1. **如果目标文件是一个多级虚拟路径，则会复制失败 。**



**因为在d2/a/b/c.txt中只有d2目录时存在的，此时属于多级虚拟路径范畴。**

1. **复制文件或目录一定要具有read权限**

**选项：**

-r （recursive）递归复制

-i （interactive 交互式的）若目标文件已经存在时，在覆盖时会先进行询问，看是否进行覆盖

-p （preserve 保留，保持）连同文件的属性（权限、用户、时间）一起复制过去，而非使用默认属性（备份常用）（-p选项等同于 --preserve=mode,ownership,timestamps）

-a 与“-dr --preserve=all”相同（--preserve=all 相对于-p有时保留的属性项更多一些）

-s (symbolic link )复制成为符号链接文件，类似于Windows中“快捷方式”

-d 若文件是链接文件，则复制链接文件，而不是链接文件所对应的真实文件。（默认情况下复制链接文件，就是复制链接文件所指向的真实文件，如果想要复制链接文件本身就需要加-d选项了）

#### 删除

命令：rm 文件或目录

选项：

-r 递归删除

-i (interactive)交互模式，每次删除会询问是否删除，在CentOS7中默认带有该选项

-f (force)忽略不存在的文件、不会出现警告信息，不会询问是否删除、

#### 剪切

命令：mv 源文件/文件夹 目标文件名/文件夹名

选项：

-i （interactive）若目标文件已经存在时，询问是否覆盖

-u 若目标文件已经存在，且source比较新，才会更新

注意：剪切兼顾重命名的操作

与cp原理一样，如果目标文件是一个级虚拟目录，那么最后一个斜杠后面的文字就是重命名的文件名，如果目标目录是真实存在的，那么就会将这个文件夹复制到那个目录下。

例：mv tjd aa 如果aa文件夹存在，那么将会将tjd目录移动到aa目录下，如果aa不存在，则就是将tjd复制到aa的上一级目录（aa的上一级目录就是当前的工作目录），复制后文件夹的名称为aa，这样也就达到了重命名的效果。

### 查看文件内容

#### 整体查看

##### cat命令（正向输出文件所有内容）

命令：cat [选项] 文件名

选项：

-n （number）打印出行号，空白行也会有行号

-b 打印出行号，仅针对非空白行做行号显示，空白行不标行号。

-E 将结尾的换行符用$显示

-T 将[tab]按钮以^I显示出来

-v 列出一些看不出来的特殊字符

-A 相当于-vET的整合选项

##### tac命令（反向输出文件所有内容）

tac命令是cat命令的反拼，命令的意义也与cat命令相反，就是由最后一行到第一行反向的输出在屏幕上。

命令： tac 文件名

#### 可翻页查看

##### more命令

命令：more 文件名

操作：

回车键：向下翻一行

空格键：向下翻一页

/字符串：在这个显示内容中，向下查找字符串这个关键字

:f：立刻显示出文件名以及目前显示的行数。

q：代表立即离开more，不再显示该文件内容

b或ctrl+b：（back）代表往回翻页

##### less命令

less命令相对于more命令更有弹性。

命令：less 文件名

操作：

pagedown键：向下翻一行

pageup键：向上翻一行

空格键：向下翻一页

回车键：向下翻一行

/字符串：向下查找字符串

?字符串：向上查找字符串

n：重复前一个查找（与/或?无关）

N：反向的重复前一个查找（与/或？有关）

q：离开less程序

#### 截取查看

##### head命令（截取前面几行）

命令：head [-n number] 文件名

选项：

-n：-n后面加数字表示显示几行的意思

**注：当数字位正数时表示显示前面n行，例如 head -n 100 xxx.txt 表示显示xxx.txt文件前面100行，当数字位负数时，表示不显示后面的n行。例如xxx.txt文件有131行，现在是使用 head -n -100 xxx.txt，表示显示前面的31行，后面的100行不显示。**

##### tail命令（截取后面几行）

命令：tail [选项] 文件名

选项：

-n：后面接数字，代表显示几行的意思

**-f：表示持续刷新显示后面所接文件中的内容，按Ctrl+C结束刷新（常用于查看日志文件）**

### 文件的默认权限（P195）

**查看umask（查看默认权限）：**

命令：umask 或 umask -S

前者以数字的形式展示（与一般权限有关的是后面三个数字，第一个数字是给特殊权限）



umask显示的数字是**默认最大权限**需要减掉的权限

**默认最大权限是什么**：默认最大权限在创建文件和文件夹时是不一样的。

创建文件默认最大权限：rw\_rw\_rw\_(666)

创建文件夹默认最大权限：rwxrwxrwx（777）

后者以符号类型方式来显示权限：



**修改umask（修改默认权限）：**

命令：umask 数字



### 文件的隐藏权限（P197）

**查看文件的隐藏权限**

命令：lsattr [选项] 文件或目录名

选项：

-a 隐藏文件的属性也显示出来

-d 如果接的是目录，仅列出目录本身的属性而非目录内的文件名

-R 连同子目录的数据一并列出来



**修改文件的隐藏属性**

命令：chattr [选项] [+-=][属性] 文件或目录名

选项：

-R 递归设置指定目录下所有文件的隐藏属性

属性：

i 它可以让一个文件**不能删除，改名，设置链接也无法写入数据或新增数据**，如果试图删除则会提示权限不足，对系统安全性有相当大的帮助。**只有Root能设置此属性。**

a 当设置a后，这个文件**只能增加数据**，而不能删除也不能修改数据，只有root才能设置该属性。

c 设置c属性后，将会自动的将此文件**压缩**，在读取的时候会自动解压缩，在存储时会自动压缩后再存储进去。

s 当文件设置了s属性后，如果该文件被删除了，它将会完全的从磁盘中删除，如果误删完全没法恢复。

u 与s相反，当使用u来配置文件时，如果该文件被删除了，则数据内容还存在磁盘中，可以恢复此文件。

S 一般文件是非同步写入磁盘的，如果加上S（大写）这个属性时，当你对文件做任何修改，该修改会**同步**到磁盘当中去。

例：

### 查看文件类型

命令：file 文件名

该命令用于查文件的基本信息，例如属性ASCII或是数据文件或是二进制文件。



### 查找可执行文件

命令：which [选项] 执行文件

选项：

-a 将所有Path目录中可以找到的命令均列出，而不只是列出第一个找到的命令



### 查找文件

**whereis命令**

命令：whereis [选项] 文件或目录名

选项：

-l(小写) （list）可以列出whereis回去查询的几个主要目录

-b 只查找binary（二进制）格式的文件

-m 只找在说明文件manual路径下的文件

-s 只找source源文件

-u 查找不在上述三个项目当中的其他特殊文件



**locate命令**

命令：locate [选项] keyword

选项：

-i 忽略大小写差异

-c 不输出文件名，只显示找到的文件数量

-l 仅输出几行的意思，-l 5 代表仅输出5行

-S 输出locate所使用的数据库文件的相关信息，包括该数据库记录的文件/目录数量等。

-r 后面可接正则表达式的显示方式

### 文件压缩及打包命令

#### 常用的压缩文件扩展名

\*.Z compress程序压缩

\*.zip zip程序压缩的文件

\*.gz gzip程序压缩文件

\*.bz2 bzip2程序压缩文件

\*.xz xz程序压缩文件

\*.tar tar程序打包文件（并没压缩）

\*.tar.gz tar程序打包文件，并且经过gzip程序压缩

\*.tar.bz2 tar程序打包文件，并且经过bzip2程序压缩

\*.tar.xz tar程序打包文件，并且经过xz的压缩。。

#### gzip，zcat/zmore/zless/zgrep压缩命令（P264）

gzip可以说是应用最广泛的压缩文件了，目前gzip可以解开compress，zip和gzip等软件所压缩的文件。gzip压缩的文件在windows中也可以通过解压软件进行解压。

**命令**：gzip [选项] 文件名

**选项**：

-c 将压缩的数据输出到屏幕上，可以通过数据流重定向来处理

-d 有-d选项代表进行解压操作

-t 可以用来检验一个压缩文件的一致性，看看文件有无错误

-v 可以显示原文件/压缩文件的压缩比等信息。

-# #为数字的意思，代表压缩等级，-1最快，但是压缩比最低；-9最慢，但是压缩比最高，默认是-6

**注意：**

1. gzip压缩命令只能压缩单个文件，不能压缩文件夹。
2. 我们可以通过zcat/zmore/zless等快捷查看压缩文件内容，这样就不需要解压后再看了。（前提是压缩前就是一个文本文件，这些命令的使用与cat,more,less类似）

**示例：**

压缩示例：



上面的压缩方法压缩后原文件就没有了，并且压缩后的名字是固定的（\*.gz），如果我们需要打破这个习惯（压缩后保留原文件并且压缩文件名自己定），可以这么做：（详见P264）



解压示例：



#### bzip2，bzcat/bzmore/bzless/bzgrep压缩命令（P265）

如果说gzip是用于取代compress，那么bzip2就是用来取代gzip命令的，并且提供更好的压缩比。bzip2

的用法几乎与gzip相同。

命令：bzip2 [选项] 文件名

选项：

-c 将压缩的数据输出到屏幕上，可以通过数据流重定向来处理

-d 有-d选项代表进行解压操作

-k 保留原文件，而不会删除原文件

-z 代表进行压缩操作（默认值，可以不加）

-v 可以显示原文件/压缩文件的压缩比等信息。

-# 与gzip一样，都是计算压缩比的参数，-9最佳，-1最快

#### xz，xzcat/xzmore/xzless/xzgrep压缩命令（P265）

虽然bzip2已经有很好的压缩比了，不过显然某些自由软件开发者还不满足，因此还推出了xz这个压缩比更高的软件。这个软件的用法也跟gzip/bzip2几乎一模一样。

**命令**：xz [选项] 文件名

**选项**：

-d 代表解压缩

-t 测试压缩的完整性，看有没有错误

-v 可以显示原文件/压缩文件的压缩比等信息。

-l 列出压缩文件的相关信息

-k 保留原文件不删除

-c 将压缩的数据输出到屏幕上，可以通过数据流重定向来处理

-# 与gzip一样，都是计算压缩比的参数，-9最佳，-1最快

注意：xz软件的缺点也非常明显，为了追求极致的压缩比，xz软件相对于gzip和bzip2相比速度慢了很多，几乎有10倍的差距（不同场景时间差距不同）。

#### 打包命令：tar（重点理解）

**命令：**

**打包与压缩**

tar [ -z | -j | -J ] [ c v ] [-f 待建立的新文件名] 要被压缩的文件或目录名



**查看文件名**

tar [ -z | -j | -J ] [ t v ] [-f 既有的tar文件名]



**解压缩**

tar [ -z | -j | -J ] [ x v ] [-f 既有的tar文件名] [-C 解压到的目录]

（如果不指定解压的目录，那么默认解压到当前目录下）

选项：

-c 建立打包文件，可搭配-v来查看过程中被打包的文件名

-x 解包或解压缩的功能，可以搭配-C（大写）指定解包/解压缩的目标目录（-c -t -x不能同时出现在一串命令行中）

-t 查看打包文件中含有哪些文件名

-z 通过gzip的支持进行压缩或解压缩（\*.tar.gz）

-j 通过bzip2的支持进行压缩或解压缩（\*.tar.bz2）

-J 通过xz的支持进行压缩或解压缩（\*.tar.xz）（-z -j -J不能同时出现在一个命令行中）

-v 在压缩/解压过程中，把正在处理的文件名显示出来

-f -f后面要立刻跟上被处理的文件名，建议-f单独写一个选项（这样不容易忘记）

-C 这个选项用于解压缩，若要在特定目录下解压缩，可以使用这个选项

-p（小写） 保留备份数据的原本权限与属性，常用于备份（-c）重要的配置文件

**-P（大写） 保留绝对路径，亦即允许备份数据中含有根目录存在之意（详见P268）**

--exclude=FILE 在压缩过程中不要将FILE打包；

**注：**

1. 使用tar搭配gzip等压缩软件的使用，可以实现windows下整个文件夹压缩的效果，只需要在待处理文件中写文件夹名即可，解压后的文件仍然是在文件夹中的。
2. 使用tar搭配gzip等压缩工具进行打包压缩，不会删除原文件（与windows中的压缩效果类似）
3. **使用tar进行压缩时，压缩后的文件是携带压缩时指定的路径的。（重点理解）**

例如我将/etc目录复制到root目录下的myDir目录，如果此时我在root目录下对复制过来的etc进行打包压缩（tar -zx -f etc.tar.gz myDir/etc），和我先进入myDir在打包压缩etc（cd myDir；tar -zx -f tar -zx -f etc.tar.gz etc）这两种压缩方式有什么区别。



不管以哪种方式进行压缩器压缩的文件都是一样的，都是压缩的etc下面的所有文件，但是两者压缩时指定的目录时和不同的，一个是 myDir/etc 另个一个是etc，所以压缩的文件中也带有路径，所以前者的压缩文件的文件名前面会加上myDir/etc/这个路径，而后者压缩文件中的文件名都会加上etc/这个路径，从表象上看就是前者解压后的文件是myDir，后者的压缩文件解压后是以etc开头的。由于是相对路径，所以解压后由于文件前面带有路径，**系统会先找有无这些目录，如果有将解压后的文件放进这个目录中去，如果没有则会新建这些目录**。（在Windows上面也是这样）。这也就是注1中效果的实现原理，压缩后的文件仍然带有压缩路径，解压时按照压缩路径进行解压。

这样又会带来一个问题，那如果我们的压缩路径是一个绝对路径呢？Linux为了系统安全会在压缩时**默认不会携带绝对压缩路径，会携带去掉绝对压缩路径的第一个斜杠后的路径**。如果默认携带绝对路径的话那么压缩文件不管在哪个文件夹中解压，都会尝试覆盖绝对路径对应的文件，这样很多情况下会产生极大问题，例如下面这个情况。

例如我们在myDir中执行 tar -zx -f etc.tar.gz /etc会出现这么一句话



如果不删除，不管etc.tar.gz在哪里解压，都会尝试覆盖/etc下面的文件，那么岂不是会导致相当多的问题。

如果我们想在压缩时给文件带上绝对路径，那么我们就需要加上-P选项。

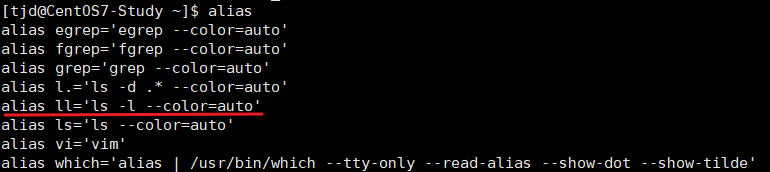
### 系统命令的别名（P334）

#### 查看系统别名

在系统中有很多命令的别名，例如ll就是ls -l 的别名，在CentOS7中vi是vim的别名（也就是说使用vi命令实际上是使用vim），那我们如何查看这些别名呢？

**命令**：alias

**示例**：



**注：**

我们可以通过alias看是否有“alias vi=‘vim’”的字样，如果有就代表在当前系统使用vi实际上是在使用vim。

#### 添加命令别名

**命令**：alias key=’命令’

例如：



这样我们在执行lm时，相当于执行ls -al

#### 删除别名

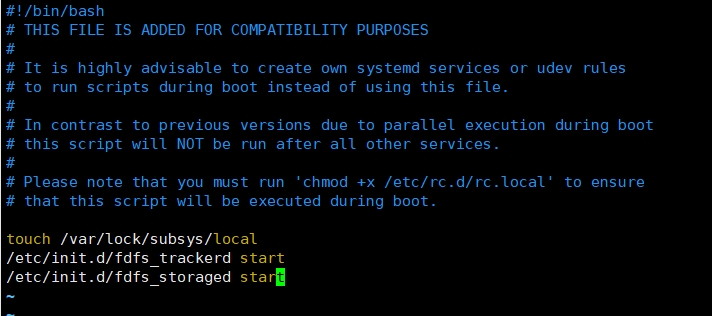
**命令**：unalias 别名



### 设置开机自启

开机自启只需要在一个配置文件中写上相应的启动命令即可：

**命令：vim /etc/rc.d/rc.local**



## VI与VIM编辑器

vi/vim编辑文件时，若不正常退出会在当前目录下生成一个.文件名.swp的临时文件，若这个文件不删除每次编辑这文件的时候就会提示“文件恢复”的消息。

### 一般命令模式

补充：

w 向右跳到下一个关键字







### 进入编辑模式

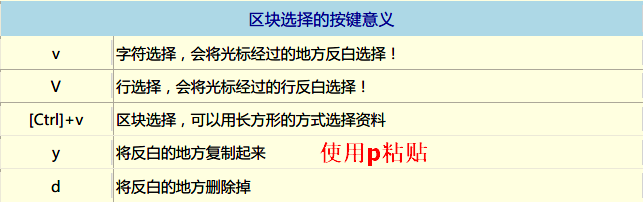


### 命令行模式



### VIM相对于VI的独特功能

**1. 可视区块（Visual Block）（P300）**



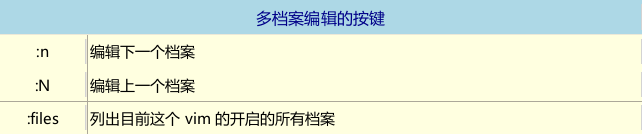
（按esc退出该模式）

**2. 多文件编辑（P301）**

如果我们需要将A文件中的数据拷贝到B文件制定位置中我们就可以使用多文本编辑，也就是一次性编辑多个文本。**可以在vim命令后接多个文件名，表示同时编辑多个文件**。



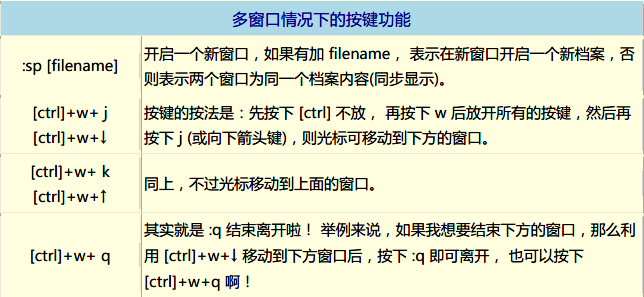
在多文件编辑时，相关按键有：



**3. 多窗口功能**

在一般窗口界面的编辑器中大多有“划分窗口”，“冻结窗口”的功能来将一个文件划分成多个窗口展示。那vim如何实现这个功能呢？

很简单只需要在命令行模式下输入“:sp 文件名”，其中文件名可以省略，如果省略就是将本文件进行多窗口展示。



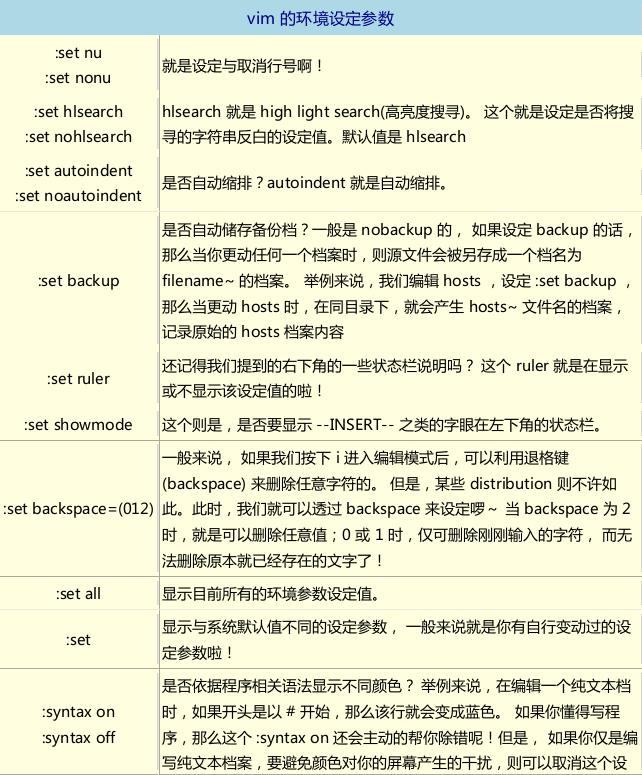
**按住Ctrl，双击w，就可以切换窗口。**

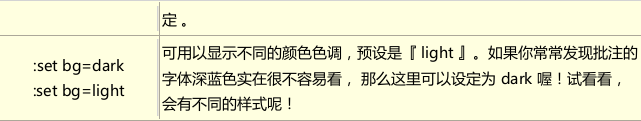
1. **vim关键字补全功能（P303）**

vim编辑器自带关键字补全，在插入模式下，使用下列组合键即可完成代码补全。

|  |  |
| --- | --- |
| 组合键 | 补齐内容 |
| ctrl+x ——> ctrl+n | 通过目前正在编辑的**文件内容**文字作为关键字补全 |
| ctrl+x ——> ctrl+f | 以当前目录内的文件名作为关键字进行补全 |
| ctrl+x ——> ctrl+o | 以扩展名作为语法补充，以vim内置的关键字，予以补充 |

1. **vim的环境参数设置（P304）**

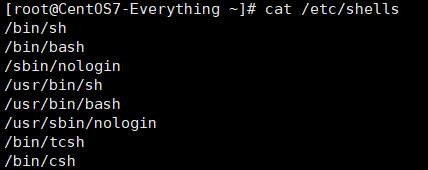




## Linux学习之Shell

### 查看当前系统所安装的Shell版本

直接查看 **/etc/shells**文件即可



### 查询命令是否为Bash Shell的内置命令

命令：type 命令名

### 查看当前正在运行的Shell

echo $SHELL

### 删除变量

命令：unset 变量名

### 打印变量

命令：echo $变量名

### 查看系统信息

命令：uname [选项]

选项：

-a 显示所有信息

-m 显示机器（硬件）类型

-n 显示机器网络节点主机名

-r **显示操作系统内核版本**

-s 显示操作系统名

-p 显示主机处理器类型

-v 显示操作系统版本

### 查看环境变量名

命令：env（environment）

### 查看所有变量名

命令：set

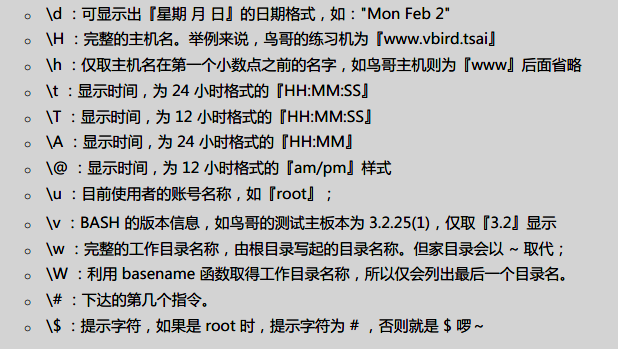
### 常见的环境变量

1. HOME：代表用户的根目录 cd ~命令就是使用的这个变量
2. SHELL：表示目前环境中使用的Shell是哪个程序
3. HISTSIZE：这个与历史命令有关，决定记录的历史命令条数（上一次登录时使用的历史记录，在家目录中.bash\_history中记录）
4. MAIL：当我们使用mail命令收信时，系统会去读取的邮箱文件。
5. PATH：执行文件的查找路径
6. LANG：当前系统的语系数据。
7. RANDOM：随机数变量。目前大多数Linux发行版都会有随机数生成器，那就是/dev/random文件。我们可以通过这个随机数文件相关的变量（RANDOM）来随机取得随机数值，在BASH环境下，这个RANDOM变量的内容，介于0~32767之间

### 常见的非环境变量

1. PS1：提示字符的设置

这个变量决定每次输入命令之前，提示符的格式：



CentOS7默认是：

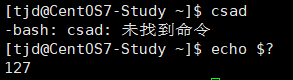
1. $ （本Shell的PID）

$本身也是一个变量，表示当前正在执行的Shell的进程PID



1. ？（上个执行命令的返回值）

问号本身也是Bash中的一个变量，它表示上一次命令的返回值，通常执行成功返回值是0。



### 将自定义变量转成环境变量

命令：export 变量名

### 开启一个子Bash程序

命令：bash

退出当前Bash：exit或logout

可以通过 echo $$ 查看当前正在执行的Bash的进程号

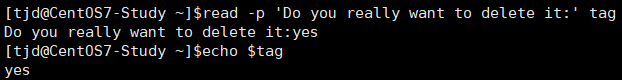
### 读取来自键盘输入的变量

命令：read [选项] 变量名

选项：

-p：后面可以接提示符

-t：后面可以接等待的秒数。



### 声明变量

命令：declare [选项] 变量名

选项：

-a：定义为数组类型(array)

-i：定义为证书类型(integer)

-x：定义为环境变量

-r：将变量设置为readonly类型，该变量不能更改内容，也不能unset

-p：列出变量的类型（通常只会单独使用，用于查看变量的类型）

注：

1. 在Bash中，变量默认为字符串类型，Bash中的数字运算默认只能到达整数形态。

数组形式的读取必须采用 ${var[index]}的形式读取，如果直接$var[index]是错误的。

1. 在Bash中变量其实无需声明，直接使用即可。

### 历史命令

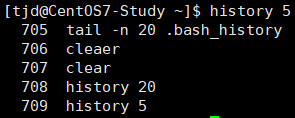
命令：history

选项：

n 数字，列出最近的几条命令的意思，如果不指定则列出所有

-c 将目前shell中的所有history内容全部清除。（清除的是内存中的历史记录）

-w 将目前history记录的数据写入histfiles中（默认为~/.bash\_history）



注：

1. histfiles文件中能保存多少条记录取决于HISTFILESIZE变量
2. 默认情况下历史记录会保存在内存中，只有在当前Shell注销时才会写入~/.bash\_history文件中
3. **调用历史命令：**

格式一：!number （执行第number条历史命令）

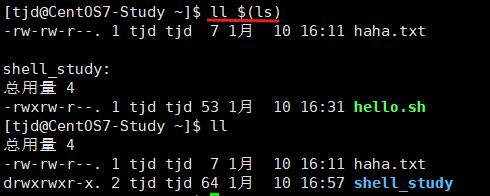
格式二：!command （由最近的命令向前找，找到以command开头的命令并执行）

格式三：!! （执行上一条命令）

### 执行子命令

即在一条命令中先执行子命令，然后在执行其他命令

**格式**：$(子命令) 或 $`子命令`（返单引号，数字1的左边）

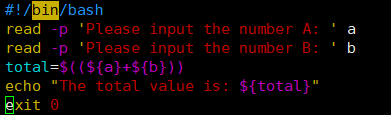


在ll $(ls)是先执行ls命令，将ls命令作为ll命令的参数，所以上述命令的效果就是查询当前目录所有文件夹中文件名。

### 执行数值运算（P393）

shell默认情况下只能进行整数运算。

**方式一：$((计算式)) 常用**



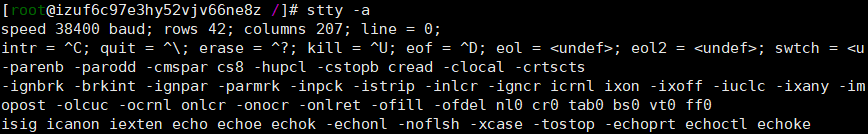
方式二：declare -I total=${a}+${b}

### 终端环境的配置（快捷键设置 P342）

命令：stty [选项]

选项：

-a 将所有的stty参数列出来



几个关键词的意义：

intr：发送一个interrupt（终端）信号给正在运行的程序（就是终止）。

quit：发送一个quit信号给正在运行的程序。

erase：向后删除字符（^?代表的是backspace）

kill：删除命令行上的所有文字。

eof：代表结束输入的意思。

stop：停止目前的屏幕输出。

start：在某个程序停止后，启动它的output。

susp ：送出一个terminal stop的信号给正在运行的程序。

**设置快捷键**：

将退格操作设置为ctrl+h

常用快捷键：

ctrl+c 终止当前程序

ctrl+u 删除当前输入的命令

### \*\*配置文件的加载顺序\*\*

bash的环境配置文件分为全局配置文件和用户配置文件两种。

登录时加载顺序总体分为两步：

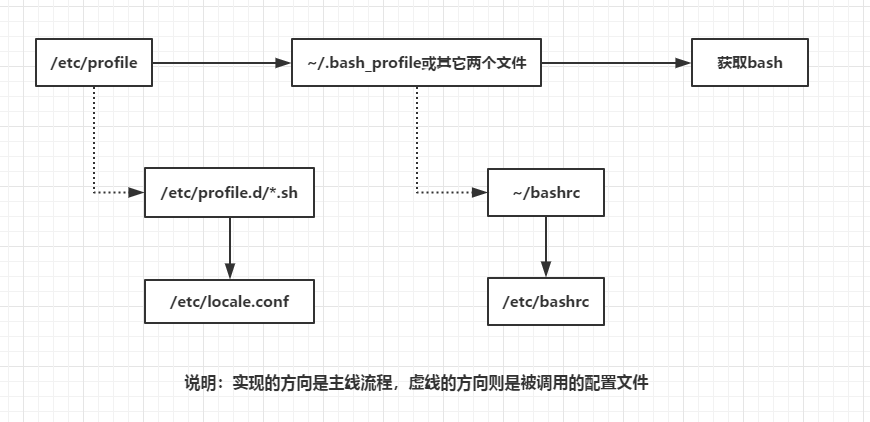
**第一步：加载全局配置文件（/etc/profile）**

系统在加载/etc/profile时还会调用其他的配置文件：

* /etc/profile.d/\*.sh：只要是/etc/profile.d文件夹下所有可读的\*sh文件都会执行，这个文件夹下的文件规范了操作界面的颜色、语系、ls命令的别名、vi命令别名、which命令别名等。如果需要添加全局的命令别名只需要在这个目录下添加一个后缀为sh的文件即可。
* /etc/locale.conf：这个文件是/etc/profile.d/lang.sh调用的，这也是我们决定bash默认使用何种语系的重要配置。文件中最重要的的就是LANG/LC\_ALL变量的配置。

**第二步：加载用户配置文件（~/.bash\_profile或~/bash\_login或~/.profile）**

其实bash的login shell只会加载其中的一个，而读取顺序就是按照上面的顺序，如果读取到一个后面的就不会读取了。



### \*\*执行脚本的三种方式的区别\*\*

执行脚本有下列三大种方式：

* ./脚本文件
* bash 脚本文件 或 sh 脚本文件
* source 脚本文件 或 . 脚本文件（点后面有个空格，然后再接脚本文件名）

三种方式的区别：

|  |  |  |  |
| --- | --- | --- | --- |
|  | **./脚本文件** | **bash 脚本文件** | **source脚本文件** |
| **是否在子进程中运行** | **是，执行脚本文件时会创建一个子bash进程，所以以此种方式运行的脚本文件中通过export导出的全局变量无效** | **是** | **否，此种方式运行脚本是直接在父进程bash中运行的，所以脚本中export导出的全局变量会生效** |
| **是否需要文件有x权限** | **需要** | **不需要** | **不需要** |

### 重新加载系统配置文件

通常我们在安装JDK后，在/etc/profile中配置环境变量后不会立即生效，那如果想立即生效在此之前需要注销当前Shell。其实我们可以使用source 来重新加载系统的配置文件。

**命令：**

**source 文件名**

**. 文件名**

**两者效果相同，第二种方式中，点和文件名必须用空格隔开**



**注：**

**登录时系统会先加载/etc/profile文件，然后加载~/.bash\_profile文件。/etc/profile是全局的配置文件，如果所有想要所有用户都生效，那么就配置在/etc/profile下；如果只需要当前用户有效，那么就配置在~/.bash\_profile文件中即可。**

**profile实际上就是一个Shell脚本，但这个Shell脚本必须使用source命令执行，不能使用bash sh等命令执行，具体原因见P395。**

**source命令只是单纯的执行这个脚本文件，并不是通知系统重新加载配置文件，所以我们如果同时更改了全局配置文件和用户配置文件，那我们就需要通过source命令分别加载。**

### 数据量重定向（P345）

**标准输入（stdin）：代码为0，使用：** < 或 <<

（将原本需要由键盘输入的数据，改由文件内容来替换）

**标准输出（stdout）：代码为1，使用：** > 或 >> 或 1> 或 1>>

**标准错误输出（stderr）：代码为2，使用：** 2> 或2>>

（当用户输入错误命令时输出的错误提示就是标准错误输出，只能使用2>>或2>重定向指定文件，不然不起作用）

**使用格式示范**：基本命令 >> 文件名

表示将基本命令的结果输出到指定文件中，如果文件不存在则自动创建文件。>表示覆盖原文件内容，>>表示在原文件内容后面追加。

**标准输出的重定向：**





**标准错误输出的重定向：**



**如果需要将标准输出和标准错误输出都重定向同一个文件中，该怎么做？**

**错误**示范：

正确语法：

2>&1 意思是将代码为2的 标准错误输出 转成 标准输出（这句话要放在句尾，表示在这个命令中，标准错误输出转到标准输出中去）

**例题：理解下面命令的意思**



这句命令的意思就是，将echo ‘dsadasd’命令产生的标准错误输出重定向到/dev/null中去，但是echo … 这个命令本身不会产生错误，所以按理说，输出的数据不会被 2>> 重定向到/dev/null中去，但是最后有一句1>&2，意思是将标准输出 转到 标准错误输出中去，所以原先走标准输出的数据开始走标准错误输出了，所以被2>>捕获了。

**Linux的黑洞设备（/dev/null）**

如果我们预知一些命令会有错误信息，但我们既不想看见它，也不想把它重定向到一个文件中，那我们我们可以将错误信息通过 2>> 重定向到Linux提供的垃圾桶黑洞设备。/dev/null可以吃掉任何导向这个设备的信息。



### Linux中逻辑运算符（P348）

在Linux中也有逻辑运算符：&& ||

在Linux中逻辑运算表达式都是从左往右执行的，**没有编程语言中的优先级一说。**

例1：cmd1 && cmd2

表示cmd1执行成功后才会执行cmd2。

例2：cmd1 || cmd2

表示cmd1执行成功了，就不会执行cmd2。

例3：ls /tmp/abc || mkdir /tmp/abc && touch /tmp/abc/haha.txt

表示先判断/tmp/abc这个文件夹在不在，如果不在就创建该文件夹，然后再创建haha.txt文件。

**注：**上一个命令执行成不成功系统用名为：？的变量表示，当$?==0时表示执行成功，当$?!=0时表示执行不成功。

### 管道命令（pipe）

管道命令是将前一个命令的标准输出作为后一个命令的标准输入。（什么是标准输入和标准输出查看P346）

注意：

1. 管道命令只能处理标准输出，对于前一个命令的标准错误会予以忽略。
2. 管道命令的后面的命令必须是可以接受来自前一个命令的数据成为标准输入继续处理才行。
3. 如果非要让管道命令处理 标准错误输出，那么需要在命令后面加上 2>&1，让标准输错误输出转到标准输出中去。（详见数据流的重定向）

例如：

如果我们想查看/etc下面有哪些文件，但是文件实在太多了，我们可以采用管道命令进行处理：



### 行选取命令grep（P352）

grep是分析一行中的内容，若当中有我们所需要的信息，我们就将这一行拿出来

**命令：grep [选项] ‘查找的字符串’ 文件名**

选项：

-a 将二进制文件以文本形式进行查找

-c 计算找到的 ’查找字符’ 的个数

-A 后面可加数字，表示除列出该行以外，后续的n行也列出来

-B 后面可加数字，表示除了列出该行以外，前面的n行也列出来

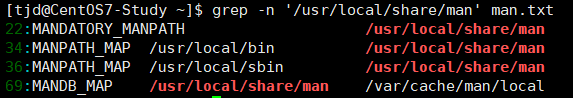
-i 忽略大小写的不同

-n 输出行号

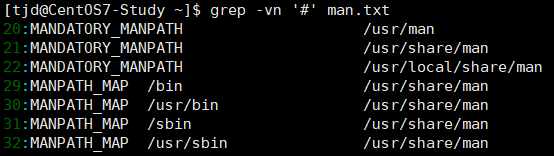
-v 反向选择，也就是显示出没有 ‘查找字符’ 的行

--color=auto 将找到的字符用其他颜色显示（在CentOS7中，使用alias别名将这个选项默认加进去了）

查看包含指定内容行：



查看不是注释的行：



### 划分命令split

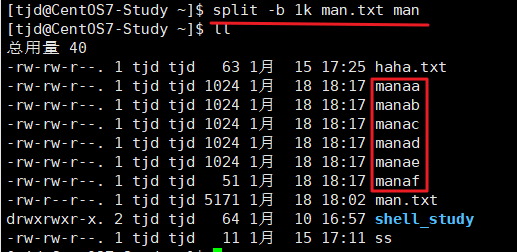
如果你有文件太大，导致携带不太方便的话，用split命令可以将一个大文件，依据文件大小或行数划分成过个小文件。

命令：split [选项] 文件名 划分后的文件名前缀

选项：

-b 后面可接欲划分成的文件大小，后面可接单位（b,k,m等）

-l (小写L)以行数进行划分



### 正则表达式（P366）



### 脚本的执行方式差异（P394）

**执行方式：**

1. **source scripts**
2. **sh scripts**
3. **bash scripts**
4. **./scripts**

方式2-4都是开启一个子进程来执行脚本。如果使用2-4方式的命令执行脚本，哪怕是在程序中export了一个变量，在程序执行完毕后，在父程序中也是无法找到export的全局变量，因为export是在子程序中使用的，只是成为了子程序的全局变量，但是脚本执行完毕后子进程就关闭了，所以就没有效果了。

那为什么子进程中可以访问父进程的全局变量呢，实际上是因为子进程会复制父进程的全局变量，但仅仅是复制，所以在**子进程中改变父进程中继承的全局变量时，在父进程中是没有效果的**。

方式1不会创建子进程执行脚本，而是直接在当前进程中执行脚本，这也是为什么我们修改/etc/profile文件后（例如添加JDK的环境变量）,需要使用source /etc/profile来重新加载配置文件，使得Path环境变量的修改生效。

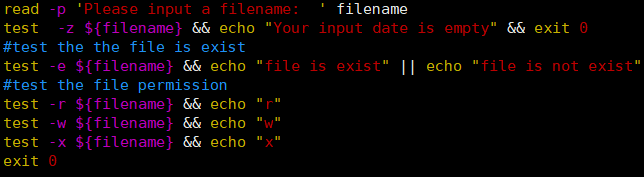
### test命令进行条件判断（P397）

**命令：test [选项] xxx （详见最后的例题）**

（档名即文件名，档案就是文件的意思）



**示例：输入一个文件名判断这个文件在当前路径下是否存在，如果存在输出文件的权限。**



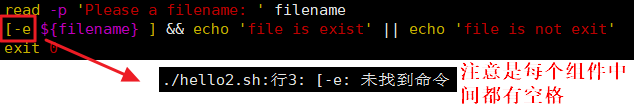
### 中括号[]进行条件判断（P397）

中括号进行条件判断的语法基本上与test一模一样，就是将test换成了一对中括号，常用于if..then语句中使用。



**使用中括号进行条件判断的注意事项：**

1. **在中括号[]内每个组件都需要有空格分隔。**



1. **在中空号内的变量都最好以双引号括起来。**

因为如果不加引号，如果字符串中包含空格，那么就会把这个字符串当成两个参数，但是很多操作符只能接收一元参数，此时就会报错。



**我们应该把代码写成这样：**



**输出：**



1. **在中括号中的常数最好以单引号或者双引号括起来。**

### Shell脚本的默认变量，解决脚本参数问题

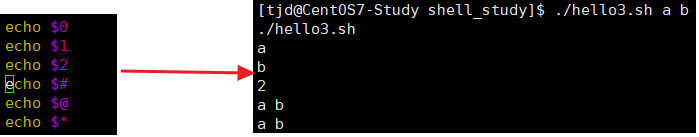
通常我们使用的命令后面都会接一个或读个参数，那么Shell脚本是如何获取用户执行脚本时所带的参数呢？

**Shell脚本针对参数已经设置好了一些变量名称，具体如下：**



**特殊的变量：**

1. **$# 代表后接的参数个数，以上面为例，这里显示4**
2. **$@ 代表【"$1””$2””$3””$4”】的意思，每个变量会独立的用双引号括起来**
3. **$\* 代表【"$1c$2c$3c$4”】，其中c为分隔符，默认为空格，所以在本例中代表【”$1 $2 $3 $4”】**



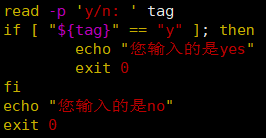
### if then语句（P400）

**单分支：**

if [ 条件判断 ]; then

分支体

fi



**多分支：**

if [ 条件判断 ]; then

分支体1

elif [ 条件判断 ]; then

分支体2

else

分支体3

fi

**注意事项：**

1. 条件判断中括号后面的分号不能遗漏
2. 中括号中的各个组件必须用空格隔开
3. 最后的分支体需要你用fi结尾(if的反序)

### case … esac语句

shell中的case…esac语句类似于java中的switch语句。

语法格式：

case $变量名 in

“第一个变量内容”)

程序段

;;

“第二个变量内容”)

程序段

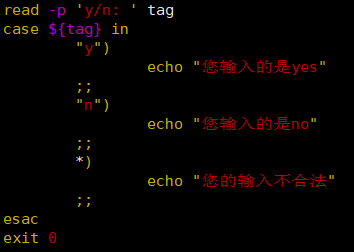
;;

\*）

未匹配前面某一个程序段时才执行这个

；；

esac



## 账户管理与ACL权限设置

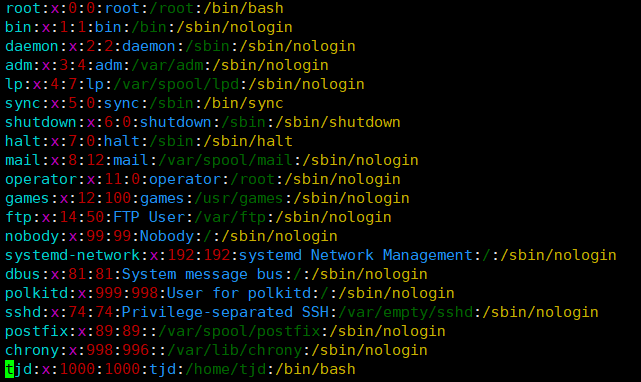
### 用户标识符：UID和GID

每个用户登录后至少包含两个ID，一个是UID（User ID），另一个是GID（Group ID）。Linux进行权限验证的时候实际上就是根据UID和GID来确定的，用户名和用户所在的组名都是给人看的。

### 用户账号相关文件理解

#### /etc/passwd文件结构：

passwd文件中存放的是账户的一些基本信息：账户名、用户ID、组ID、家目录、shell



**每一行代表一个用户，其中有很多用户是系统正常运行所必须的，我们可以称之为系统用户。**

**文件结构：**

**每一行中间使用6个冒号（：）隔开。**

1. **账户名称**
2. **密码**

密码早期的Unix系统的密码就是放在这个字段上面的，但因为这个文件可以被所有程序读取，所以将其放入到了/etc/shadow文件上面。

1. **UID**
2. **GID**
3. **用户信息说明栏**

这一栏无特定作用

1. **家目录**
2. **shell**

当用户登录系统后会获取一个shell来与系统的内核沟通，这个部分就是指定这个用户默认的shell。

#### /etc/shadow文件结构：

shadow文件中放置的都是账户的一些限制信息：密码、账户过期时间、密码修改的限制时间、密码的过期时间等



1. **账户名称**
2. **密码**
3. **最近修改密码的日期（从1970年1月1日开始算1，每过一天加1）**
4. **密码不可被修改的天数**
5. **密码需要重新修改的天数**
6. **密码需要修改期限前的提示天数**
7. **密码过期后的账户宽限时间（密码过期后账户可以进入，但是强制更改密码）**
8. **账户失效日期（也是从1970年月1日算起，超过指定日期后密码即作废；通常用于收费系统中）**
9. **保留字段**

#### 查询/etc/shadow中“密码”的加密机制

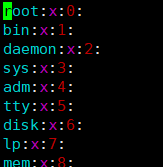
**命令：**authconfig –test | grep hashing



### 用户组相关文件理解

#### /etc/group文件结构（P424）

/etc/group记录该系统所有用户组的配置：



每一行代表一个用户组，每一行用冒号分割成4各部分：

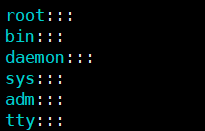
* **第一个部分：组名**
* **第二个部分：用户组密码**
* **第三个部分：GID（用户组对应的GID）**
* **第四个部分：此用户组支持的账号名称**。一个用户组可以有多个用户加入，多个用户之间由逗号隔开。

举例来说：如果我想让dmtsai与alex用户加入root这个用户组，那么在第一行的后面加上“dmtsai,alex”即可，使其变成：root:x:0:dmtsai,alex。

**注意：在新版Linux中，初始用户组的用户群已经不会加入第四个字段。**

#### /etc/gshadow文件结构

该文件最大的作用是指定用户组的管理员



* 第一栏：组名
* 第二栏：密码栏，密码栏是!或为空表示该用户组不具有用户组管理员，只能通过root用户来管理用户组。
* 第三栏：用户组管理员的账号
* 第四栏：加入该用户组的账号（与/etc/group内容相同）

#### 有效用户组与初始用户组

**初始用户组：**所谓用户的初始用户组就是在/etc/passwd文件中的第四栏中写的GID，当用户一登录系统，立刻就会拥有这个用户组的相关权限。

**有效用户组：**“有效用户组”决定当前用户在创建文件时，该文件属于哪个组。通常情况下有效用户组就是用户登录时，从/etc/passwd的第四栏中读出的GID所对应的用户组。

**通过groups命令可查看当前用户的有效用户组是什么，输出的第一个就是有效用户组**

#### 查询当前用户所加入的 用户组 和 有效用户组

**方式一：**

命令：groups



**第一个用户组表示当前用户的有效用户组**

**方式二：**

查看/etc/group文件

#### 有效用户组的切换（P425）

命令：newgrp 组名

注意：你想要切换的组，必须是你已经有支持的组。

### 账号管理

#### 新增用户（useradd P426）

命令：useradd [-u UID] [-g 用户组名] [-G 次要用户组] [-mM] \

参数：

-u 直接指定一个特定的UID给这个账户，该信息放在/etc/passwd 第三个档位

-g 指定初始用户组，该信息会放到/etc/passwd第四个档位

-G 指定用户的次要用户组，该信息会放到/etc/group文件 第四个档位

-m 强制，不要建立使用者家目录

-M 强制，要建立使用者家目录

-c 指定账户的说明情况，该信息保存在/etc/passwd 第五个档位

-d 指定某个目录为当前用户的家目录（务必使用绝对路径）

-s 指定用户的登录时的初始shell，默认情况下是/bin/bash

使用useradd命令创建账户后，系统会根据你提供的信息或者是默认值，在/etc/passwd、/etc/shadow、/etc/group、/etc/gshadow文件中添加新的账户信息，并自动创建家目录（系统自动复制/etc/skel目录到home文件夹作为新用户的家目录）。但是在/etc/shadow中不会有密码，所以该账户此时无法登陆，需要手动设置密码。

在使用useradd命令创建用户时，系统至少会参考：/etc/default/useradd文件（账户创建所需的默认值，详情请参考P428）、/etc/skel目录（账户创建时家目录的骨架）。

#### 为新账户添加密码（passwd）

命令： passwd [--stdin] [账户名称] 所有人均可使用更改密码（如果不指定账户名称就是更改自己的密码）

选项：

--stdin：通过来自前一个管道的数据来作为密码输入。

例如：echo "980613" | passwd test

#### 删除用户（userdel）

命令：userdel [-r] 用户名

选项：

-r ：连通用户家目录一起删除

该命令用于删除用户的相关数据，而用户数据有：

* 用户账号/密码相关参数：/etc/passwd、/etc/shadow
* 用户组相关参数：/etc/group、/etc/gshadow
* 用户个人文件数据：/home/username、/var/spool/mial/username

### 用户组管理

#### 新增用户组（groupadd）

命令：groupadd [-g gid] [-r] 用户组名称

选项：

-g：指定当前用户组的GID

-r：建立系统用户组

#### 删除用户组（groupdel）

命令：groupdel 用户组名称

注意：用户组只有在没有成员时才能被删除，如何查看一个用户组有没有成员，我们只需要**查看/etc/group的第四栏中成员**，以及**/etc/passwd文件中第四栏GID有没有指向这个用户组，如果没有就表明这个用户组中没有用户**。

#### 用户组管理（gpasswd）

**使用root用户管理组**

通常情况下，用户组只能通过root用户来管理（成员的增、删）。我们既可以通过gpasswd命令管理用户组，也可以直接修改/etc/group文件中的内容来实现用户组的管理。

例如：

**向用户组添加成员（add）**

gpasswd -a 用户名 组名

**删除用户组中的某个成员（delete）**

gpasswd -d 用户名 组名

但是上述命令只能用root账户来执行，但是大多数时候，超级管理员有很多事情要做，所以我们可以通过用户组管理员来实现用户组的管理。

**添加用户组管理员**

**添加用户组管理员：**gpasswd -A 用户组管理员名 用户组名

该命令实际上就是在/etc/gshadow文件的第三栏中添加用户名，所以如果我们想要删除用户组管理员，我们只需要删除第三栏的数据即可。用户组管理员可像root那样管理用户组（增加和删除组员）。

### 主机的详细系统权限规划：ACL的使用（Access Control List）

如果有一个目录，需要给一堆人使用，而这一堆人的权限控制并不完全相同，但此时我们无法通过传统的Linux三种身份认证去完成权限控制的功能，此时我们就可以通过ACL来完成这个工作。

#### setfacl命令（P441）

命令：setfacl [-bkRd] [{-m|-x} ACL参数] 目标文件/目录

参数：

-m：设置后续的ACL参数给文件使用。

-x：删除后续的ACL参数，不可与-m合用

-b：删除所有的ACL参数

-R：递归设置ACL参数，包括所有子目录和子文件。

#### ACL参数

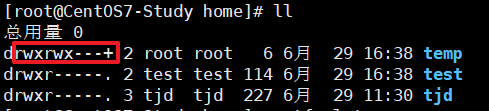
**为单一用户添加ACL参数**

格式：【u：用户名：权限】

例：setfacl -m u:tjd:rwx temp/ （该命令执行后，tjd用户对temp文件夹将有rwx权限）

**需要注意的是：如果用户名为空，代表为该文件拥有者设置ACL。**

**设置完ACL参数后的文件夹或文件在显示权限时会多出一个”+”号：**



**为用户组添加ACL参数**

格式：【g：用户组：权限】

例：setfacl -m g:student:rwx temp/ （该命令执行后，student用户组对temp文件夹将有rwx权限）

**针对有效权限设置**

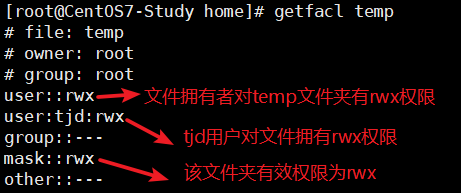
格式：【m：权限】

所谓有效权限是指：**用户或用户组所设置的权限必须要存在于mask的权限设置范围内才会生效**。

例：setfacl -m m:rx temp/ （表示为temp设置的ACL权限最多只有rx生效，例如上面tjd用户和student用户组虽然设置了rwx，但是只有rx生效）

#### getfacl命令

命令：getfacl 文件或文件夹



### sudo命令

sudo命令的作用是以其他账户身份去执行命令。

**命令：**sudo -b [-u 账户名]

**参数：**

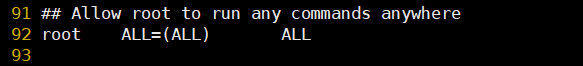
-b：将后序的命令放到后台中让系统自动执行，而不会影响当前正在执行的shell

-u：后面可以接欲切换的使用者，如果省略该选项则表示切换为root用户

实际上并不是所有人都能够执行sudo命令的，默认情况下只有root用户允许执行sudo命令。这是因为**用户每次执行sudo命令时，系统就会查看/etc/sudoers文件是否配置该用户的执行权限**。

我们可以在root用户下使用vim编辑器直接编辑/etc/sudoers文件，但是我们不推荐这么做，应为sudoers有自己的语法规范，如果语法有误则会导致系统故障（sudo命令无法使用），我们推荐使用**visudo命令**，它实质上也是去编辑/etc/sudoers文件，但它会在保存时校验格式是否正确。

#### 为单一用户添加sudo权限



上图是系统默认为root用户添加的sudo权限，该语句包含四个部分：

1. 【用户账号】：操作系统的那个账号可以使用sudo命令
2. 【登陆者来源的主机名】：也就是说这台账户可以是由哪台主机连接过来的（添加信任来源）
3. 【（可切换的身份）】：表示该用户在执行sudo命令时可切换的身份，ALL代表可切换为所有用户
4. 【可执行的命令】：表示该用户sudo权限可以执行哪些命令，注意这些命令一定要写绝对路径。

如果我们需要为tjd用户添加sudo权限，我们只需要将92行复制到93行，然后修改相应配置即可。

#### 为用户组添加sudo权限



%开头代表的是用户组的意思，CentOS7中默认为wheel组添加sudo权限，也就是说我们只需要将用户加入到wheel组中即可获得sudo权限。在Ubantu中，默认为sudo组添加sudo权限，同样的如果需要只需要加入这个组即可。

#### 免密执行sudo

在上述配置模式下，用户在执行sudo命令时都需要输入自己的密码才能执行（root用户除外），有时候我们会认为这样很不方便，所以我们可以配置免密执行sudo，同样的在/etc/sudoers中给出了模板案例：



如果我们需要配置只需要将102行的#号去掉即可，上面是为wheel组添加免密执行，如果我们想为个人配置，把前面的%去掉即可。

## 任务与进程管理

Linux操作系统是典型的多人多任务操作系统，而Windows是单人多任务操作系统。所谓多人多任务操作系统是指在同一时间可以若干用户登录，然后同时执行多个任务的操作系统。

### 任务管理

#### 什么是任务管理（job control）

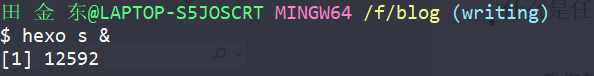
我们都说Linux是多任务操作系统，也就是说一个用户登陆后可以让多个任务同时运行，但是你面前只能控制一个bash界面，这就导致只有一个任务是放在前台（foreground）的，而大多数任务是放在后台（background）的。而我们把前台任务放入后台，或将后台任务调到前台来都是属于任务管理。

使用Bash进行任务管理的限制：

* 这些任务所触发的进程必须来自你Shell的 子进程（只能管理自己Bash产生的任务）
* 前台：可以控制与执行命令的这个环境称为前台的任务。
* 后台：可以自动执行的任务，你无法使用Ctrl+c来终止它，可使用bg，fg调用该任务。。
* 后台执行的任务不能等待用户的输入

#### 将命令放入后台执行

想要命令在后台执行，我们只需要在命令后面加入“&”符号即可



**输出格式：【job number】 PID**

方括号里面的是这个任务的序列号，是这个任务的标识符；后面的数字是执行进程的PID。

#### 将【目前】的任务丢到后台中【暂停】

当我们正在用vim编辑一个文件时，此时我们需要退出查询一个东西，那么此时我们不需要保存文件退出，只需要将当前任务丢到后台中暂停，然后查询完毕后再重新调出该任务即可。

**我们使用Ctrl+Z即可将【当前】任务放入后台【暂停】**



#### 将后台任务拿到前台来处理（foreground）

命令：fg %job\_number （%可有可无）

#### 将后台暂停的任务变成运行中

命令：bg %job\_number

#### 查看目前后台任务状态

命令：jobs [-lrs]

参数：

-l：除了列出job number和命令串之外，还列出PID。

-r：仅列出正在后台run的任务（后台任务状态有两种：stopped、running）。

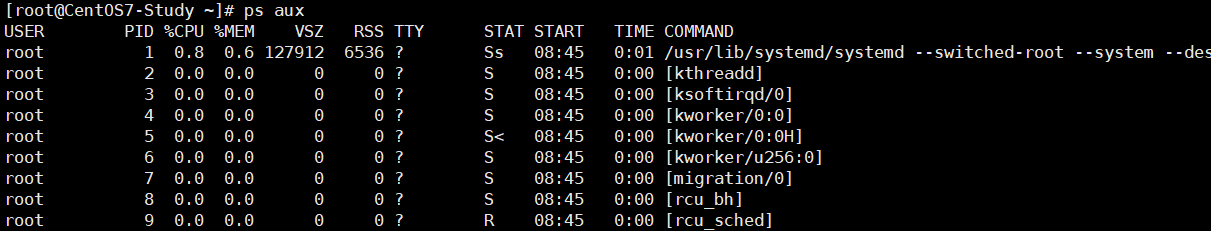
-s：仅列出正在后台暂停的任务

### 进程管理

#### 查看进程

**查看所有进程**

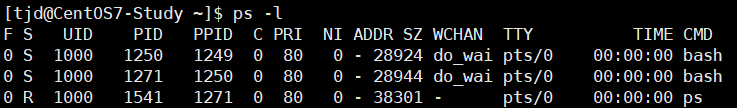
命令：ps aux



* USER：该进程所属的用户账号
* PID：该进程的ID
* %CPU：该进程使用的CPU占比
* %MEM：该进程所占用的物理内存占比
* VSZ：该进程所使用的虚拟内存量
* RSS：该进程占用的固定的内存量
* TTY：该进程在哪个终端上运行，若与终端无关，则显示问号(?)；tty1-tty6是本机上面的登录进程，若为pts/0等，则表示是由网络连接进入主机的进程。
* STAT：进程当前的状态（与ps -l命令中S字段相同）
* TIME：该进程实际使用CPU运行的时间
* COMMAND：触发该进程实际的命令是什么？

**查看自己bash相关进程**

命令：ps -l



* F：代表这个进程的标识（process flags），说明这个进程的权限。

常见的号码有：

* + 若为4则表示此进程的权限为root
  + 若为1则表示此子进程仅执行复制（fork）而没有实际执行（exec）
* S：代表进程执行的状态（Status），主要状态有：
  + R（Running）：该进程正在运行中
  + S（Sleep）：该进程正在休眠中，但可以被唤醒
  + D：代表该进行处于不可唤醒的休眠状态，通常这个进程在等待I/O。
  + T：停止状态，可能是在任务控制（暂停状态）或跟踪状态
  + Z（Zombie）：僵尸进程，进程已经终止但无法从内存中删除
* UID：此进程被UID这个用户所拥有
* PID：进程的PID号码
* PPID：该进程的父进程的PID（Parent PID）
* C：代表CPU使用率（单位百分比）
* PRI/NI：代表此进程被CPU所执行的优先级
* TTY：登录者的终端位置，若为远程登录则使用动态接口名称（pts/n）
* TIME：使用CPU的时间，注意是此进程实际花费CPU的运行时间。
* CMD：触发该进程的命令

#### 进程管理

进程之间是可以相互控制的，进程是如何相互管理的呢？其实是通过给与进程一个信号，去告知进程你想要它做什么。

**命令：**

**kill -signal PID （Signal是信号量，PID是进程的ID）**

**主要信号的代号、名称：**

|  |  |  |
| --- | --- | --- |
| 代号 | 名称 | 内容 |
| 1 | SIGHUP | 启动被终止的进程，可让该PID重新读取配置文件，类似于重新启动 |
| 2 | SIGINT | 相当于用键盘Ctrl+C来中断一个进程的运行 |
| 9 | SIGKILL | 代表强制中断一个进程的执行，如果该进程执行到一半，那么尚未完成的部分可能会有半成品出现，类似于vim就会有.filename.swp文件保留 |
| 15 | SIGTERM | 已正常的方式结束进程来终止该进程。由于是正常的终止，所以后续的操作会将它完成。不过如果该进程已经发生了问题，就是无法使用正常的方法终止时，输入这个信号也是没用的。 |
| 19 | SIGSTOP | 相当于键盘输入Ctrl+Z来暂停一个进程的运行 |

**我们还可通过kill -l命令查看所有信号量。**

#### 查看系统资源

##### 查看内存使用情况（free）

命令：free [-b|-k|-m|-g|-h] [-t] [-s N -c N]

参数：

b,k,m,g,h：指定显示内存大小的单位，b（Bytes），m（MB），g（GB），h表示让系统自己指定单位。

t：在输出的最终结果，显示物理内存和swap的总量。

s：让系统不断刷新显示数据。

c：与-s同时处理，，让free列出几次的意思。

##### 查看内核信息（uname）

命令：uname [-asrmpi]

选项参数：

-a：将所有系统相关的信息，包括下面的数据都会被列出来。

-s：系统内核名称。

-r：内核的版本

-m：本系统的硬件架构

-p：CPU类型

-i：硬件平台

## 软件安装

#### 传统方式编译C语言程序案例（P686）

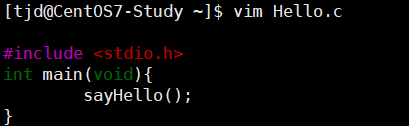
命令：

gcc xxx.c 编译xxx.c，直接生成二进制程序

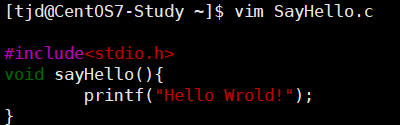
gcc -c xxx.c 编译xxx.c，生成目标文件（Object file）

gcc -o xxx.o 将目标文件编译为二进制程序

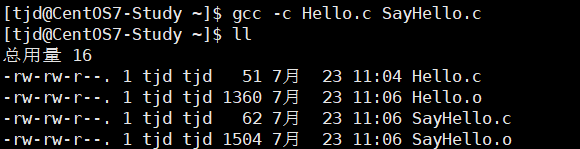
新建Hello.c文件



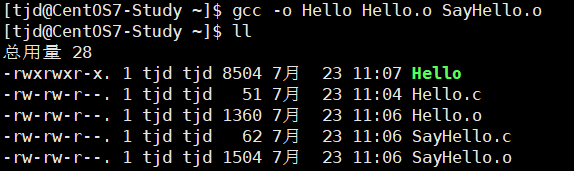
新建SayHello.c文件



将Hello.c与SayHello.c编译为目标文件



将链接目标文件生成二进制文件



#### 什么是make和configure

有时候c语言源文件非常多，那么使用gcc编译器手动编译会非常复杂**，此时我们可以使用make命令来简化编译过程。当执行make命令时，它会在当前目录下自动寻找makefile或Makefile文本文件，这里面记录了编译源码的详细信息，make命令会自动根据文件信息编译源码，而不需要我们自己进行这些繁琐的操作。**

那么makefile文件如何生成呢？通常软件开发商会编写一检测程序来检测用户的操作环境，该检测程序执行完毕后会自动生成makefile文件，通常这个检测程序名为configure或config。

**检测程序通常对下面几个方面进行检测：**

1. 是否有适合的编译器编译本软件程序代码。
2. **是否存在本软件所需要的软件库，或其他的依赖文件**
3. 操作系统平台是否适合本软件，包括Linux内核版本
4. 内核的头文件（Header include）是否存在（驱动程序必须要的检测）

由于不同版本的Linux发行版的函数库文件存放路径、函数库的文件名或是默认安装的编译器，以及内核版本都不相同，因此理论上，你无法将CentOS7.x中编译的源码放到SUSE中运行，这个操作通常是不会成功的。

#### 使用make进行宏编译

make命令的优势：

1. 简化编译时所需要执行的命令
2. 若在编译完成之后，修改了某些源文件，则make仅会针对被修改了的文件进行编译，不会重新编译其他文件。

#### Tarball软件的安装

所谓的tarball软件指的是使用tar命令将源文件打包压缩的源代码（\*.tar.gz、\*.tar.bz2、\*.tar.xz文件），

##### Tarball软件安装的大致步骤

第一步：获取原始文件

将tarball文件在/usr/local/src目录下解压缩（在默认情况下，原本Linux发行版发布安装的软件大多是在/usr里面，而用户自己安装的软件最好放入/usr/local下面，源码通常放在/usr/local/src目录下，这里是考虑到管理用户所安装的软件的便利性）；

注：usr的意思是：Unix Software Resource。8

第二步：获取步骤流程

进入解压目录下，去查看INSTALL或README等相关文件内容；

第三步：安装依赖文件

根据INSTALL/README的内容，安装该软件所需要的依赖软件。

第四步：建立makefile文件：

执行自动检测程序（configure或cnfig）达到检测操作环境目的，完成后会自动生成makefile文件。在执行configure程序时通常可以指定软件的安装路径（prefix）

第五步：编译

用make程序，并使用该目录下的makefile文件作为它的参数配置文件，来进行make（编译或其他）操作。

第五步：安装

以make这个程序，并以makefile这个参数配置文件，根据install这个目标的指定来安装到正确的路径。注意：编译完成的可执行文件与相关配置文件还在源代码所在的目录当中。因此，最后要进行make install来将编译完成的所有东西都安装到正确的目录中去，这样就可以使用该软件了。

#### 软件管理器

Linux界有两大主流的软件管理器：RPM（RedHat Package Manager）和DPKG（Debian Packager）。在前面我们介绍了tarball软件安装方式，这种软件方式实际上就是我们自己下载软件源码，然后自行编译源文件得到可执行程序。为什么软件开发商不给我们编译好呢？那是因为不同版本的Linux函数库版本和存储路径以及文件名并不统一，那么编译出来的可执行文件是无法放到不同Linux发行版中运行。

Linux发行版厂商将软件编译好了，放在远端，然后这个编译好的可执行的软件直接发布给用户来安装。如此一来，由于我们使用的是同一个厂商的发行版，所以操作系统中的环境都相同，所以此发行版的系统都能运行这个软件。

##### 什么是RPM和SRPM

RPM是打包好的数据，也就是说，里面的数据已经编译完成了，该软件安装文件几乎只能安装在原本默认的硬件与操作系统版本中。所以不同的Linux发行版本所发布的RPM文件，并不能用在其他的Linux发行版本上。举例来说，RedHat发布的RPM文件，通常无法在SUSE上面进行安装。更有甚者，相同Linux发行版的不同版本之间也无法互通，例如CentOS 6.x的RPM软件无法直接在CentOS 7.x上面运行。

RPM由于只能在各自的发行版本上运行，那么如果我们真的想要安装别的平台的RPM软件该怎么解决呢？SRPM为我们解决了这个困境。SRPM顾名思义，它是Source RPM的意思。也就是说这个RPM文件中保存的仍然是软件的源码（未经过编译），**通常SRPM的文件格式为\*.src.rpm**。为什么SRPM里面保存的是源代码，为什么我们不采用Tarball的形式安装呢？这是因为SRPM内容虽然是源代码，但是里面仍然含有该软件所需要的依赖性软件说明以及所有RPM文件所提供的数据。同时它与RPM不同的是，它也提供了参数配置文件（Configure或Config）。

所以如果我们下载的是SRPM，那么安装时，你就必须要：

* 先将软件一RPM管理的方式编译，此时SRPM会被编译成为RPM文件。
* 然后将编译完成的RPM文件安装到Linux系统当中。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 文件格式 | 文件名 | 直接安装 | 内含程序类型 | 可否修改参数编译 |
| RPM | \*.rpm | 可以 | 已编译 | 不可以 |
| SRPM | \*.src.rpm | 不可以 | 源码 | 可以 |

##### RPM软件包常见格式

例如：rp-pppoe-3.11-5.el7.x86\_64.rpm

rp-pppoe：软件名称

-3.11-：软件版本

5：发布的次数

.el.x86\_64：适用的硬件平台

#### RPM软件管理程序

##### RPM默认安装路径

/etc：一些配置文件放置的目录

/usr/bin：一些可执行文件

/usr/lob：一些程序使用的动态函数库

/usr/share/doc：一些基本的软件使用手册与说明文件

/usr/share/man：一些man page文件

##### RPM安装

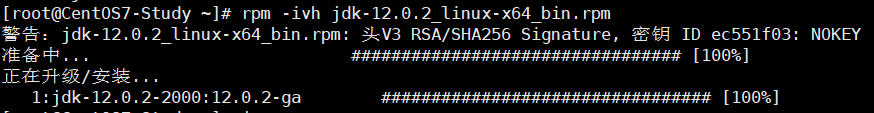
命令：rpm [-ivh] 文件名… （注：如果同时安装多个软件可以用空格隔开）

参数：

-i：install安装的意思。

-v：查看更详细的安装信息。

-h：显示安装进度。



##### RPM更新

使用RPM来更新软件是在太简单了，就以-Uvh或-Fvh选项来升级即可：

-Uvh：后面的软件即使没有安装过，则系统予以直接安装；若后面的软件安装过，那么直接升级

-Fvh：后面软件没有安装过则不会安装；只有在存在旧版本时升级

##### RPM查询

在使用RPM安装软件时，RPM包管理器会检查该软件所依赖的软件是否存在，若不存在则不会进行安装。若检查合格，才会进行安装。安装完成后，系统会将这个软件信息写入/var/lib/rpm/目录下的数据库文件中了。刚开始RPM在检查环境时寻找依赖软件环境就是从这个数据库中读取的。

RPM在查询时，其实查询的地方就是/var/lib/rpm这个目录下的文件。另外RPM也可以查询未安装的RPM文件内的信息。那该如何查询呢？

命令：

rpm -qa

rpm -a[licdR] 已安装软件名称

rpm -qf 存在于系统上面的某个文件名

rpm -qp[licdR] 未安装的某个文件名

参数：

-q：仅查询，后面接的软件名称是否有安装。

-qa：列出已安装在本机Linux系统上面的所有软件名称。

-qi：列出该软件的详细信息（information），包含开发商和版本说明

-ql：列出该软件所有的文件与目录所在完成文件名

-qc：例出该软件所有配置文件（找出/etc/下面的文件名而已）

-qd：列出该软件的所有说明文件（找出man有关的文件）

-qR：列出与该软件有关的依赖软件所含的文件（Required的意思）

-qf：由后面接的文件名，找出该文件属于哪个已安装的软件

-qp[licdR]：注意-qp后面接的所有参数以上面的说明一致，但用途仅在于找出某个RPM文件内的信息，而非已安装软件的信息。



##### RPM卸载

需要注意的是卸载安装的过程一定要由最上层往下解除。

**命令：**

rpm -e 软件名

##### RPM重建数据库

由于RPM软件经常会安装、删除、升级等，某些操作会导致rpm数据库文件损坏（/var/lib/rpm）。如果遇到这种情况，我们可以进行重建数据库的操作。

**命令：**

rpm –rebuilddb

#### YUM在线安装

由于Linux中的程序大多是小程序。程序与程序之间存在非常复杂的依赖关系。RPM无法解决软件包的依赖关系。

YUM被称为 Yellow dog Updater, Modified，是一个在Fedora和RedHat以及SUSE中的Shell前端软件包管理器。YUM使用Python语言写成。YUM客户端基于RPM包进行管理，可以通过HTTP服务器下载、FTP服务器下载、本地软件池的等方式获得软件包，可以从指定的服务器自动下载RPM包并且安装，可以自动处理依赖性关系。

##### YUM查询

命令：

yum [option] [查询工作选项] [相关参数]

选项参数：

[option] 主要参数：

-y：当yum要等待使用者输入时，这个选项可以自动提供yes的回应

--installroot=/some/path：将该软件安装在/some/path这个路径下，而不是默认路径。

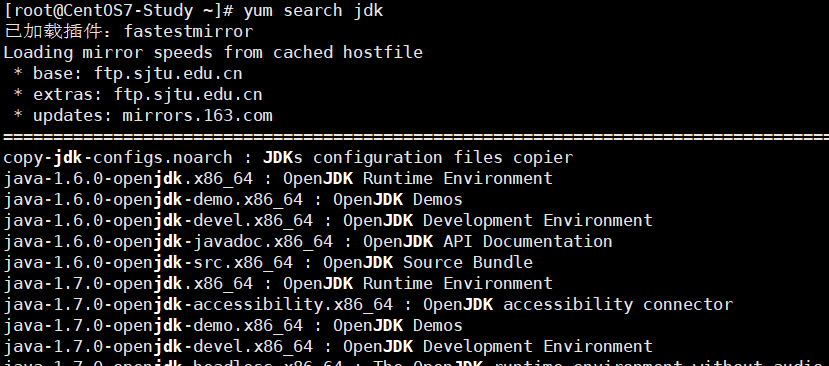
[查询工作选项][相关参数]主要有：

search：查找某个软件名称或是描述（description）的重要关键字

list：列出当前yum所管理的所有软件与版本，类似于rpm -qa;

info：同上，不过有点类似rpm -qai的执行结果

provides：从文件查找软件，类似于rpm -qa；



##### YUM安装与升级

命令：

yum [install | update]

参数与选项

install：后面接需要安装的软件

update：后面接要升级的软件，若要整个系统都升级，则直接update即可

##### yum卸载

命令：

yum remove 软件名

##### yum的配置文件

## CentOS服务器搭建

### JDK安装

**第一步：解压缩**

将JDK压缩包传输至Linux服务器，执行tar -zxvf jdk8-xxxx.tar.gz -C /usr/local/

**第二步：配置环境变量**

执行vi /etc/profile，按一下大写G进入文件末尾，在末尾插入：

JAVA\_HOME=/usr/local/jdk1.8.0\_171

PATH=${PATH}:${JAVA\_HOME}/bin

### Vim安装

执行：yum install -y vim