# Linux 分区与挂载

## 磁盘分区

**磁盘分区是使用分区编辑器(partition editor)在磁盘上划分几个逻辑部分。碟片一旦划分成数个分区(partition)，不同类的目录与文件可以存储进不同的分区。**

**思考：为什么要分区？**

## 分区类型

**主分区：最多只能有4个。**

**拓展分区：**

**最多只能有一个。**

**主分区加拓展分区最多有4个**

**不能写入数据，只能包含逻辑分区**

**逻辑分区**

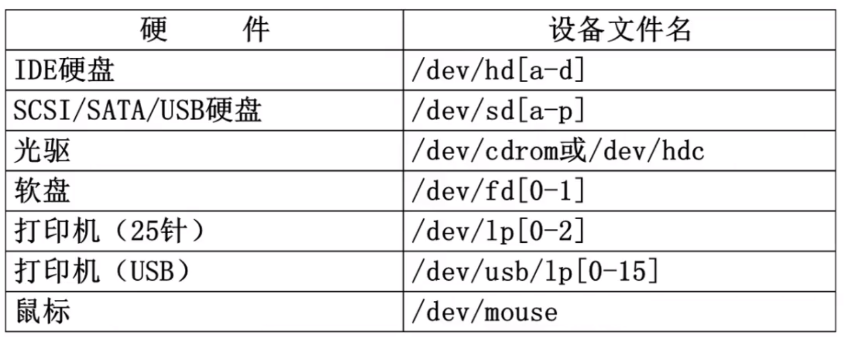
**为什么主分区只能有4个？**

**磁盘首个512字节存放的是主引导记录（MBR，Main Boot Record），为446字节的 BootLoader（操作系统内核运行之前(通电后)运行的一段小程序） + 64 字节的分区表 + 2 字节分区标志。分区表中每个分区记录占16字节，所以最大的为主分区 + 拓扩展分区数为 64/16 = 4 。如果需要超过四个分区，则 MBR 中的一个分区记录必须为扩展分区，多出的分区表信息存放在磁盘其他地方，扩展分区就是用来指示这些多出的分区信息到底存放在哪里。**

## 格式化

**格式化（高级格式化）又称逻辑格式化，它是指根据用户选定的文件系统（如 FAT16 、 FAT32 、 NTFS 、 EXTZ 、 EXT3 、 EXT4 、XFS等），在磁盘的特定区域写入特定数据，在分区中划出一片用于存放文件分配表、目录表等用于文件管理的磁盘空间。**

## 硬件设备文件名

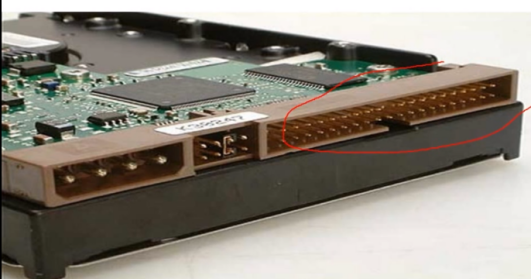


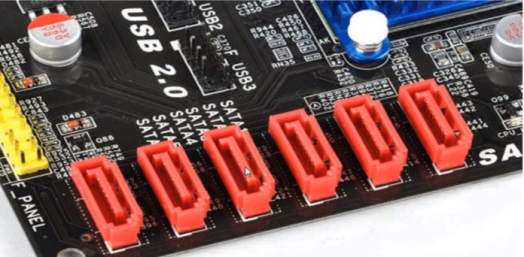
**设备文件名:**

**/dev/hda1(IDE硬盘接口)**

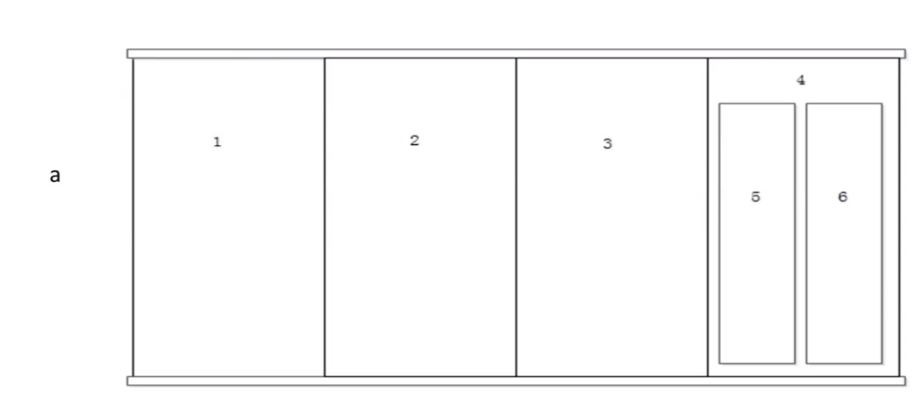
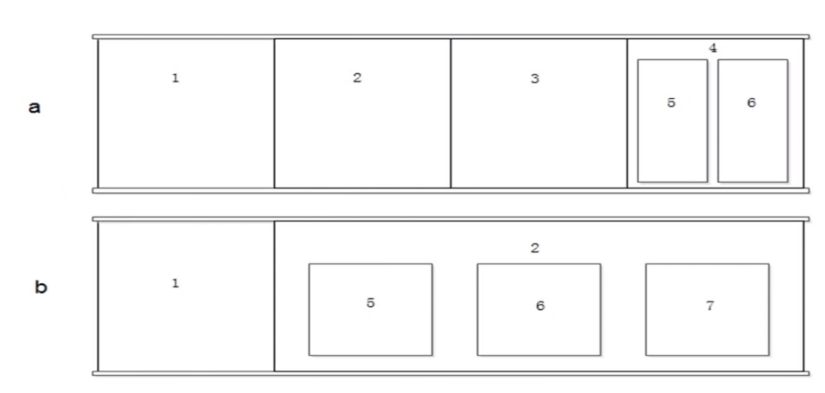
**/dev/sda1(SCSI硬盘接口、SATA硬盘接口)**

ide

scsi

sata

## 分区表示

## 挂载

**必须分区**

**/ (根分区)**

**swap分区(交换分区，内存2倍，不超过2GB)**

**swap分区大小建议：内存小于4G，建议2倍， 内存大于4G，和内存同样大小即可**

**推荐分区**

**/boot (启动分区，200MB)**

## 总结

**分区：把大硬盘分为小的逻辑分区**

**格式化：清空数据并写入文件系统．**

**分区设备文件名：给每个分区定义设备文件名．**

**挂载：给每个分区分配挂载点**

# 文件目录命令

## 命令提示符

**[root@localhost ~]#**

**其中:**

**root 当前登录用户**

**localhost 主机名**

**~ 当前所在目录(家目录)**

**# 超级用户的提示符**

**普通用户的提示符是$**

## 命令格式

**命令 [选项] [参数]**

**如 ls -a /**

**注意:个别命令使用不遵循此格式**

**当有多个选项时,可以写在一起**

**简化选项与完整选项**

**-a等于--all**

## 查询目录中内容:ls

**ls [选项] [文件或目录]**

**选项**

**-a显示所有文件,包括隐藏文件**

**-I 显示详细信息**

**-d查看目录属性**

**-h人性化显示文件大小**

**-i 显示 inode**

**文件权限:**

**-rw-r--r--**

**- 文件类型(-文件 d目录 l软链接文件)**

**rw- r-- r--**

**u所有者 g所属组 o其他人**

**r读 w写 x执行**

## 建立目录:mkdir

**mkdir -p [目录名]**

**-p递归创建**

**命令英文原意: make directories**

## 切换所在目录:cd

**cd [目录]**

**命令英文原意: change directory**

**简化操作**

**cd ~ 进入当前用户的家目录**

**cd 同上**

**cd - 进入上次目录**

**cd .. 进入上一级目录**

**cd . 进入当前目录(没有实际意义)**

**相对路径:参照当前所在目录,进行查找**

**如: [root@ localhost ~]# cd ../usr/local/src/**

**绝对路径:从根目录开始指定,一级一级递归查找。在任何目**

**录下,都能进入指定位置**

**如: [root@ localhost ~]# cd /etc**

## 查询所在目录位置:pwd

**pwd**

**命令英文原意: print working directory**

## 删除空目录: rmdir

**rmdir [目录名]**

**命令英文原意: remove empty directories**

## 删除文件或目录:rm

**rm -rf [文件或目录]**

**命令英文原意: remove**

**选项**

**-r删除目录**

**-f强制**

## 复制命令:cp

**cp [选项] [原文件或目录] [目标目录]**

**命令英文原意:copy**

**选项:**

**-r复制目录**

**-p连带文件属性复制**

**-d若源文件是链接文件,则复制链接属性**

**-a相当于-pdr**

## 剪切或改名命令:mv

**mv [原文件或目录] [目标目录]**

**命令英文原意:move**

## 常用目录的作用

**/ 根目录**

**/bin命令保存目录(普通用户就可以读取的命令)**

**/boot启动目录,启动相关文件**

**/dev设备文件保存目录**

**/etc配置文件保存目录**

**/home普通用户的家目录**

**/root 管理员的家目录**

**/lib系统库保存目录**

**/mnt系统挂载目录**

**/media挂载目录**

**说明：**

**根目录下的bin和sbin,usr目录下的bin和sbin,**

**这四个目录都是用来保存系统命令的。**

**proc和sys目录不能直接操作,这两个目录**

**保存的是内存的过载点。**

**可以在家目录root或home,以及tmp目录**

**下随便放内容**

## 链接命令

**In -s [原文件] [目标文件]**

**命令英文原意:link**

**功能描述:生成链接文件**

**选项:-s创建软链接**

**硬链接特征:**

**1、拥有相同的节点和存储 block块,可以看做是同一个文件**

**2、可通过i点识别**

**3、不能跨分区**

**4、不能针对目录使用**

**软链接特征**

**1、类似 Windows快捷方式**

**2、软链接拥有自己的I节点和Bock块,但是数据块中只保存原**

**文件的文件名和点号,并没有实际的文件数据**

**3、Irwxrwxrwx I代表软链接**

**软链接文件权限都为 rwxrwxrwx**

**4、修改任意文件,另一个都改变**

**5、删除原文件,软链接不能使用**

# 搜索命令

## 文件搜索命令 locate

**命令格式**

**locate文件名**

**在后台数据库中按文件名搜索,搜索速度更快**

**/var/lib/mlocate**

**# locate命令所搜索的后台数据库**

**updatedb**

**更新数据库命令**

**/etc/updatedb.conf配置文件**

**PRUNE\_BIND\_ MOUNTS = "yes"**

**#开启搜索限制**

**PRUNEFS**

**#搜索时,不搜索的文件系统**

**PRUNENAMES**

**#搜索时,不搜索的文件类型**

**PRUNEPATHS**

**#搜索时,不搜索的路径**

## 命令搜索命令 whereis

**whereis命令名**

**#搜索命令所在路径及帮助文档所在位置**

**选项**

**-b:只查找可执行文件**

**-m:只查找帮助文件**

## 命令搜索命令 which

**which文件名**

**#搜索命令所在路径及别名**

## PATH环境变量

**PATH环境变量:定义的是系统搜索命令的路径**

**[root@localhost ~]# echo $PATH**

**/usr/ib/qt-3.3/bin:**

**/usr/local/ sbin: /usr/local/bin: / sbin: /bin: /usr/sbin: /usr/bi**

**n root/bin**

## 文件搜索命令find命令

**find命令**

**find [搜索范围] [搜索条件]**

**#搜索文件**

**find / -name install log**

**#避免大范围搜索,会非常耗费系统资源**

**#find是在系统当中搜索符合条件的文件名。如果需要匹配,**

**使用通配符匹配,通配符是完全匹配**

**Linux中的通配符**

**\* 匹配任意内容**

**? 匹配任意一个字符**

**[] 匹配任意一个中括号内的字符**

**其他选项**

**find /root -iname install,log**

**#不区分大小写**

**find root -user root**

**#按照所有者搜索**

**find /root -nouser**

**#查找没有所有者的文件**

**根据时间查找**

**find /var/log/ -mtime +10**

**#查找10天前修改的文件**

**-10 10天内修改文件**

**10 10天当天修改的文件**

**+10 10天前修改的文件**

**atme文件访问时间**

**ctime改变文件属性**

**mtime修改文件内容**

**根据大小查找**

**find . -size 25k**

**#查找文件大小是25KB的文件**

**-25k 小于25KB的文件**

**25k 等于25KB的文件**

**+25k大于25KB的文件**

**根据i节点查找**

**find . -inum 262422**

**#查找i节点是262422的文件**

**综合应用**

**find /etc -size +20k -a -size -50k**

**#查找/etc/目录下,大于20KB并且小于50KB的文件**

**-a and逻辑与,两个条件都满足**

**-o or逻辑或,两个条件满足一个即可**

**find /etc/ -size +20k -a -size -50k -exec ls -lh {} \;**

**#查找/etc/目录下,大于20KB并且小于50KB的文件,并显示详细信息**

**# -exec/-ok命令{} \; 对搜索结果执行操作**

## 搜索字符串命令grep

**grep [选项] 字符串 文件名**

**#在文件当中匹配符合条件的字符串**

**选项**

**-i 忽略大小写**

**-v 排除指定字符串**

**如: grep “size” anaconda-ks.cfg**

**find命令与grep命令的区别**

**find命令:在系统当中搜索符合条件的文件名,如果需要匹配**

**使用通配符匹配,通配符是完全匹配。**

**grep命令:在文件当中搜索符合条件的字符串,如果需要匹配**

**,使用正则表达式进行匹配,正则表达式时包含匹配**

# 帮助命令

## 帮助命令man

**man 命令**

**#获取指定命令的帮助**

**man ls**

**#查看ls的帮助**

**man的级别**

**1 查看命令的帮助**

**2 查看可被内核调用的函数的帮助**

**3 查看函数和函数库的帮助**

**4 查看特殊文件的帮助(主要是/dev目录下的文件)**

**5 查看配置文件的帮助**

**6 查看游戏的帮助**

**7 查看其它杂项的帮助**

**8 查看系统管理员可用命令的帮助**

**9 查看和内核相关文件的帮助**

**查看命令拥有那个级别的帮助**

**man -f 命令**

**相当于**

**whatis命令**

**举例:**

**man 5 passwd**

**man 4 null**

**man 8 ifconfig**

**查看和命令相关的所有帮助**

**man -k命令**

**相当于**

**apropos命令**

**例如**

**apropos passwd**

## 选项帮助 –help

**命令 --help**

**#获取命令选项的帮助**

**例如**

**Is –help**

## shell内部命令帮助

**help shell内部命令**

**#获取shell内部命令的帮助**

**例如**

**type cd**

**#确定是否是shell内部命令**

**此处可以比较 type -a ls 和 type cd的不同去感受**

**help cd**

**#获取内部命令帮助**

## 详细命令帮助info

**info命令**

**回车:进入子帮助页面(带有\*号标记)**

**u: 进入上层页面**

**n: 进入下一个帮助小节**

**p: 进入上一个帮助小节**

**q: 退出**

**例如：**

**Info ls**

# 压缩与解压缩命令

**常用压缩格式: .zip .gz .bz2**

**常用压缩格式: .tar.gz .tar.bz2**

## zip格式压缩

**zip压缩文件名 源文件**

**#压缩文件**

**zip -r压缩文件名 源目录**

**#压缩目录**

## zip格式解压缩

**unzip压缩文件**

**#解压缩zip文件**

## gz格式压缩

**gzip源文件**

**#压缩为gz格式的压缩文件,源文件会消失**

**gzip -c源文件 > 压缩文件**

**#压缩为gz格式,源文件保留**

**例如:gzip- c cangls> cangls.gz**

**gzip -r 目录**

**#压缩目录下所有的子文件,但是不能压缩目录**

## gz格式解压缩

**gzip -d 压缩文件**

**#解压缩文件**

**gunzip 压缩文件**

**# 解压缩文件**

**gunzip -r 压缩目录**

**# 解压缩目录**

## bz2格式压缩

**bzip2源文件**

**#压缩为.bz2格式,不保留源文件**

**bzip2 -k源文件**

**#压缩之后保留源文件**

**注意:bzip2命令不能压缩目录**

## bz2格式解压缩

**bzip2 -d 压缩文件**

**#解压缩,-k保留压缩文件**

**bunzip2 压缩文件**

**#解压缩,-k保留压缩文件**

## 打包命令tar

**tar -cvf 打包文件名 源文件**

**选项:**

**-c:打包**

**-v:显示过程**

**-f:指定打包后的文件名**

**例如**

**tar -cvf longzls.tar longzls**

## 解打包命令

**tar -xvf 打包文件名**

**选项:**

**-x:解打包**

**例如:**

**tar -xvf longzls.tar**

## tar.gz压缩格式

**其实 tar . gz格式是先打包为tar格式,再压缩为gz格式**

**tar -zcvf 压缩包名.tar.gz 源文件**

**选项:**

**-c: 压缩**

**-z: 压缩为tar.gz格式**

**tar -zxvf 压缩包名.tar.gz**

**选项:**

**-x: 解压缩.tar.gz格式**

## tar.bz2压缩格式

**tar -jcvf 压缩包名.tar.bz2 源文件**

**选项**

**-j:压缩为tar.bz2格式**

**tar -jxvf压缩包名.tar bz2**

**选项:**

**-x:解压缩.tar.bz2格式**

# 关机和重启命令

## shutdown关机命令

**root@ Localhost~]# shutdown [选项] 时间**

**选项:**

**-c:取消前一个关机命令**

**-h:关机（不要用这个选项）**

**-r:重启**

**例如：**

**[root@localhost ~]# shutdown -r 05:30**

**# 指定05:30重启**

**~~root@localhost ~]# shutdown -r 05: 30&~~**

**~~# &是后台运行不占用当前命令行的意思~~**

**root@localhost ~]# shutdown -h now**

**# 现在立即关机**

## 其他关机命令

**[root@localhost ~]# halt**

**[root@localhost ~]# poweroff**

**[root@localhost ~]# init 0**

**不安全,因为在关机的时候不会像 shutdown那样正确保存服务**

## 其他重启命令

**[root@localhost -] reboot**

**[root@localhost - ] init 6 #建议不要用**

## 系统运行级别

|  |  |
| --- | --- |
| **运行级别** | **含义** |
| **0** | 关机 |
| **1** | 单用户模式,可以想象为 windows的安全模式,主要用于系统修复 |
| **2** | 不完全的命令行模式,不含NFS服务 |
| **3** | 完全的命令行模式,就是标准字符界面 |
| **4** | 系统保留 |
| **5** | 图形模式 |
| **6** | 重启 |

**[root@localhost -]# cat /etc/inittab**

**#里面记录了修改系统默认运行级别的方法**

**~~id: 3: initdefault.~~**

**[root@localhost -] runlevel**

**#查询系统运行级别**

**[root@ localhost~]#init运行级别**

**#修改运行级别命令**

## 退出登录命令

**[root@localhost ~] # logout**

**# 同学们在操作完之后还是要养成logout退出登录的习惯**

**优秀工程师的修养**

# 挂载命令

## 查询与自动挂载

**[root@localhost ] mount**

**#查询系统中已经挂载的设备**

**[root@localhost ] mount -a**

**#依据配置文件 /etc/fstab的容,自动挂载**

## 挂载命令格式

**[root@ localhost~]# mount [-t文件系统] [-o特殊选项] 设备文件名 挂载点**

**选项:**

**-t文件系统:加入文件系统类型来指定挂载的类型,可以ext3、ext4、iso9660等文件系统**

**-o特殊选项:可以指定挂载的额外选项**

**不要求记住下表，通常都是默认就ok了**

|  |  |
| --- | --- |
| **参数** | **说明** |
| **atime/noatime** | **更新访问时间/不更新访问时间。访问分区文件时,是否更新文件的访问时间,默认为更新** |
| **async/sync** | **异步/同步,默认为异步** |
| **auto/noauto** | **自动/手动, mount-a命令执行时,是否会自动安装/et/stab文件内容挂载,默认为自动** |
| **defaults** | **定义默认值,相当于 rw, suid, dev, exec,auto; nouser, async这七个选项** |
| **exec/noexec** | **执行/不执行,设定是否允许在文件系统中执行可执行文件,默认是exec允许** |
| **remount** | **重新挂载已经挂载的文件系统,一般用于指定修改特殊权限** |
| **rw/ro** | **读写/只读,文件系统挂载时,是否具有读写权限,默认是rw** |
| **suid/nosuid** | **具有/不具有SUID权限,设定文件系统是否具有SUID和SGID的权限,默认是具有** |
| **user/houser** | **允许/不允许普通用户挂载,设定文件系统是否允许普通用户挂载,默认是不允许,只有root可以挂载分区** |
| **usrquota** | **写入代表文件系统支持用户磁盘配额,默认不支持** |
| **grpquota** | **写入代表文件系统支持组磁盘配额,默认不支持** |

**例如:**

**[root@localhost - ] mount-o remount, noexec /home**

**重新挂载/home分区,并使用 noexec权限**

**[root@localhost sh]# cd/home**

**Troot@localhost boot]# vim hello. sh**

**#!/bin/bash**

**echo “hello world”**

**[root@localhost boot]# chmod 755 hello. sh**

**[root@localhost boot]# /hello. sh**

**[root@localhost boot]# mount-o remount, exec /home**

**记得改回来啊,要不会影响系统启动的**

## 挂载光盘

**[root@localhost -1# mkdir /mnt/cdrom/**

**# 建立挂载点**

**[root@localhost - ] mount -t iso9660 /dev/cdrom /mnt/cdrom/**

**# 挂载光盘**

**[root@localhost ] mount /dev/sr0 /mnt/cdrom/**

**# 光盘默认是iso9660的文件系统，所以可以去掉该选项**

## 卸载命令

**root@ localhost~]# umount 设备文件名或挂载点**

**[root@localhost ~]# umount /mnt/cdrom**

## 挂载U盘

**[root@localhost ~]# fdisk -l**

**#查看U盘设备文件名**

**[root@localhost ~]# mount -t vfat /dev/sdb1 /mnt/usb/**

**注意: Linux默认是不支持NTFS文件系统的,不过现在出现了一个叫ntfs-3g的家伙**

# 用户登录查看命令

## 查看登录用户信息

**w用户名**

**命令输出:**

**USER:登陆的用户名**

**TTY:登陆终端;**

**~~FROM:从哪个IP地址登陆~~**

**LOGIN@:登陆时间**

**IDLE:用户闲置时间**

**JCPU:指的是和该终端连接的所有进程占用的时间。这个时间里并不包括过**

**去的后台作业时间,但却包括当前正在运行的后台作业所占用的时间;**

**PCPU:是指当前进程所占用的时间**

**WHAT:当前正在运行的命令**

**who用户名**

**命令输出**

**用户名**

**登录终端**

**登录时间(登录来源IP地址)**

## 查询当前登录和过去登录的用户信息

**last**

**last命令默认是读取/var/log/wtmp文件数据**

**命令输出**

**用户名**

**登录终端**

**登录IP**

**登录时间**

**退出时间(在线时间)**

## 查看所有用户的最后一次登录时间

**lastlog**

**lastlog命令默认是读取/var/log/lastlog文件内容**

**命令输出**

**用户名**

**登录终端**

**登录IP**

**最后一次登录时间**

# vim编辑器

## 安装

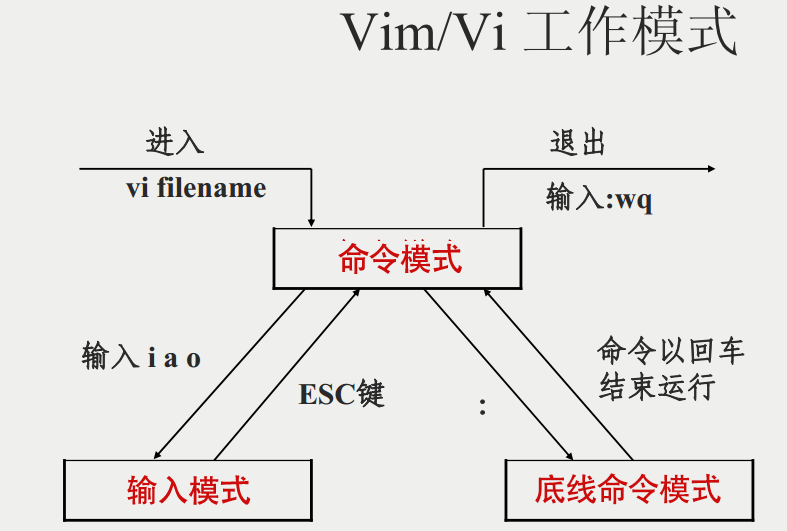
yum -y install vim

## 教程和学习方法

在命令行输入： vimtutor 查看30分钟速度入门教程。

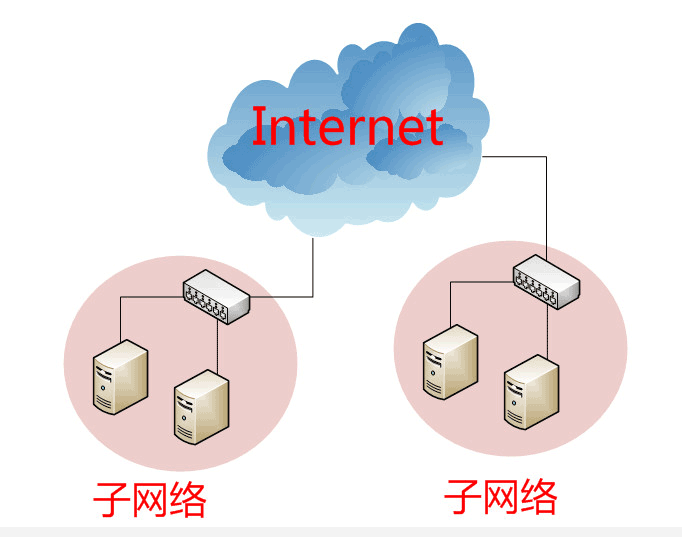
**务必按该教程操作2遍以上！**

## 三种工作模式



# 网络基础知识

**网络基础知识和计算机的基础知识一样重要！！！**



参考链接:

<http://www.ruanyifeng.com/blog/2012/05/internet_protocol_suite_part_i.html>

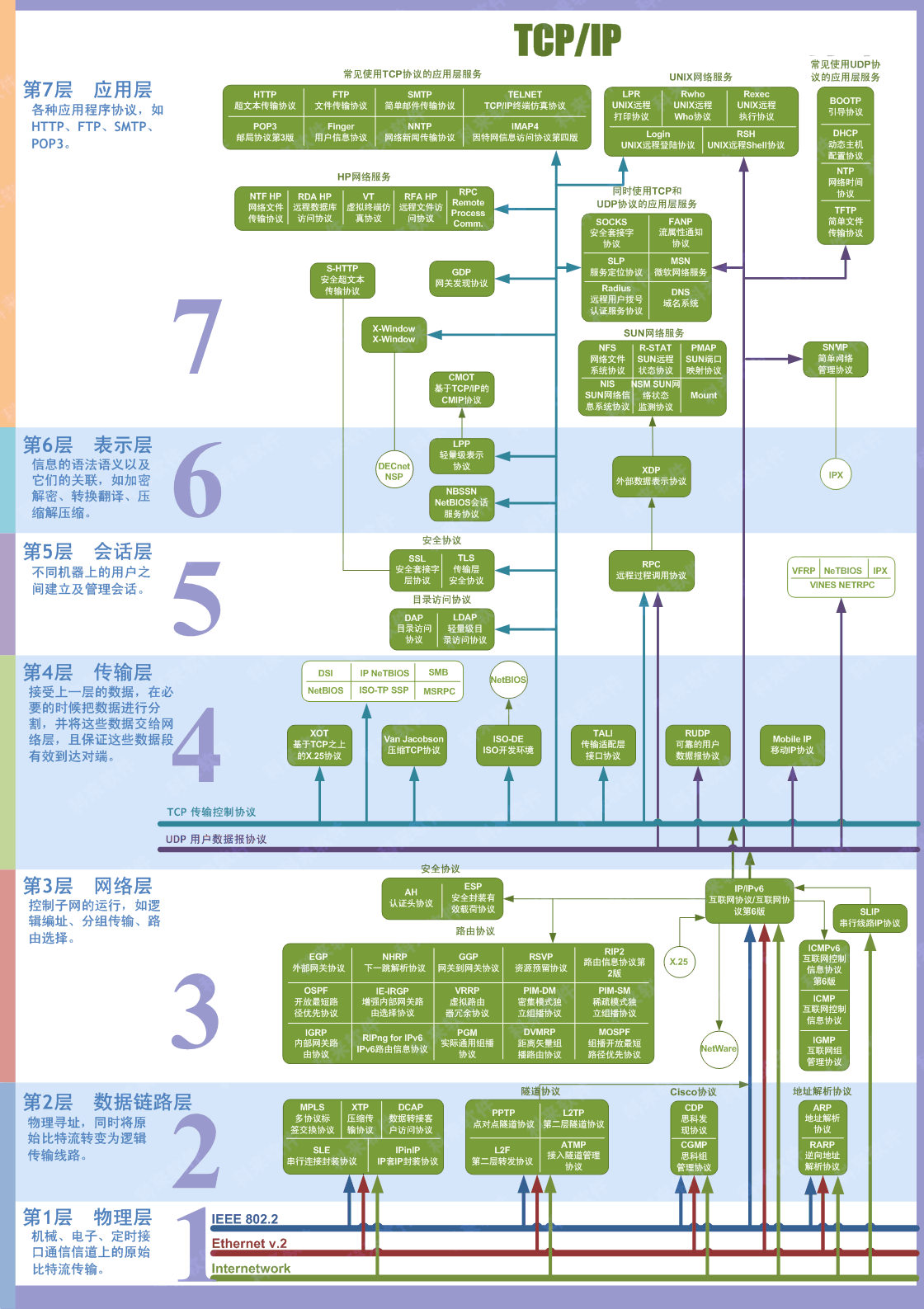
<http://www.ruanyifeng.com/blog/2012/06/internet_protocol_suite_part_ii.html>

注意:

下图是网络知识大全图，大而全，建议别看，会有头痛，眼晕的症状

知识的海洋太大，穷其一生我们也不能做到样样精通，所以需要有所选择。

选择一个对的方向

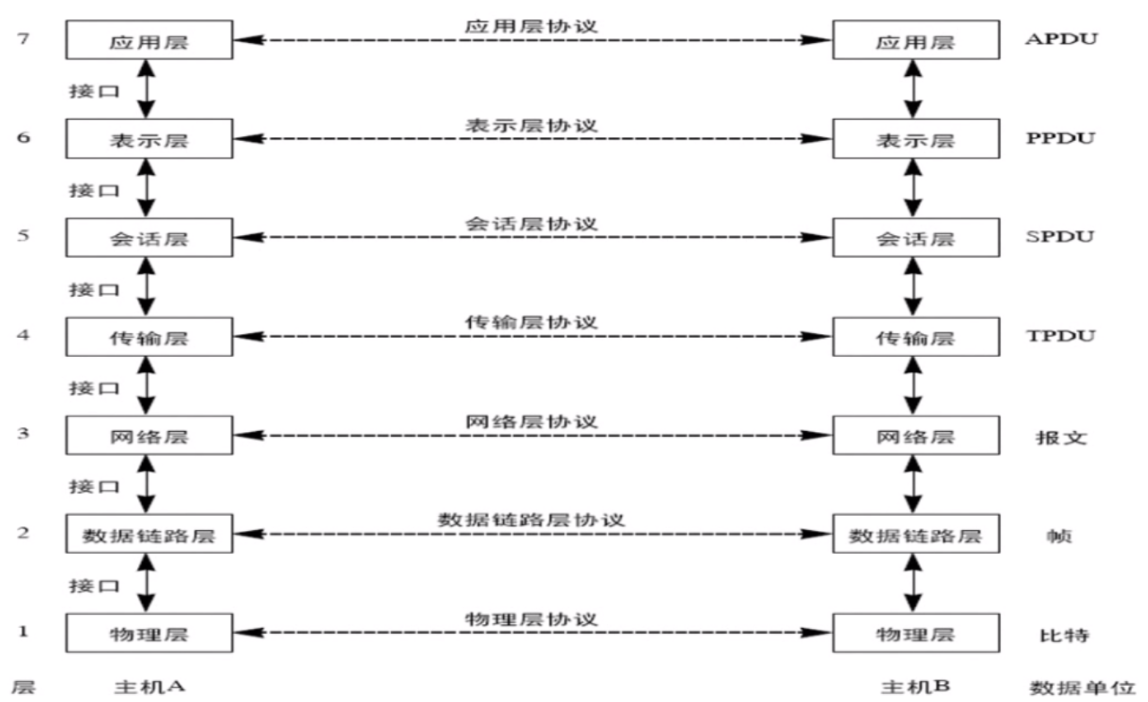
## ISO/OSI七层模型

**ISO:国际标准化组织**

**OSI:开放系统互联模型**

**IOS:苹果手机操作系统**

### OSI的七层模型

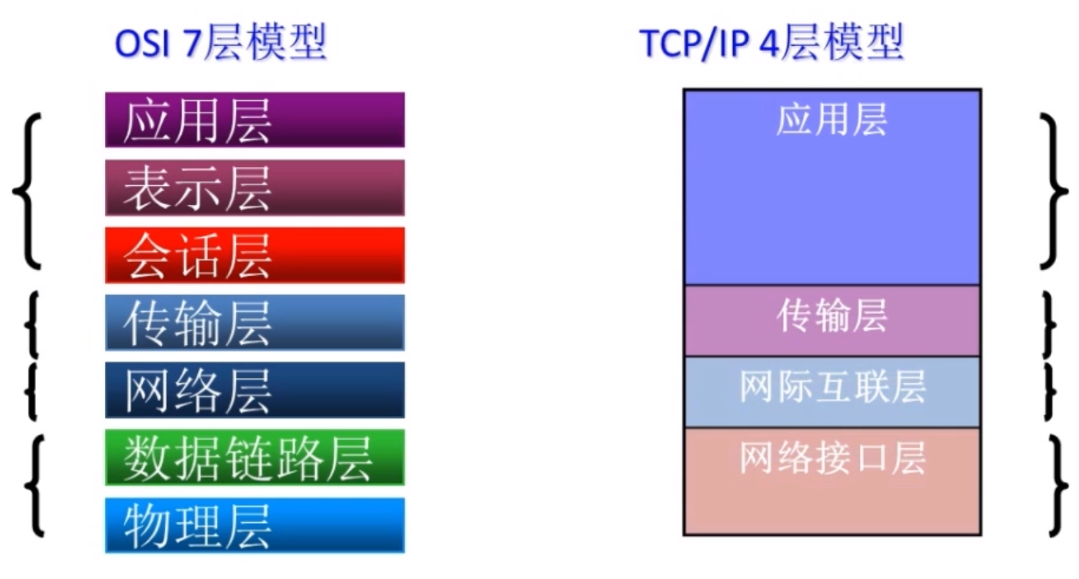


### OSI七层模型详解

|  |  |
| --- | --- |
| **名称** | **作用** |
| **应用层** | 用户接口 |
| **表示层** | 数据的表现形式,特定功能的实现,如加密 |
| **会话层** | 对应用会话的管理、同步 |
| **传输层** | 可靠与不可靠的传输、传输前的错误检测、流控 |
| **网络层** | 提供逻辑地址、选路 |
| **数据链路层** | 成帧、用MAC地址访问媒介、错误检测与修正 |
| **物理层** | 设备之间的比特流的传输、物理接口、电气特性等 |

## TCP/IP四层模型

### TCP/IP模型与OSI模型的对应关系



### 网络接口层

**网络接入层与OSI参考模型中的物理层和数据链路层相对应。它负责监视数据在主机和网络之间的交换。事实上,TCP/IP本身并未定义该层的协议,而由参与互连的各网络使用自己的物理层和数据链路层协议,然后与TCP/IP的网络接入层进行连接。地址解析协议(ARP)工作在此层,即OSI参考模型的数据链路层**。

**在windows中尝试输入: arp -a**

### 网际互联层

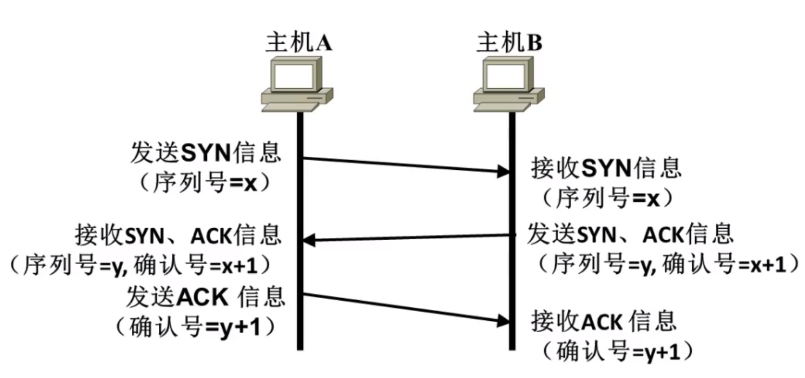
**网际互联层对应于OSI参考模型的网络层,主要解决主机到主机的通信问题。它所包含的协议设计数据包在整个网络上的逻辑传输。该层有三个主要协议:网际协议(IP)、互联网组管理协议(IGMP)和互联网控制报文协议(ICMP)。**

### 传输层

**传输层对应于osi参考模型的传输层,为应用层实体提供端到端的通信功能,保证了数据包的顺序传送及数据的完整性。该层定义了两个主要的协议:传输控制协议(TCP)和用户数据**

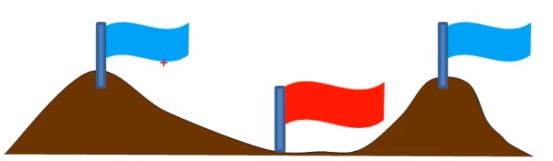
**报协议(UDP)**

### TCP/IP三次握手

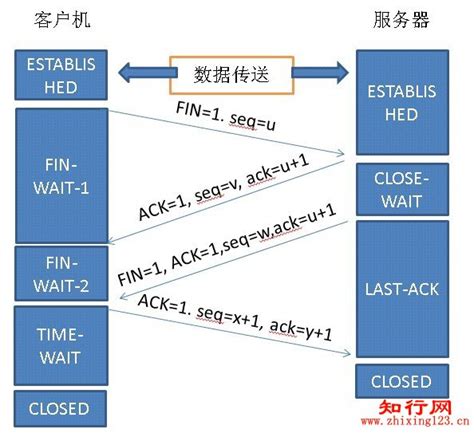


**为什么要三次握手？什么是两军问题？**

**一支红色部队在山谷里扎营,在周围的两边山坡上驻扎着蓝色部队红色部队比两支蓝色部队中的任意一支都要强大,但两支蓝色部队加在一起就要比红色部队强大如果一支蓝色部队单独作战,那么它就会被红色部队击败;如果两支蓝色部队同时进攻,他们将能够把红色部队击败两支蓝色部队需要同步他们的进攻,但他们惟一的通信媒介是步行进入山谷,在那里他们可能被俘虏,从而将信息丟失(换言之,他们必须使用非可靠的通信信道).问题是,是否存在一个协议,能够使得蓝色部队取胜?这就是有名的两军问题**



### TCP/IP四次挥手



**第一次挥手**

**将设客户端首先发起断开连接，那么客户端回想服务端发送FIN置为1的TCP包，请求断开连接，意思就是我要断开和你的连接了，但是如果你还有数据没有发送完给我你不必立即关闭连接。**

**第二次挥手**

**服务端收到客户端的断开连接请求立即响应一个ACK报文，意思是告诉客户端你发起的断开连接请求我已经收到了，但是我还没有准备好，你在等一会，这个时候服务器端可能还有数据要发送给客户端，也可能正在准备释放资源。这个时候客户端进入FIN\_WAIT状态，继续等待服务端的FIN报文。**

**第三次挥手**

**服务端确认已经发往客户端的数据已经发送完成，则向客户端发送FIN报文，告诉客户端我已准备好关闭连接了。**

**第四次挥手**

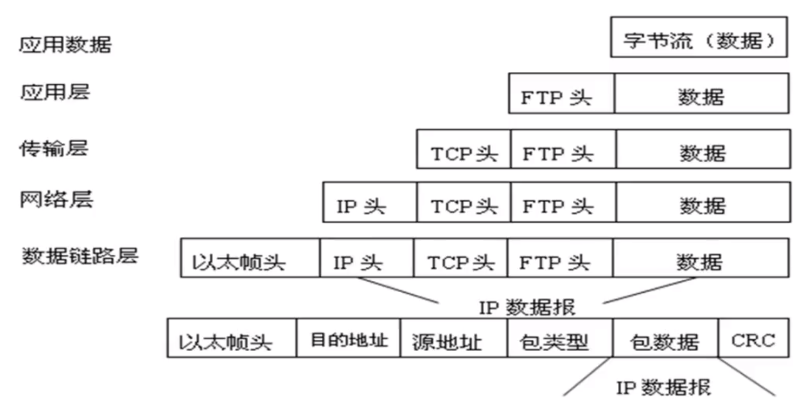
**客户端收到服务端的FIN报文后就知道可以关闭连接了，当时它还是不确定，怕服务端还是不知道我要关闭连接了，所以发送一个ACK包后进入TIME\_WAIT状态，如果服务端没有收到ACK那么服务端则可以发起重传，如果收到了ACK报文，客户端等待2MSL后已然没有收到回复，则证明服务端已经正常关闭了，那我也就可以关闭连接了。**

### 应用层

**应用层对应于OSI参考模型的高层,为用户提供所需要的各种服务**

**例如:FTP、Telnet、DNS、SMTP等.**

### 数据封装过程



### TCP/IP模型与OSI模型的比较

**共同点**

**(1)OSI参考模型和 TCP/IP参考模型都采用了层次结构的概念,**

**(2)都能够提供面向连接和无连接两种通信服务机制**

**不同点**

**(1)前者是七层模型,后者是四层结构**

**(2)对可靠性要求不同(后者更高)**

**(3)OSI模型是在协议开发前设计的,具有通用性TCP是先有协议集然后建立模型,不适用于非TCP/IP网络**

**(4)实际市场应用不同(OSI模型只是理论上的模型,并没有成熟的产品,而**

**TCP/IP已经成为“实际上的国际标准”)**

## IP地址

### ip包头



### IP地址

**00000000.00000000.00000000.00000000**

**11111111.11111111.11111111.11111111**

**0.0.0.0**

**255.255.255.255**

### ip地址分类

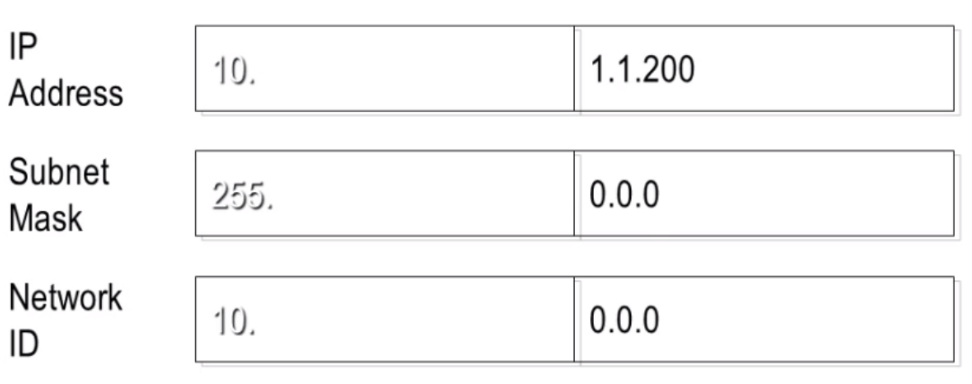
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **网络类别** | **最大网络数** | **IP地址范围** | **最大主机数** | **私有IP地址范围** |
| **A** | **126(2^7-2)** | **1.0.0.0--126.255.255.255** | **2^24-2** | **10.0.0.0--10.255.255.255** |
| **B** | **16384(2^14)** | **128.0.0.0--191.255.255.255** | **2^16-2** | **172.16.0.0--172.31.255.255** |
| **C** | **2097152(2^21)** | **192.0.0.0-223.255.255.255** | **2^8-2** | **192.168.0.0--192.168.255.255** |

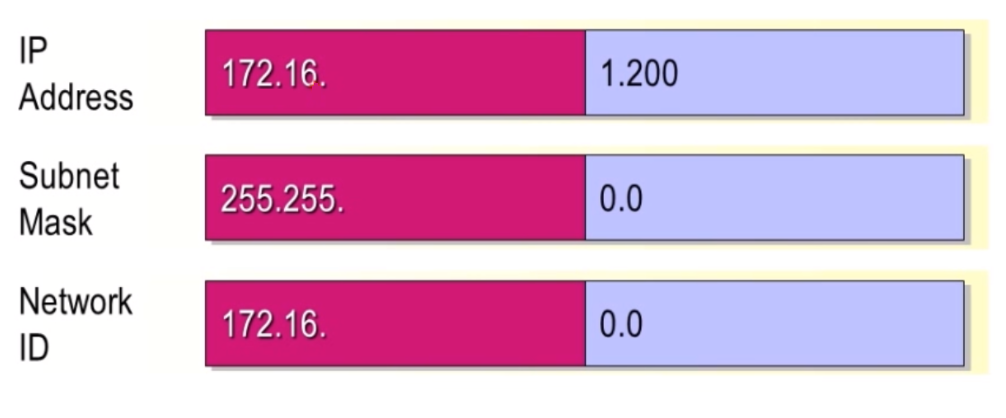
**1.255.255255代表当前网络的广播地址 1.0.0.0代表网络本身,不能分配**

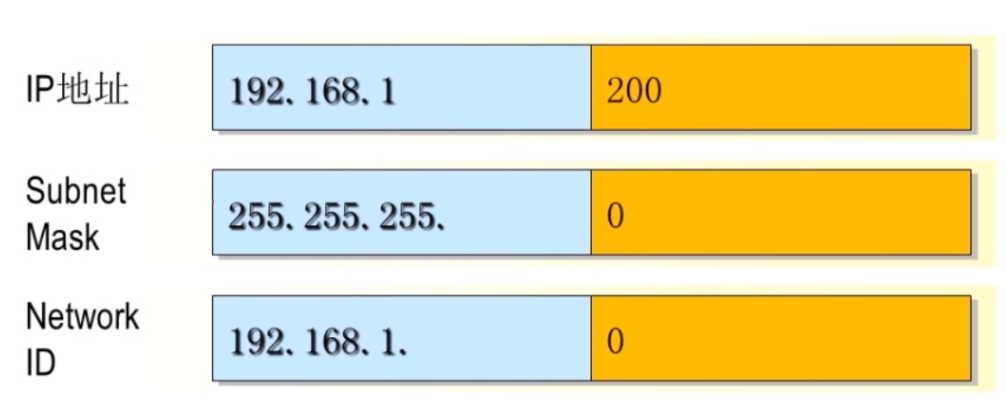
## 子网掩码

### 子网掩码的使用

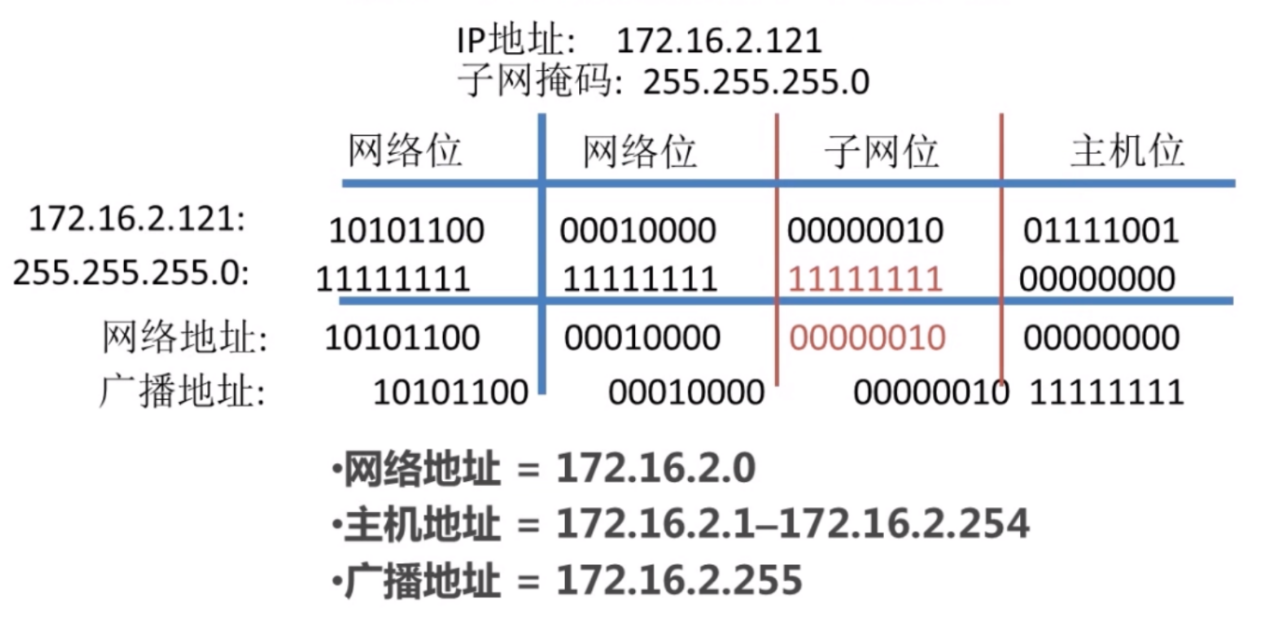
**255.0.0.0 255.255.0.0 255.255.255.0**







### 变长子网掩码及子网规划



**广播地址是专门用于同时向网络中所有工作站发送的一个地址**

## 端口

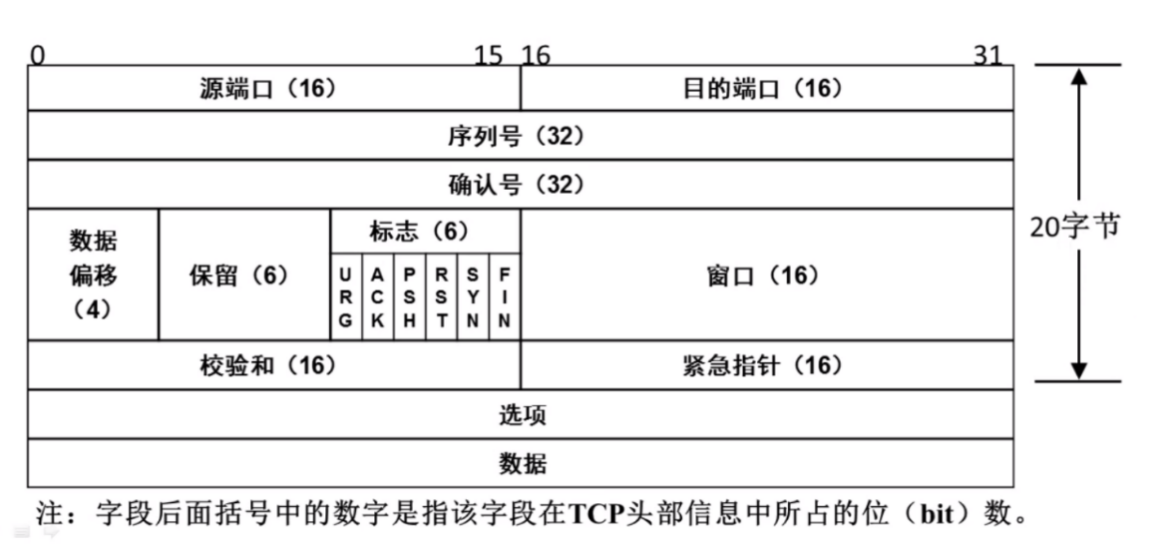
**端口号是什么?**

**端口号有多少种分类?**

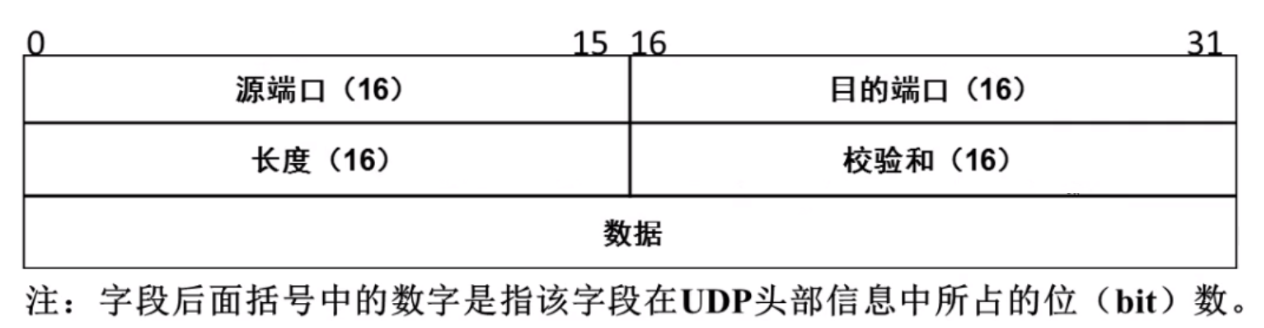
**1024以内常见服务端口、1024以上自己编程用的**

**常见的端口号有哪些?**

### TCP协议包头



### UDP协议包头



### 常见端口号

**FTP(文件传输协议):端口号20 21**

**SSH(安全shell物议):端口号22**

**telnet(远程登录协议):端口号23**

**DNS(域名系统):端口号53**

**http(超文本传输协议):端口号80**

**SMTP(简单邮件传输协议):端口号:25**

**POP3(邮局协议3代):端口号:110**

### 查看本机启用的端口

**[root@localhost ~]# netstat -an**

**选项**

**-a:查看所有连接和监听端口**

**-n:显示IP地址和端口号,而不显示域名和服务名**

## DNS

### 名称解析概述

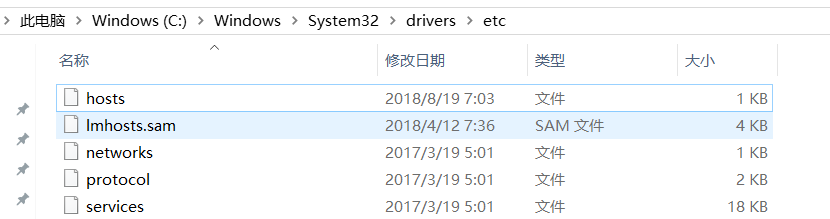
**在互联网中,通过IP地址来进行通信**

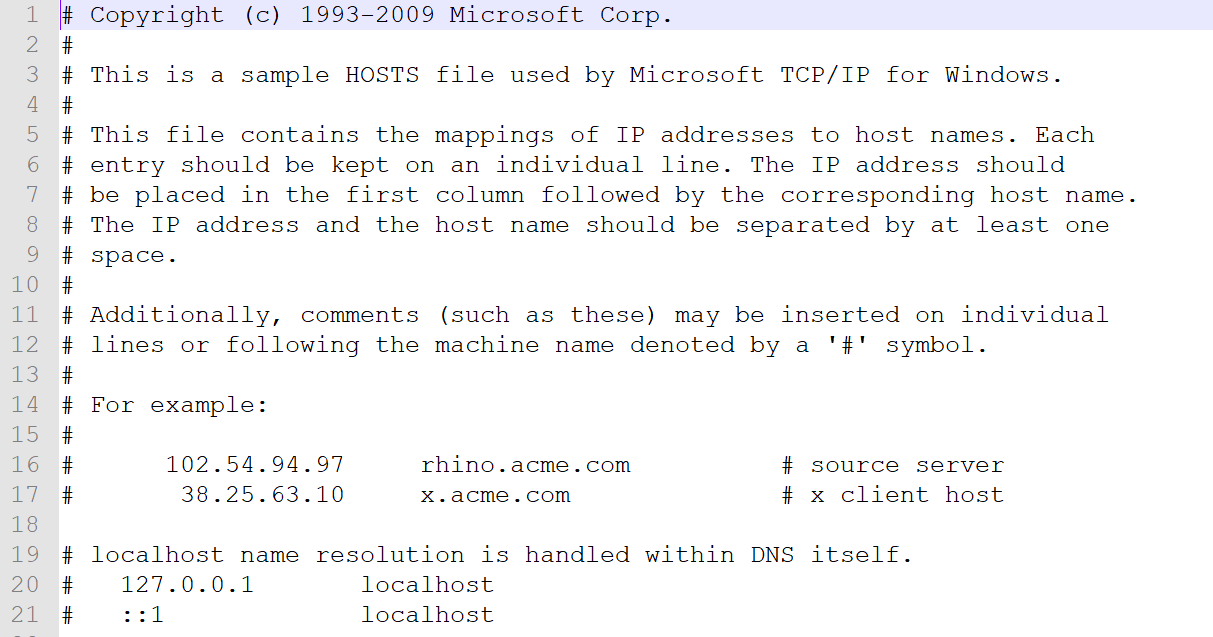
**IP地址用数字表示,记忆起来太困难了(如116213120232)**

**人对域名更加敏感,如:htp://www.google. com/**

**我们把DNS叫做 Domain Name System的缩写, 也就是域名系统的缩写, 也称作名称解析**

### Host文件





### 从 Hosts文件到DNS

**早期 Hosts文件解析域名**

**名称解析效能下降**

**主机维护困难**

**DNS服务**

**层次性**

**分布式**

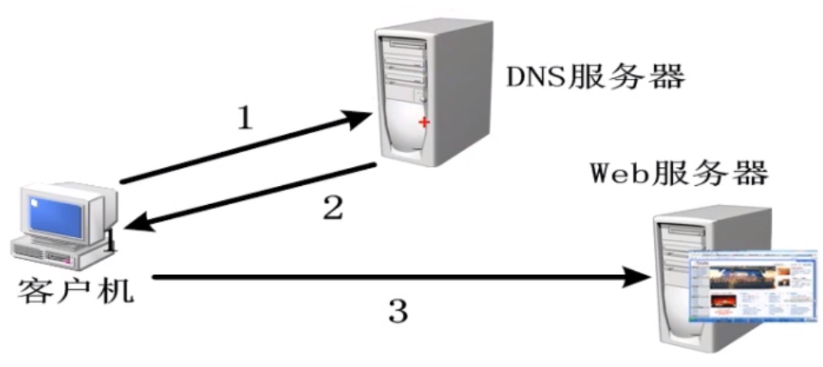
### DNS服务的作用

**将域名解析为IP地址**

**-客户机向DNS服务器发送域名查询请求**

**-DNS服务器告知客户机Web服务器的IP地址**

**-DNS服务器客户机与Web服务器通信**



## 网关

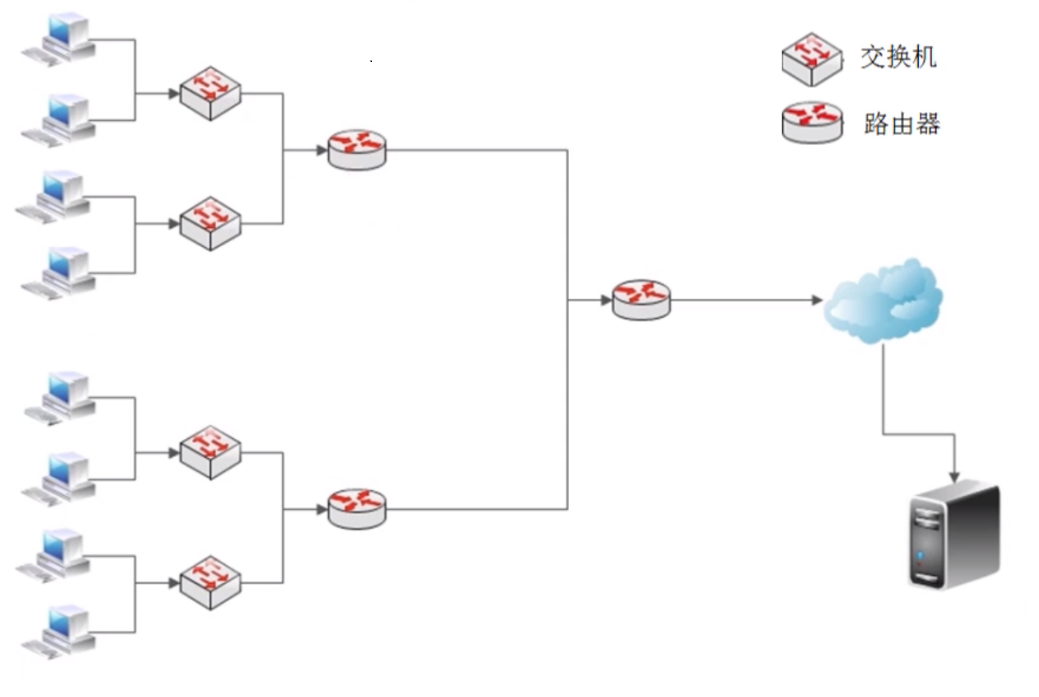
### 网关概念

**1、网关( Gateway)又称网间连接器、协议转换器。**

**2、网关在网络层以上实现网络互连,是最复杂的网络互连设备,仅用于两个高层协议不同的网络互连。**

**3、网关既可以用于广域网互连,也可以用于局域网互连。**

**4、网关是一种充当转换重任的服务器或路由器。**



**网关示意图**

### 网关作用

**网关在所有内网计算机访问的不是本网段的数据包时使用**

**网关负责将内网IP转换成公网IP, 公网IP转换为内网IP**

# 网络配置相关命令

## 网络配置的主要方法：

**ifconfig命令临时配置**

**修改网络配置文件配置**

**图形界面配置(不做说明)**

## ifconfig临时配置

**ifconfig命令**

**ifconfig**

**#查看与配置网络状态命令**

**ifconfig eth0 192. 168.0.200 netmask 255.255.255.0**

**#临时设置eth0网卡的P地址与子网掩码**

## 修改网络配置文件

### 网卡信息文件

**[root@localhost ~]# vim /etc/sysconfig/network-scripts/ifcfg-eth0**

**~~DEVICE=eth0 网卡设备名~~**

**BOOTPROTO=none 是否自动获取IP(none、 static、dhcp)**

**HWADDR=00:0c29:17:c4:09 MAC地址**

**NM\_ CONTROLLED=yes 是否可以由 Network Manager图形管理工具托管**

**ONBOOT=yes 是否随网络服务启动eth0生效,必须设置为yes!!!**

**TYPE= Ethernet 类型为以太网**

**UUID=44b76c8a-b59f-44d5-83fa-7f98fda86b3d 唯一识别码**

**IPADDR=192.168.0.252 IP地址**

**NETMASK=255.255.255.0 子网掩码**

**GATEWAY=192.168.0.1 网关**

**DNS1=202.106.0.20 DNS**

**IPV6INIT=no IPv6没有启用**

**~~USERCTL=no 不允许非root用户控制此网卡~~**

### 主机名文件

**[root@localhost ~]# vim /etc/hostname**

**localhost.localdomain #不像windows中那么重要,在linux中可以重复**

**[root@ localhost~]# hostname [主机名]**

**#查看与临时设置主机名命令**

### DNS配置文件

**[root@localhost ~]# vim /etc/resolv. conf**

**nameserver 202.106.0.20**

**# 有多个dns可以打个空格继续接着写,或者换行写一个 namserver然后接着写**

**search localhost**

### 启动网卡

**[root@localhost -]#t vi/etc/sysconfig/network-scripts/ifcfg-etho**

**把ONBOOT=no**

**改为 ONBOOT=yes**

**[root@localhost - ] systemctl restart network**

**#重启网络服务**

## 网络环境查看命令

### ifconfig命令

**ifconfig**

**#查看与配置网络状态命令**

### 关闭与启动网卡

**Ifdown 网卡设备名**

**#禁用该网卡设备**

**ifup 网卡设备名**

**#启用该网卡设备**

### 查询网络状态

**netstat选项**

**选项:**

**-t:列出TCP协议端口**

**-u:列出UDP协议端口**

**-n:不使用城名与服务名,而使用IP地址和端口号**

**-l:仅列出在监听状态网络服务**

**-a:列出所有的网络连接**

**两个常用方法**

**netstat -tuln**

**#可以判断当前主机开启了哪些服务**

**netstat -an**

**#查看所有**

**netstat -an | grep ESTABLISHED | wc -l**

**#查看有多少人正在连接该主机**

**netstat -rn**

**选项**

**-r:列出路由列表,功能和 route命令一致**

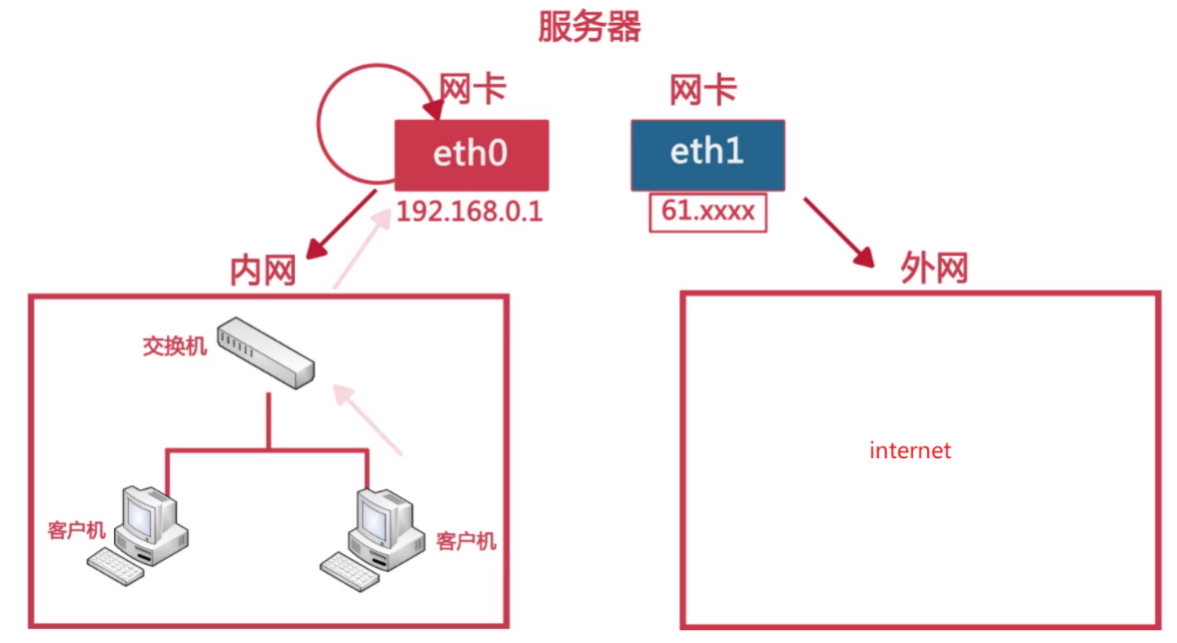
**route -n**

**#查看路由列表(可以看到网关)**

**~~route add default gw 192.168.1.1~~**

**~~#临时设定网关~~**

**临时设定网关的命令不建议用，因为会造成网络不能用**



### 域名解析命令

**nslookup [主机名或IP]**

**#进行域名与IP地址解析,就是翻译域名对应哪个IP**

**[root@localhost ~]# nslookup**

**> server**

**#查看本机DNS服务器**

**> exit**

**#退出**

## 网络测试命令

### ping命令

**ping [选项] [ip或者域名]**

**#探测指定IP或域名的网络状况**

**选项:**

**-c次数:指定ping包的次数**

### telnet命令

**telnet [域名或IP] [端口]**

**#远程管理与端口探测命令，windows和linux中效果一致**

**#由于采用明文传递导致的不安全性，windows和Linux默认都不再开启telnet服务**

**yum install telnet-server**

**#安装服务端**

**yum install telnet.\***

**#安装客服端**

**telnet 192.168.222.60 80**

**# 远程探测**

### traceroute路由跟踪命令

**traceroute [选项] IP或域名**

**#路由跟踪命令**

**选项**

**-n使用IP,不使用域名,速度更快**

**ICMP是(Internet Control Message Protocol)Internet控制报文协议**

### wget下载命令

**wget 下载地址**

**#下载命令**

### tcpdump命令

**tcpdump -i etho -nnX port 21**

**#抓包命令**

**选项**

**-i 指定网卡接口**

**-nn 将数据包中的域名与服务转为IP和端口**

**-X 以十六进制和ASCI码显示数据包内容**

**port指定监听的端口**

**一个有趣的实验**

**yum install vsftpd**

**#安装vsftpd的ftp服务器**

**systemctl stop firewalld**

**#临时关闭防火墙**

**systemctl start vsftpd**

**#启动服务器**

**netstat -tuln**

**useradd user1**

**passwd user1**

**ftp 192.168.222.60**

**#在其他电脑上输入该命令访问**

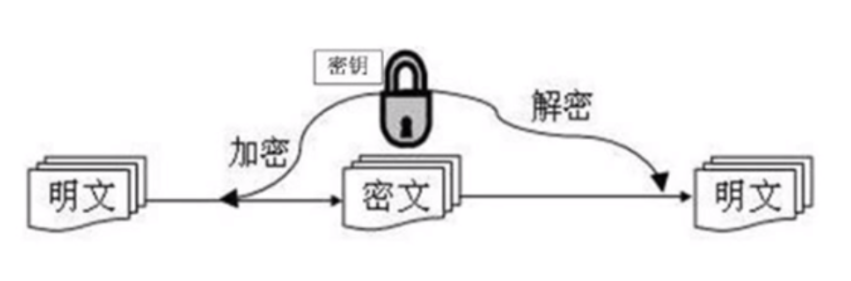
**#此处有口误，ftp和telnet是完全两个不同的协议**

**#我们常见的http协议也是明文传输的，https则是加密传输的安全协议，所以网站都建议使用https**

## ssh协议

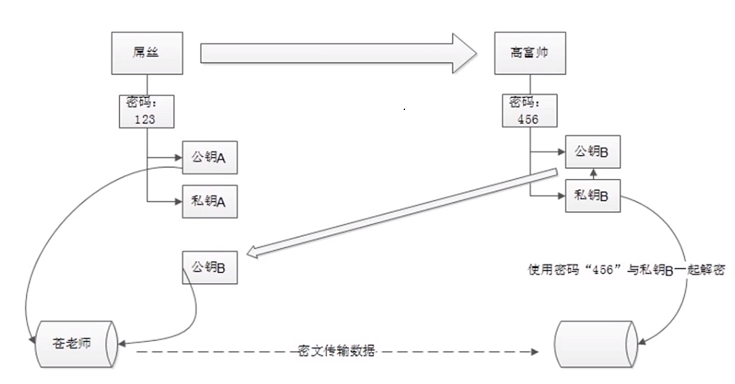
### 对称加密算法

**采用单钥密码系统的加密方法,同一个密钥可以同时用作信息的加密和解密,这种加密方法称为对称加密,也称为单密钥加密**

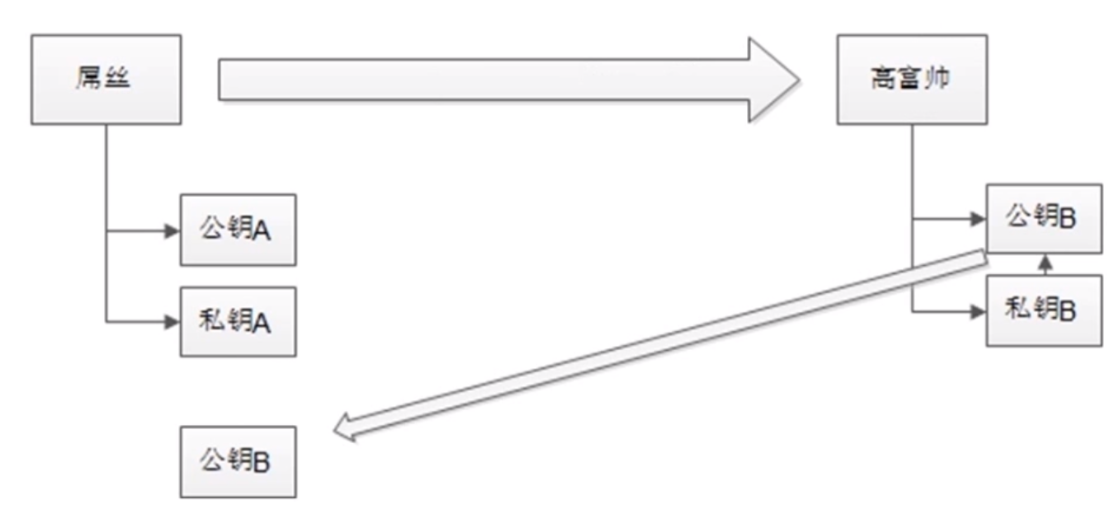


### 非对称加密算法

**非对称加密算法( asymmetric cryptographic algorithm)又名“公开密钥加密算法”,非对称加密算法需要两个密钥公开密钥( public key)和私有密钥( private key)**



### SSH安全外壳协议



### SSH命令

**ssh用户名@ip**

**#远程管理指定 Linux服务器**

**#ssh公钥存放在家目录.ssh目录下的known\_hosts文件中**

**scp [-r] 本地文件 用户名@ip:上传路径**

**#上传文件**

**scp [-r] 用户名@ip:文件路径 本地路径**

**#下载文件**

### 拓展阅读

**ssh远程登录：http://www.ruanyifeng.com/blog/2011/12/ssh\_remote\_login.html**

**数字签名：<http://www.ruanyifeng.com/blog/2011/08/what_is_a_digital_signature.html>**

### 课后实验：SSH免密码登录

**在60机器上操作免密连62,红色为输入的命令,可以只看红色部分**

**[root@localhost ~]# ssh 192.168.222.62**

**root@192.168.222.62's password:**

**#第一次需要输入密码**

**[root@localhost ~]# exit**

**#输入密码后就进到60了，输入exit退出回到60上**

**[root@localhost ~]# ssh-keygen -t rsa**

**#然后再交互中按三次回车**

**[root@localhost ~]# ls .ssh/**

**id\_rsa id\_rsa.pub known\_hosts**

**#发现多了两个文件id\_rsa(私钥文件) id\_rsa.pub(公钥文件)**

**# know\_hosts (已知的主机公钥清单)**

**方式1（推荐方式）,通过ssh-copy-id命令实现:**

**[root@localhost ~]# ssh-copy-id -i ~/.ssh/id\_rsa.pub root@192.168.222.62**

**[root@localhost ~]# ssh 192.168.222.62**

**#这次不再需要输入密码**

**方式2, 通过scp实现（便于理解方式1）：**

**[root@localhost ~]# scp .ssh/id\_rsa.pub root@192.168.222.62:/root/.ssh/authorized\_keys**

**[root@localhost ~]# ssh 192.168.222.62**

**#这次不再需要输入密码**

**有兴趣的同学还可以，尝试用方式1把第二台机器如61，免密码连上62**

## 防火墙

**防火墙是整个数据包进入主机前的第一道关卡。防火墙主要通过Netfilter(数据包过滤机制)与TCPwrappers(程序管理机制)两个机制来管理的，关于数据包过滤机制有两个软件：firewalld与iptables。**

**从CentOS7(RHEL7)开始，官方的标准防火墙设置软件从iptables变更为firewalld，相信不少习惯使用iptables的人会感到十分不习惯，但实际上firewalld更为简单易用。**

**firewalld 的配置储存在 /usr/lib/firewalld/ 和 /etc/firewalld/ 里的各种 XML 文件里，这样保持了这些文件被编辑、写入、备份的极大的灵活性，使之可作为其他安装的备份等等。**

**拓展阅读:**

**redhat安全性指南：https://access.redhat.com/documentation/zh-cn/red\_hat\_enterprise\_linux/7/html/security\_guide/**

**防火墙官方说明: https://access.redhat.com/documentation/zh-cn/red\_hat\_enterprise\_linux/7/html/security\_guide/sec-using\_firewalls**

### 常见命令

**启动： systemctl start firewalld**

**关闭： systemctl stop firewalld**

**重启： systemctl restart firewalld**

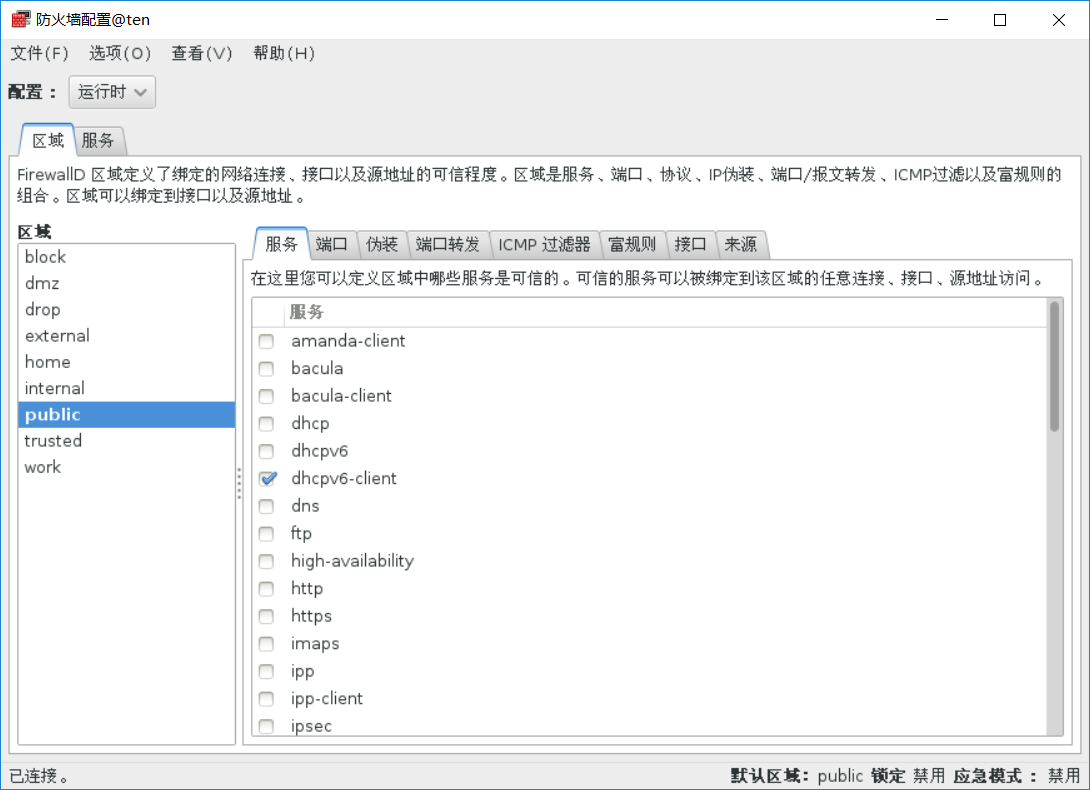
**查看状态： systemctl status firewalld**

**开机禁用  ： systemctl disable firewalld**

**开机启用  ： systemctl enable firewalld**

### 图形化配置工具firewall-config

**[root@localhost ~]# firewall-config &**



### 命令行客户端firewalld-cmd

**查看版本： firewall-cmd --version**

**查看帮助： firewall-cmd --help**

**显示状态： firewall-cmd --state**

**查看所有打开的端口： firewall-cmd --zone=public --list-ports**

**更新防火墙规则： firewall-cmd --reload**

**查看区域信息:  firewall-cmd --get-active-zones**

**查看指定接口所属区域： firewall-cmd --get-zone-of-interface=eth0**

**拒绝所有包：firewall-cmd --panic-on #此命令不要随便尝试，会导致你的ssh中断的**

**取消拒绝状态： firewall-cmd --panic-off**

**查看是否拒绝： firewall-cmd --query-panic**

**怎么用firewall-cmd开启一个端口**

**添加**

**[root@localhost ~]# firewall-cmd --zone=public --add-port=80/tcp --permanent**

**#--permanent永久生效，没有此参数重启后失效**

**重新载入**

**[root@localhost ~]# firewall-cmd --reload**

**查看**

**[root@localhost ~]# firewall-cmd --zone=public --query-port=80/tcp**

**删除**

**[root@localhost ~]# firewall-cmd --zone=public --remove-port=80/tcp --permanent**

### systemctl简介

**systemctl是CentOS7的服务管理工具中主要的工具，它融合之前service和chkconfig的功能于一体。**

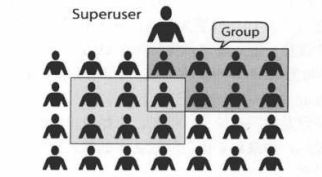
**拓展阅读：http://www.ruanyifeng.com/blog/2016/03/systemd-tutorial-commands.html**

**如:**

**启动一个服务：systemctl start firewalld.service  
关闭一个服务：systemctl stop firewalld.service  
重启一个服务：systemctl restart firewalld.service  
显示一个服务的状态：systemctl status firewalld.service  
在开机时启用一个服务：systemctl enable firewalld.service  
在开机时禁用一个服务：systemctl disable firewalld.service  
查看服务是否开机启动：systemctl is-enabled firewalld.service  
查看已启动的服务列表：systemctl list-unit-files | grep enabled  
查看启动失败的服务列表：systemctl --failed**

# 用户和用户组管理

**Linux系统是一个多用户多任务的分时操作系统，要使服务器的用户，都必须首先向系统管理员申请一个账号，然后以这个账号的身份进入系统。**



## 常见文件目录

**用户文件 /etc/passwd**

**密码文件 /etc/shadow**

**组文件 /etc/group**

## 查看当前登陆用户名whoami

**whoami**

**#我是谁**

## 查看用户信息id

**id -a 用户名**

**如：id -a root**

## 添加用户useradd

**useradd [选项] [用户名]**

**选项**

**-d 指定家目录，**

**-m 自动建立用户的登入目录**

**-s指定登陆时使用的shell**

**-g 指定用户所属的群组**

**-G指定用户所属的附加群组**

**如：**

**useradd ten**

## 修改用户usermod

**usermod [选项] [用户名]**

**选项**

**-d 指定家目录，**

**-m 自动建立用户的登入目录**

**-s指定登陆时使用的shell**

**-g 指定用户所属的群组**

**-G指定用户所属的附加群组**

## 删除用户userdel

**userdel [选项] [用户名]**

**选项**

**-r 参数代表删除用户时同事删除用户主目录**

**如userdel -r ten**

## 修改用户密码passwd

**passwd [选项] [用户名]**

**#默认修改当前用户的密码**

**选项:**

**-d 使账号无口令**

**-f 强迫用户下次登录时修改口令**

**-l 禁用账号 lock**

**-u解禁账号 unlock**

## 查看用户所在组

**groups 用户名**

**#显示用户名所在的组，默认查询当前用户所在组**

## 新增用户组groupadd

**groupadd 选项 组名**

**选项:**

**-g 组id号,指定新用户组的组id号**

## 删除用户组groupdel

**groupdel 用户组名**

## 修改用户组命令groupmod

**groupmod [选项] [用户组]**

**选项：**

**-g GID为用户组指定新的组标识号。**

**-o 与-g选项同时使用，用户组的新GID可以与系统已有用户组的GID相同。**

**-n新用户组 将用户组的名字改为新名字**

## 用户组切换newgrp

**newgrp 用户组名**

**# 如果一个用户同时属于多个用户组，那么用户可以在用户组之间切换，以便具有其他用户组的权限**

# 权限管理

**在服务器上,所有用户都用管理员权限登录吗？**

**答案是no，所以需要进行权限管理。**

**权限设定思路**

* **设定权限需求**
* **按照权限的需求，设定权限**



## 文件基本权限

### 文件权限类型

**-rwxr-xr-x.**

**- 文件类型(-文件 d目录 l软链接文件)**

**此外还有**

**b:块设备文件(例如硬盘、光驱等)**

**p:管道文件**

**c:字符设备文件（例如猫等串口设备）**

**s:套接字文件(例如启动一个MySql服务器时会产生一个mysql.sock文件)**

**rwx r-x r-x**

**u所有者 g所属组 o其他人**

**r读 w写x执行**

**核心原则**

**分配文件基本权限时,核心原则:在最小权限情况下能够实现要求即可!**

### chmod修改权限命令

**chmod [选项] 模式 文件名**

**选项**

**-R 递归**

**模式**

**[ugoa][+-=] [rwx]**

**[mode=421]**

**a 代表所有all**

**例如**

**chmod u+x canals.av**

**chmod g+w,o+w furong.av**

**chmod a=rwx fengJie.av**

**u所有者 g所属组 o其他人 a所有人**

**权限的数字表示**

**为什么不是1、2、3？**

**r----4**

**w----2**

**x ----1**

**rwxr-xr-x**

**755**

**例如：**

**chmod 755 hello.sh**

### 权限对文件的作用

**r:读取文件内容( cat more head tail)**

**w:编辑、新增、修改文件内容(vi echo)**

**#但是不包含删除文件。为什么呢？**

**x:可执行**

### 权限对目录的作用

**r:可以查询目录下文件名(ls)**

**w:具有修改目录结构的权限。如新建文件和目录,删除此目录下文件和目录,重命名此目录下文件和目录,剪切( touch rm mv cp)**

**x:可以进入目录(cd)**

### 权限对目录与文件作用的异同

**通俗来说，不那么准确的说: 对目录来讲:最高权限是w**

**0(---) 5(r-x) 7 (rwx) 对目录而言只有057权限**

**4(r--) 1(--x) 6(-wx) 416权限对目录没有意义**

### chown修改文件的所有者

**chown 用户名 文件名**

**例如: chown user1:group1 feng.av**

### chgrp修改文件的所属组

**chgrp 组名 文件名**

**例如: chgrp groupl fengjie.av**

### 权限案例

**让用户对文件及目录拥有一定的权限**

**要求**

**拥有一个av目录**

**让加藤老师拥有所有的权限**

**让本课程学员有查看的权限**

**其他所有人不许查看这个目录**

## 文件默认权限

### 查看默认权限的命令umask

**umask**

**#查看默认权限**

**0022**

**第一位0:文件特殊权限**

**022:文件默认权限**

### 文件的默认权限

**文件默认不能建立为执行文件,必须手工赋予执行权限**

**所以文件默认权限最大为666**

**默认权限需要换算成字母再相减**

**建立文件之后的默认权限,为666减去 umask值**

**例如:**

**文件默认最大权限666 umask值033**

**-rw-rw-rw-减去 -----wx-wx等于-rw-r--r--**

**文件默认最大权限666 umask值033**

**-rw-rw-rw- 减去 -----wx-wx 等于 -rw-r--r—**

### 目录的默认权限

**目录默认权限最大为777**

**默认权限需要换算成字母再相减**

**建立文件之后的默认权限,为777减去 umask值**

**例如**

**目录默认最大权限为777 umask值022**

**-rwxrwxrwx减去-----w--w-等于-rwxr-xr-x**

### 修改 umask值

**临时修改**

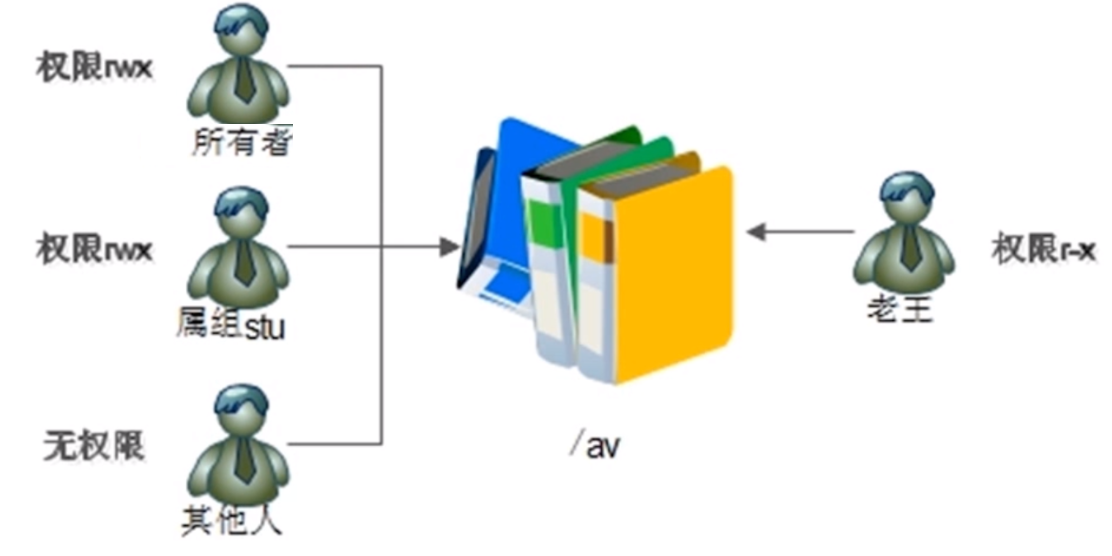
**umask 0002**

**永久修改**

**vi /etc/profile**

## ACL权限

### ACL权限简介



**Centos7现在已经默认开启，无需再特别开启ACL权限了**

### ~~查看分区ACL权限是否开启~~

**~~[root@localhost ~]# xfs\_growfs /dev/mapper/centos-root | grep acl~~**

**~~#xfs文件系统请使用该命令~~**

**~~[root@localhost ~]# dumpe2fs -h /dev/sda5~~**

**~~#dumpe2fs命令是查询指定分区详细文件系统信息的命令~~**

**~~选项~~**

**~~-h仅显示超级块中信息,而不显示磁盘块组的详细信息~~**

### ~~临时开启分区ACL权限~~

**~~[root@localhost ~ ]# mount -o remount,acl /~~**

**~~#重新挂载根报分区,并挂载加入acl权限~~**

### ~~永久开启分区ACL权限~~

**~~[root@localhost ~]# vi/etc/fstab~~**

**~~UUID=c2ca6f57-b15c-43ea- bca0-f239083d8bd2 / ext4 defaults,acl 1 1~~**

**~~#加入acl~~**

**~~[root@localhost ~]# mount -o remount /~~**

**~~#重新挂载文件系统互重启动系统,使修生效~~**

### 查看ACL命令

**[root@ localhost~]# getfacl文件名**

**#查看acl权限**

### 设定ACL权限的命令

**[root@ localhost~]# setfacl 选项 文件名**

**选项:**

**-m 设定ACL权限**

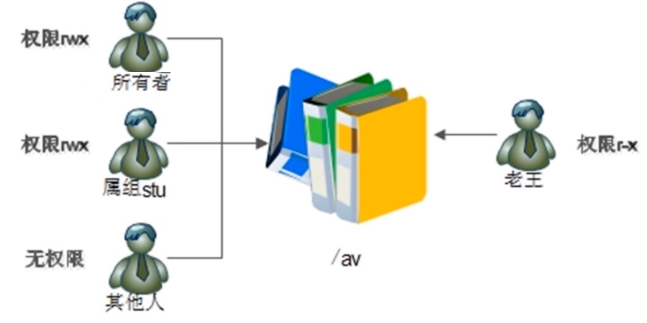
**-x 删除指定的ACL权限**

**-b 删除所有的ACL权限**

**-k 删除默认ACL权限**

**-R 递归设定ACL权限**

### 给用户设定ACL权限



**[root@ localhost~]# useradd ten**

**[root@ localhost~]# groupadd stu**

**[root@ localhost~]# mkdir av**

**[root@ localhost~]# chown ten:stu av**

**[root@ localhost~]# chmod 770 av**

**[root@ localhost~]# useradd lw**

**[root@ localhost~]# setfacl -m u:Iw:rx av**

**#给用户st赋予r-x权限, 使用 "u:用户名:权限” 格式**

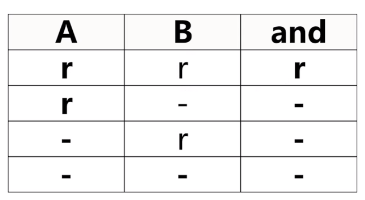
**[root@ localhost~]# getfacl av**

**[root@ localhost~]# setfacl -m u:Iw:rx av**

**u: 用户 g:组 o:其他人**

### 最大有效权限mask

**mask是用来指定最大有效权限的。如果我给用户赋予了ACL权限,是需要和mask的权限“相与”才能得到用户的真正权限**



### 修改最大有效权限

**[root@ localhost ~]# setfacl -m m:rx 文件名**

**#定mask权限为r-x。使用”m:权限”格式**

### 删除ACL权限

**root@ localhost/]# setfacl -x u:用户名 文件名**

**#删除指定用户的ACL权限**

**[root@ ocalhost/]# setfacl -x g:组名 文件名**

**#删指定用户组的ACL权限**

**[root@ localhost/]# setfacl -b 文件名**

**#会删除文件的所有的ACL权原**

### 递归ACL权限

**递归是父目录在设定ACL权限时,所有的子文件和子目录也会拥有相同的ACL权限。**

**setfacl -m u:用户名:权限 -R 目录名**

**#递归权限仅能赋予目录,不能赋予文件**

**会存在权限溢出的问题！**

### 默认ACL权限

**默认ACL权限的作用是如果给父目录设定了默认ACL权限,那么父目录中所有新建的子文件都会继承父目录的ACL权限。**

**setfacl -m d:u:用户名:权限 目录名**

**#default**

## sudo权限

### sudo权限

**root把本来只能超级用户执行的命令赋予普通用户执行。**

**sudo的操作对象是系统命令**

### sudo使用

**[root@localhost ~]# visudo**

**#实际修改的是/etc/sudoers文件**

**root ALL=(ALL) ALL**

**#用户名 被管理主机的地址=(可使用的身份) 授权命令(绝对路径)**

**# %wheel ALL=(ALL) ALL**

**#%组名 被管理主机的地址=(可使用的身份) 授权命令(绝对路径)**

### 例子1普通用户执行sudo赋予的命令

**[root@localhost ~]# visudo**

**ten ALL=(ALL) /sbin/shutdown -r now**

**#最后插入**

**[root@localhost ~]# su user1**

**[user1@localhost ~]$ sudo -l**

**#查看可用的sudo命令**

**[user 1@localhost ~]$ sudo /sbin/shutdown -r now**

**#普通用户执行sudo赋予的命令**

### 例子2:授权普通用户可以添加其他用户

**[root@localhost ~]# visudo**

**user1 ALL=/usr/sbin/useradd**

**user1 ALL=/usr/bin/passwd**

**#授予用户设定密码的权限**

**user1 ALL=/usr/bin/passwd [A-Za-z0-9]\*, !/usr/bin/passwd “”, !/usr/bin/passwd root**

**[root@localhost ~]# ll /etc/shadow**

**#密码文件**

## 文件特殊权限

### SetUID

#### SetUID的功能

* **只有可以执行的二进制程序才能设定SUID权限**
* **命令执行者要对该程序拥有x(执行)权限**
* **命令执行者在执行该程序时获得该程序文件属主的身份(在执行程序的过程中灵魂附体为文件的属主)**
* **SetUid权限只在该程序执行过程中有效,也就是说身份改变只在程序执行过程中有效**

#### 设定 SetUID的方法

**4代表SUID**

* **chmod 4755 文件名**
* **chmod u+s 文件名**

**s = S + x**

**u+s ----- SUID**

**g+s ---- SGID**

**o+s ---- SBIT**

#### 案例分析

**[root@localhost ~]# ll /etc/shadow**

**passwd命令拥有 SetUID权限,所以普通可以修改自己的密码**

**[root@localhost ~]# ll /usr/bin/passwd**

**-rwsr-xr-x. 1 root root 27832 6月 10 2014 /usr/bin/passwd**

**cat命令没有 SetUID权限,所以普通用户不能查看/etc/shadow文件内容**

**[root@localhost ~]# ll /usr/bin/cat**

**-rwxr-xr-x. 1 root root 54048 6月 10 2014 /usr/bin/cat**

#### 取消 SetUID的方法

**chmod 0755文件名**

**chmod u-s文件名**

#### 危险的 SetUID

**关键目录应严格控制写权限。比如”/”、“/usr"等**

**用户的密码设置要严格遵守密码三原则**

**对系统中默认应该具有 SetUID权限的文件作一列表,定时检查有没有这之外的文件被设置了 SetUID权限**

**下面是检查有无其他文件有SUID权限的脚本**

**此处由于没有学shell编程，可以在学了以后再看**

**find / -perm -4000 -o -perm -2000 > /root/suid.list**

**#将当前有suid和sgid权限的文件记录起来**

**#然后新建扫描脚本如下**

**#!/bin/bash**

**find / -perm -4000 -o -perm -2000 > /tmp/setuid.check**

**#搜索系统中所有拥有SID和SGD的文件,并保存到临时目录中。**

**for i in $(cat /tmp/setuid.check)**

**#做循环,每次循环取出临时文件中的文件名**

**do**

**grep $i /root/suid.list > /dev/null**

**#比对这个文件名是否在模板文件中**

**if [ "$?" != "0" ];then**

**#检测上一个命令的返回信,如果不为0,证明上一个命令报错**

**echo "$i isn't in listfile!" >> /root/suid\_log\_$(date +%F)**

**#如果文件名不再模板文件中,则输出错误信息,并把报错报错到日志中**

**fi**

**done**

**rm -rf /tmp/setuid.check**

**#删除临时文件**

### SetGID

#### 设定 SetGID的方法

**2代表SGID**

* **chmod 2755 文件名**
* **chmod g+s 文件名**

#### SetGID针对文件的作用

* **只有可执行的二进制程序才能设置SGID权限**
* **命令执行者要对该程序拥有ⅹ(执行)权限**
* **命令执行在执行程序的时候,组身份升级为该程序文件的属组**
* **SetGID权限同样只在该程序执行过程中有效，也就是说组身份改变只在程序执行过程中有效**

#### 案例分析

**[root@localhost opt]# ll /usr/bin/locate**

**-rwx--s--x. 1 root slocate 40496 6月 10 2014 /usr/bin/locate**

**[root@localhost opt]# ll /var/lib/mlocate/mlocate.db**

**-rw-r-----. 1 root slocate 2380254 8月 25 11:12 /var/lib/mlocate/mlocate.db**

**过程**

* **/usr/bin/locate是可执行二进制程序,可以赋予SGID**
* **执行用户lamp对/usr/bin/ locate命令拥有执行权限**
* **执行/usr/bin/locate命令时,组身份会升级为slocate组,而 dislocate组对var/lib/ locate/ locate.db数据库拥有r权限所以普通用户可以使用 ocate命令查询 locate db数据库**
* **命令结束,lamp用户的组身份返回为lamp组**

#### SetGID针对目录的作用

* **普通用户必须对此目录拥有r和x权限,才能进入此目录**
* **普通用户在此目录中的有效组会变成该目录的属组**
* **若普通用户对此目录拥有w权限时,新建的文件的默认属组是这个目录的属组**

**案例演示**

**[root@localhost ~]# cd /opt/**

**[root@localhost opt]# mkdir test**

**[root@localhost opt]# chmod 2777 test**

**[root@localhost opt]# ll -d test/**

**drwxrwsrwx. 2 root root 4096 8月 25 15:19 test/**

**[root@localhost opt]# su ten**

**[ten@localhost opt]$ cd test/**

**[ten@localhost test]$ touch abc**

**[ten@localhost test]$ ll abc**

**-rw-rw-r--. 1 ten root 0 8月 25 15:20 abc**

### Sticky BIT

#### SBIT粘着位作用

* **粘着位目前只对目录有效**
* **普通用户对该目录拥有w和x权限,即普通用户可以在此目录拥有写入权限**
* **如果没有粘着位,因为普通用户拥有W权限,所以可以删除此目录下所有文件,包括其他用户建立的文件。一但赋予了粘着位,除了root可以删除所有文件,普通用户就算拥有w权限,也只能删除自己建立的文件,但是不能删除其他用户建立的文件**

**设置粘着位**

* + **chmod 1777 目录名**
  + **chmod o+t 目录名**

**取消粘着位**

* + **chmod 0777 目录名**
  + **chmod o-t 目录名**

**案例：**

**[root@localhost ~]# cd /home/ten/**

**[root@localhost ten]# mkdir demo**

**[root@localhost ten]# chmod 777 demo**

**[root@localhost ten]# touch demo/aaa**

**[root@localhost ten]# su ten**

**[ten@localhost ~]$ rm -rf demo/aaa**

**[ten@localhost ~]$ exit**

**exit**

**[root@localhost ten]# chmod 1777 demo**

**[root@localhost ten]# touch demo/bbb**

**[root@localhost ten]# su ten**

**[ten@localhost ~]$ rm -rf demo/bbb**

**rm: 无法删除"demo/bbb": 不允许的操作**

## chattr权限(不可改变权限)

### chattr命令格式

**chattr [+-=] [选项] 文件或目录名**

**+: 增加权限**

**-: 删除权限**

**=: 等于某权限**

**changattr的缩写**

**选项:**

**-i: (insert)如果对文件设置属性,那么不允许对文件进行删除、改名,也不能添加和修改数据;如果对目录设置属性,那么只能修改目录下文件的数据,但不允许建立和删除文件。**

**-a: (append)如果对文件设置a属性,那么只能在文件中增加数据,但是不能删**

**除也不能修改数据;如果对目录设置a属性,那么只允许在目录中建立**

**和修改文件,但是不允许删除**

### 查看文件系统属性

**[root@ Localhost~]# Isattr选项 文件名**

**选项**

**-a显示所有文件和目录**

**-d若目标是目录,仅列出目录本身的属性,而不是子文件的**

**此外linux中还有pam和selinux等，因为使用不多，就不做更多说明了**

# Linux软件安装

**课程目的:学习 Linux中的软件安装方法**

## 软件包分类

* **源码包**
* **二进制包(RPM包、系统默认包)**
* **脚本安装包(实际上用到的还是上述两种)**

### 源码包

**源码包的优点:**

* **开源,如果有足够的能力,可以修改源代码**
* **可以自由选择所需的功能**
* **软件是编译安装,所以更加适合自己的系统,更加稳定也效率更高**
* **卸载方便**

**源码包的缺点**

* **安装过程步骤较多,尤其安装较大的软件集合时(如LAMP环境搭建),容易出现拼写错误**
* **编译过程时间较长,安装比二进制安装时间长**
* **因为是编译安装,安装过程中一旦报错新手很难解决**

**如果只有源码包可用怎么办呢? 初学者很苦恼!!!**

**源码包安装不适合初学者!!!**



### RPM包

**二进制包的优点**

* **包管理系统简单,只通过几个命令就可以实现包的安装、升级、查询和卸载**
* **安装速度比源码包安装快的多**

**二进制包缺点**

* **经过编译,不再可以看到源代码**
* **功能选择不如源码包灵活**
* **依赖性**

### 脚本安装包

**所谓的脚本安装包,就是把复杂的软件包安装过程写成了程序脚本,初学者可以执行程序脚本实现一键安装。但实际安装的还是源码包和二进制包**

* **优点:安装简单、快捷**
* **缺点:完全丧失了自定义性**

## rpm命令管理

### RPM包的来源

**RPM包在系统光盘中**

### RPM包命名原则

**httpd-2.4.6-17.el7.centos.1.x86\_64.rpm**

**httpd软件包名**

**2.2.15软件版本**

**15软件发布的次数**

**el6.centos适合的Linux平台**

**x86\_64适合的硬件平台**

**rpm rpm包扩展名**

### RPM包依赖性

**树形依赖: a→b→c**

**环形依赖: a→b→c→a**

**#需要把a、b、c放到一条命令中安装**

**模块(库文件)依赖,查询网站:www.rpmfind.net**

**[root@localhost Packages]# rpm -ivh mysql-connector-odbc-…**

### 包全名与包名

**包全名:操作的包是没有安装的软件包时,使用包全名,而且要注意路径**

**包名:操作已经安装的软件包时,使用包名是搜索/var/lib/rpm/中的数据库**

### RPM安装

**rpm -ivh 包全名**

**选项:**

**-i( instal)安装**

**-v( verbose)显示详细信息**

**-h(hash)显示进度**

**--nodeps不检测依赖性**

**#案例(不要死记硬背，要学习解决问题的方法)**

**rpm -ivh httpd-…**

### RPM包升级

**rpm -Uvh 包全名**

**选项:**

**-U( upgrade)升级**

### RPM卸载

**rpm -e 包名**

**选项:**

**-e(erase)卸载**

**--nodeps不检查依赖性**

### 查询是否安装

**由于yum不能查询哪些软件已经安装，所以此处的rpm查询还是很重要的**

**rpm -q 包名**

**#查询包是否安装**

**-q 查询( query)**

**rpm -qa | grep ‘httpd’**

**#查询所有已经安类的RPM包，推荐配合管道符使用该方法查询**

**-a 所有(all)**

### 查询软件包详细信息

**rpm -qi包名**

**选项:**

**-i 查询软件信息( information)**

**-p 查询未安装包信息( package)，软件包信息是写在了rpm文件里面的，所以就算未安装也查询**

**作用:可以看到官方网站**

### 查询包中文件安装位置

**rpm -ql 包名**

**选项:**

**-l 列表(ist)**

**-p 查询未安装包信息( package)**

### rpm包默认安装位置

|  |  |
| --- | --- |
| **RPM包默认安装路径** | |
| **/etc/** | **配置文件安装目录** |
| **/usr/bin/** | **可执行的命令安装目录** |
| **/usr/lib/** | **程序所使用的函数库保存位置** |
| **/usr/share/doc/** | **基本的软件使用手册保存位置** |
| **/usr/share/man/** | **帮助文件保存位置** |

### 查询系统文件属于哪个RPM包

**rpm -qf 系统文件名**

**选项**

**-f 查询系统文件属于哪个软件包(file)**

### 查询软件包的依赖性

**rpm -qR 包名**

**选项:**

**-R 查询软件包的依赖性( requires)**

**-p 查询未安装包信息( package)**

### RPM包校验

**rpm -V 已安装的包名**

**选项**

**-V 校验指定RPM包中的文件( verify)**

**验证内容中的8个信息的具体内容如下:**

* **S文件大小是否改变**
* **M文件的类型或文件的权限(rwx)是否被改变**
* **5文件MD5校验和是否改变(可以看成文件内容是否改变)**
* **D设备的主从代码是否改变**
* **L文件路径是否改变**
* **U文件的属主(所有者)是否改变**
* **G文件的属组是否改变**
* **T文件的修改时间是否改变**

**文件类型**

* **c配置文件( config file)**
* **d普通文档( documentation)**
* **g“鬼”文件( ghost file),很少见,就是该文件不应该被这个RPM包包含**
* **L授权文件( license file)**
* **r描述文件( read me)**

### RPM包中文件提取

**思考当你的系统文件丢失的时候怎么办？**

**[root@localhost~]# rpm2cpio 包全名 | cpio -idv .文件绝对路径**

* **rpm2cpio**

**#将rpm包转换为cpio格式的命令**

* **cpio**

**#是一个标准维工具,它用于创建软件档案文件和从档案文件中提取文件**

**[root@localhost~]# cpio选项 < [文件|设备]**

**选项**

**-i:copy-in模式，还原**

**-d:还原时自动新建目录**

**-v:显示还原过程**

**案例:**

**[root@localhost ~]# rpm -qf /bin/ls**

**#查询ls命令属于哪个软件包**

**[root@localhost ~]# mv /bin/ls /tmp/**

**#造成命令误除假象**

**[root@localhost ~]# rpm2cpio /mnt/cdrom/Packages/coreutils-8.22-11.el7.x86\_64.rpm | cpio -idv ./usr/bin/ls**

**#提取RPM包中对应的命令./usr/bin/ls到当前目录的./usr/bin/ls下**

**[root@localhost ~]# cp ./usr/bin/ls /usr/bin/**

**#把命令复制会的/bin/目录,修复文件丢关**

## yum在线安装

**安装过程中rpm包依赖性太强**

**如果所有rpm包都是手工安装,则rpm包使用难度较大**

**于是出现了yum在线安装**

**好处：yum在线安装的好处是将所有软件包放到官方服务器上,当进行yum在线安装时,可以自动解决依赖性问题。**

**redhat的yum在线安装需要付费，而centos中可以免费使用**

### yum源文件

**[root@localhost ~]# ll /etc/yum.repos.d/**

**[root@localhost ~]# vi /etc/yum.repos.d/CentOS-Base.repo**

**[base] 容器名称,一定要放在[]中**

**name 容器说明,可以自己随便写**

**mirrorlist 镜像站点,这个可以注释掉**

**baseurl 我们的yum源服务器的地址。默认是 Centos官方的yum源服务**

**器,是可以使用的,如果你觉得慢可以改成你喜欢的yum源地址**

**enabled 此容器是否生效,如果不写或写成 enable=1都是生效,写成**

**enable=0就是不生效**

**gpgcheck 如果是1是指RPM的数字证书生效,如果是0则不生效**

**gpgkey 数字证书的公钥文件保存位置。不用修改**

**疑问：**

**如果没有网络,如何使用yum源?**

### 光盘搭建yum源

#### 插入光盘

#### 挂载光盘

**mkdir /mnt/cdrom**

**#建立挂载点**

**mount /dev/sr0 /mnt/cdrom/**

**#挂载光盘**

#### 使网络yum源失效

**cd /etc/yum.repos.d/**

**#进入yum源目录**

**mv Centos-Base.repo Centos-Base.repo.bak**

**#修改yum源文件后缀名,使其失效**

#### 使光盘yum源生效

**[root@localhost yum.repos.d]# vim CentoS-Media.repo**

**[c7-media]**

**name=CentOS-$releasever - Media**

**baseurl=file:///mnt/cdrom**

**#地址为你自己的光盘挂载地址**

**gpgcheck=1**

**gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7**

**[root@localhost ~]# yum clean all**

**# 清理本地缓存**

**[root@localhost ~]# yum list**

### yum命令

#### 查询

**yum list**

**#查询所有可用软件包列表**

**yum search关键字**

**#搜索服务器上所有和关键字指关的包**

#### 安装

**yum -y install 包名**

**选项**

**-install 安装**

**-y 自动回答yes**

**例如: yum -y install gcc**

#### 升级

**yum -y update 包名**

**选项**

**-update 升级**

**-y 自动回答yes**

**千万不要直接运行yum -y update！！！**

**也不要随随便便升级！！！**

#### 卸载

**yum -y remove 包名**

**选项:**

**-remove 卸载**

**-y 自动回答yes**

**服务器使用最小化安装,用什么软件安装什么,尽量不卸载！！！**

### yum安装nginx

#### 添加yum源

**vim /etc/yum.repos.d/nginx.repo**

**输入以下信息**

**[nginx]**

**name=nginx repo**

**baseurl=http://nginx.org/packages/centos/7/$basearch/**

**gpgcheck=0**

**enabled=1**

#### 安装nginx

**yum -y install nginx**

### yum软件组管理命令

**LANG=en\_US**

**LANG=zh\_CN.utf8**

**yum grouplist**

**#列出所有可用的软件组列表**

**yum groupinstall软件组名**

**#安装指定教件组,组名可以grouplist查询出来**

**yum groupremove 软件组名**

**#卸载指定软件组**

## 源码包管理

### 源码包与RPM包区别

#### 概念上不同

#### 安装位置的不同

##### RPM包安装位置

**一般是安装在默认位置中**

|  |  |
| --- | --- |
| **RPM包默认安装路径** | |
| **/etc/** | **配置文件安装目录** |
| **/usr/bin/** | **可执行的命令安装目录** |
| **/usr/lib/** | **程序所使用的函数库保存位置** |
| **/usr/share/doc/** | **基本的软件使用手册保存位置** |
| **/usr/share/man/** | **帮助文件保存位置** |

**rpm安装也可以指定安装位置**

**> rpm --help | grep prefix**

##### 源码包安装位置

**安装在指定位置当中,一般是**

**/usr/local/软件名/**

**#源码包没有卸载命令**

##### 安装位置不同带来的影响

**RPM包安装的服务可以使用系统服务管理(systemctl)来管理,例如RPM包安装的 apache的启动方法是:**

**> systemctl start httpd**

**而源码包安装的服务则不能被服务管理命令管理,因为没有安装到默认路径中。所以只能用绝对路径进行服务的管理**

**如**

**> /usr/local/apache/bin/apachectl start**

### 源码包安装过程

#### 安装准备

* **安装C语言编译器**
* **下载httpd源码包**

**由于新版安装会有很多问题，此处请大家下载**

**百度云盘httpd-2.2.9.tar.gz**

**链接：https://pan.baidu.com/s/1xcLL1y8V\_aaY22I2-vtlQQ 密码：26fl**

**更多查看官网：http://www.apache.org/dyn/closer.cgi**

**rpm包和源码包,选择哪一个呢?**

#### 安装注意事项

* **源代码保存位置:/usr/local/src/**
* **软件安装位置:/usr/local/**
* **如何确定安装过程报错:**

**>安装过程停止**

**>并出现eror、 Warning或no的提示**

#### 源码包安装过程

* **下载源码包**
* **解压缩下载的源码包**
* **进入解压缩目录**
* **./configure软件配置与检查**
* **定义需要的功能选项。**
* **检测系统环境是否符合安装要求**
* **把定义好的功能选项和检测系统环境的信息都写入Makefile文**

**件,用于后续的编辑。**

* **make编译**
* **make clean 假如报错则执行清除**
* **make install编译安装**

**案例1源码安装apache服务器:**

**[root@localhost ~]# yum install gcc**

**[root@localhost src]# cd /usr/local/src/**

**[root@localhost src]# tar -zxvf httpd-2.2.9.tar.gz**

**[root@localhost src]# cd httpd-2.2.9/**

**[root@localhost httpd-2.2.9]# ./configure --prefix=/usr/local/apache2**

**[root@localhost httpd-2.2.9]# make**

**[root@localhost httpd-2.2.9]# make install**

**[root@localhost apache2]# /usr/local/apache2/bin/apachectl start**

**案例2源码安装nginx服务器:**

**[root@localhost ~]# cd /usr/local/src/**

**[root@localhost src]# wget <http://nginx.org/download/nginx-1.14.0.tar.gz>**

**[root@localhost src]# tar -zxvf nginx-1.14.0.tar.gz**

**[root@localhost src]# cd nginx-1.14.0/**

**[root@localhost nginx-1.14.0]# yum -y install openssl openssl-devel**

**[root@localhost nginx-1.14.0]# ./configure --prefix=/usr/local/nginx --with-http\_ssl\_module**

**[root@localhost nginx-1.14.0]# make**

**[root@localhost nginx-1.14.0]# make install**

**案例3源码安装python3**

**[root@localhost ~]# yum -y install gcc zlib zlib-devel openssl openssl-devel python-devel mysql-devel**

**#安装依赖包**

**[root@localhost ~]# cd /usr/local/src/**

**[root@localhost src]# wget https://www.python.org/ftp/python/3.6.5/Python-3.6.5.tgz**

**[root@localhost src]# tar -zxvf Python-3.6.5.tgz**

**[root@localhost src]# cd Python-3.6.5/**

**[root@localhost Python-3.6.5]# ./configure --prefix=/usr/local/python3.6 --with-ssl**

**[root@localhost Python-3.6.5]make && make install**

**[root@localhost ~]# vim /etc/profile.d/custom.sh**

**#!/bin/bash**

**PATH="$PATH":/usr/local/python3.6/bin**

**#将bin目录加入PATH环境变量中**

**[root@localhost python3.6]# chmod 755 /etc/profile.d/custom.sh**

**[root@localhost python3.6]# source /etc/profile.d/custom.sh**

#### 源码包的卸载

**不需要卸载命令,直接删除安装目录即可。不会遗留任何垃圾文件。**

## 脚本安装包

**此外还有脚本安装包，所谓的一键安装包,实际上还是安装的源码包与RPM包,只是把安装过程写成了脚本,便于初学者安装**

**优点：简单、快速、方便**

**缺点：**

**不能定义安装软件的版本**

**不能定义所需要的软件功能**

**源码包的优势丧失**

**参考lnmp.org**

# Shell编程

**shell编程主要用来简化服务器管理操作，更多的考虑程序功能实现，而不是效率。**

**其语法结构和绝大多数语言有所不同，本课程会为小伙伴们带来很多实用的小脚本**

**shell编程与我们的python、Java等有什么区别？**

* **Python与Java主要实现系统功能**
* **Shell简化服务器管理操作**

**知乎上有一个讨论: 会一门脚本语言，还有必要学习 bash 吗？<https://www.zhihu.com/question/21747474>**

**有兴趣的同学可以看看拓展一下自己的视野**

## Shell编程基础

### Shell概述

#### Shell是什么？

* **Shell是一个命令行解释器它为用户提供了一个向 Linux内核发送请求以便运行程序的界面系统级程序,用户可以用 Shell来启动、挂起、停止甚至是编写一些程序。**
* **Shell还是一个功能相当强大的编程语言,易编写,易调试灵活性较强。Shell是解释执行的脚本语言,在Shell中可以直接调用 Linux系统命令。**

shell 应用程序

内核

（硬件）

用户

#### Shell的分类

* **Bourne shell从1979起Unix就开始使用 Bourne shell,Bourne shell的主文件名为sh。**
* **C Shell: C Shell主要在BSD版的Unix系统中使用,其语法和C语言相类似而得名**
* **Shell的两种主要语法类型有 Bourne和C,这两种语法彼此不兼容。 Bourne家族主要包括sh、ksh、Bash、psh、zsh; C家族主要包括:csh、tcsh**
* **Bash:Bash与sh兼容，现在使用的 Linux就是使用Bash作为用户的基本Shell**

#### Linux支持的Shell

**查看当前shell：**

**[root@localhost ~]# echo $SHELL**

**/bin/bash**

**查看支持的shell:**

**[root@localhost ~]# cat /etc/shells**

**切换shell:**

**[root@localhost ~]# sh**

**sh-4.2# bash**

**[root@localhost ~]# exit**

**exit**

**sh-4.2#**

### 脚本执行方式

#### echo输出命令

**如**

**[root@localhost ~]# echo "gavin he ben shui geng shuai"**

**gavin he ben shui geng shuai**

**echo [选项] [输出内容]**

**选项**

**-e:支持反斜线控制的字符转换**

|  |  |
| --- | --- |
| **控制字符** | **作用** |
| **a** | **输出警告音** |
| **b** | **退格键,也就是向左删除键** |
| **n** | **换行符** |
| **r** | **回车键** |
| **t** | **制表符,也就是Tab键** |
| **v** | **垂直制表符** |
| **\0nnn** | **按照八进制ASC码表输出字符。其中0为数字零,nn是三位八进制数** |
| **\xhh** | **按照十六进制ASC码表输出字符。其中hh是两位十六进制数** |

**[root@localhost ~]# echo -e "hell\bo"**

**#删除左侧字符**

**[root@localhost -]# echo -e "h\te\tI\nl\to"**

**#制表符与换行符**

**[root@localhost ~]# echo -e "\x68\t\x65\t\x6c\n\x6c\t\x6ft\t"**

**#按照十六进制ASCII也同样可以输出**

**#Assii可显示字符可直接通过百度搜索ascii码查到**

**[root@localhost ~]# echo -e "\e[1;31m嫁人就要嫁凤姐\e[0m"**

**#输出颜色**

**#\e[1;31m 开启颜色显示**

**#\e[0m 取消颜色显示**

**#30m=第色,31m=红色,32m=绿色,33m=黄色**

**#34m=蓝色,35m=洋红,36m=青色,37m=白色**

#### 第一个脚本

**[root@localhost ~]# vim hello.sh**

**#!/bin/bash**

**#The first program**

**echo -e "\e[1;34m天上掉下个林妹妹 \e[0m"**

**[root@localhost ~]# bash hello.sh**

**天上掉下个林妹妹**

**[root@localhost ~]# chmod 755 hello.sh**

**[root@localhost ~]# ./hello.sh**

**天上掉下个林妹妹**

**[root@localhost ~]# /root/hello.sh**

**天上掉下个林妹妹**

#### 脚本执行

**赋予执行权限,直接运行**

* **chmod 755 hello.sh**
* **./hello.sh**

**通过Bash调用执行脚本**

* **bash hello.sh**

### Bash的基本功能

#### 命令别名与快捷键

##### 什么是别名?

**命令别名==人的小名**

##### 查看与设定别名

**[root@localhost ~]# alias**

**#查看系统中所有的命令别名**

**[root@localhost ~]# alias别名= ‘原命令’**

**#设定命令别名**

**如:**

**[root@localhost ~]# alias lsnc='ls --color=never'**

**[root@localhost ~]# lsnc /**

##### 别名永久生效与删除别名

**vi ~/.bashrc**

**#写入环境变量配置文件**

**unalias别名**

**#删除别名**

**source ~/.bashrc**

**#让配置马上生效**

##### 命令生效顺序

* **第一顺位执行用绝对路径或相对路径执行的命令。**
* **第二顺位执行别名。**
* **第三顺位执行Bash的内部命令。**
* **第四顺位执行按照$PATH环境变量定义的目录查找顺序找到的第一个命令**

**[root@localhost ~]# echo $PATH**

**那么什么是内部命令和外部命令？**

##### 内部命令和外部命令

**内部命令：实际上是shell程序的一部分，其中包含的是一些比较简单的linux系统命令，这些命令由shell程序识别并在shell程序内部完成运行，通常在linux系统加载运行时shell就被加载并驻留在系统内存中。内部命令是写在bashy源码里面的，其执行速度比外部命令快，因为解析内部命令shell不需要创建子进程。比如：exit，history，cd，echo等**

**外部命令：是linux系统中的实用程序部分，因为实用程序的功能通常都比较强大，所以其包含的程序量也会很大，在系统加载时并不随系统一起被加载到内存中，而是在需要时才将其调用内存。通常外部命令的实体并不包含在shell中，但是其命令执行过程是由shell程序控制的。shell程序管理外部命令执行的路径查找、加载存放，并控制命令的执行。外部命令是在bash之外额外安装的，可通过“echo $PATH”命令查看外部命令的存储路径，比如：ls、vi等。**

**type命令可以分辨内部命令与外部命令**

**例如：**

**[root@localhost ~]# type cd**

**cd 是 shell 内嵌**

**[root@localhost ~]# type -a ls**

**ls 是 `ls --color=auto' 的别名**

**ls 是 /usr/bin/ls**

##### 常用快捷键

**ctrl+ c 强制终止当前命令**

**ctrl+l 清屏**

**ctrl+a 光标移动到命令行首**

**ctrl+e 光标移动到命令行尾**

**ctrl+u 从光标所在位置删除到行首**

**ctrl+z 把命令放入后台 (后面工作管理的时候学习，此处先不多做说明)**

**ctrlt+r 在历史命令中搜索**

#### 历史命令history

##### 历史命令

**history [选项] [历史命令保存文件]**

**选项:**

**-c:清空历史命令**

**-w:把缓存中的历史命令写入历史命令保存文件~/.bash\_history**

**建议不要随便清除历史命令！**

**历史命令默认会保存1000条可以在环境变量配置文件/etc/profile中进行修改**

**[root@localhost ~]# vi /etc/profile**

**HISTSIZE=1000**

##### 历史命令的调用

* **使用上、下箭头调用以前的历史命令**
* **使用“!n”重复执行第n条历史命令**
* **使用“!!″重复执行上一条命令**
* **使用“!字串”重复执行最后一条以该字串开头的命令**

##### 命令与文件补全

**在Bash中,命令与文件补全是非常方便与常用的功能,我们只要在输入命令或文件时,按“Tab”键就会自动进行补全**

#### 输出重定向

##### 标准输入输出

|  |  |  |  |
| --- | --- | --- | --- |
| **设备** | **设备文件名** | **文件描述符** | **类型** |
| **键盘** | /dev/stdin | 0 | 标准输入 |
| **显示器** | /dev/stout | 1 | 标准正确输出 |
| **显示器** | /dev/sdterr | 2 | 标准错误输出 |

##### 输出重定向

|  |  |  |
| --- | --- | --- |
| **类型** | **符号** | **作用** |
| 标准正确输出重定向 | 命令 **>**文件 | 以**覆盖**的方式,把命令的正确输出输出到指定的文件或设备当中。 |
| 命令 **>>**文件 | 以**追加**的方式,把命令的正确输出输出到指定的文件或设备当中。 |
| 标准错误输出重定向 | 错误命令 2>文件 | 以覆盖的方式,把命令的错误输出输出到指定的文件或设备当中 |
| 错误命令 2>>文件 | 以追加的方式,把命令的错误输出输出到指定的文件或设备当中。 |
|  | | |
| **正确输出和错误输出同时保存** | 命令 >文件2>&1 | 以覆盖的方式,把正确输出和错误输出都保存到同一个文件当中。 |
| **命令 >> 文件 2>&1** | **以追加的方式,把正确输出和错误输出都保存到同一个文件当中。** |
| 命令 &> 文件 | 以覆盖的方式,把正确输出和错误  保存输出都保存到同一个文件当中。 |
| **命令 &>> 文件** | **以追加的方式,把正确输出和错误输出都保存到同一个文件当中。** |
| **命令 >>文件1 2>>文件2** | **把正确的输出追加到文件1中,把错误的输出追加到文件2中。** |

##### 输入重定向

**[root@ localhost ~]# wc [选项] [文件名]**

**选项:**

**-c统计字符数**

**-w统计单词数**

**-l统计行数**

**命令<文件把文件作为命令的输入**

**命令<<标识符**

**…**

**标识符把标识符之间内容作为命令的输入**

#### 管道符

##### 多命令顺序执行

|  |  |  |
| --- | --- | --- |
| **多命令执行符** | **格式** | **作用** |
| **;** | 命令1;命令2 | 多个命令顺序执行,命令之间没有任何逻辑联系 |
| **&&** | 命令1&&命令2 | 逻辑与  当命令1正确执行,则命令2才会执行  当命令1执行不正确,则命令2不会执行 |
| **||** | 命令1||命令2 | 逻辑或  当命令1执行不正确,则命令2才会执行  当命令1正确执行,则命令2不会执行 |

**例子:**

**[root@localhost ~]# pwd;touch abc;ls;date**

**[root@localhost ~]# ls anaconda-ks.cfg && echo yes**

**[root@localhost ~]# ls anaconda-ks.cfg1 && echo no**

**[root@localhost ~]# ls anaconda-ks.cfg && echo yes||echo no**

##### 管道符

**[root@ localhost~]# 命令1 | 命令2**

**#命令1的正确输出作为命令2的操作对象**

**例子:**

**[root@localhost ~]# ll -a /etc/ | more**

**[root@localhost ~]# netstat -an | grep "LISTEN"**

### 通配符

**主要用来匹配文件目录，比如删除特定规则的文件等**

|  |  |
| --- | --- |
| **通配符** | **作用** |
| **?** | 匹配一个任意字符 |
| \* | 匹配0个或任意多个任意字符,也就是可以匹配任何内容 |
| **[]** | 匹配中括号中任意一个字符。例如:abc]代表一定匹配一个字符,或者是a,或者是b,或者是c。 |
| **[-]** | 匹配中括号中任意一个字符,-代表一个范匹配中括号中任意一个字符,-代表一个范围。例如:[a-z]代表匹配一个小写字母 |
| **[^]** | 逻辑非,表示匹配不是中括号内的一个字符。例如:[^0-9]代表匹配一个不  是数字的字符。 |

### Bash中的其他特殊符号

|  |  |
| --- | --- |
| **符号** | **作用** |
| **‘’** | 单引号。在单引号中所有的**特殊符号**,如“S”和“”(反引号)都**没有特殊含义。** |
| **“”** | 双引号。在双引号中特殊符号都没有特殊含义,但是“$”、“`”和“\”是例外,拥有“调用变量的值”、“引用命令”和“转义符”的特殊含义。 |
| **``** | 反引号。反引号括起来的内容是系统命令,在Bash中会先执行它。和$()作用一样,不过推荐使用$(),因为反引号非常容易看错，建议少用。 |
| **$()** | 和反引号作用一样,用来引用系统命令。 |
| **#** | 在Shell脚本中,#开头的行代表注释。 |
| **$** | 用于调用变量的值,如需要调用变量name的值时,需要用 $name的方式得到变量的值 |
| **\** | 转义符,跟在\之后的特殊符号将失去特殊含义,变为普通字符。如\$将输出“$”符号,而不当做变量引用。 |

**例如:**

**[root@localhost ~]# name=ten**

**[root@localhost ~]# echo '$name'**

**[root@localhost ~]# echo "$name"**

**[root@localhost ~]# echo `date`**

**[root@localhost ~]# echo "$(date)"**

## Bash变量

### 什么是变量与变量分类

**变量是计算机内存的单元,其中存放的值可以改变变量让你能够把程序中准备使用的每一段数据都赋给个简短、易于记忆的名字,因此它们十分有用**

#### 变量命名规则

* **变量名必须以字母或下划线打头,名字中间只能由字母、数字和下划线组成**
* **变量名的长度不得超过255个字符**
* **变量名在有效的范围内必须是唯一的**
* **在Bash中,变量的默认类型都是字符串型**

**在任何系统中,目录名、文件名、变量名都要有含义!！！**

#### 变量按照存储数据分类

* **字符串型**
* **整型**
* **浮点型**
* **日期**

#### 变量的分类

* **用户自定义变量。变量自定义的**
* **环境变量∶这种变量中主要保存的是和系统操作环境相关的数据。变量可以自定义,但是对系统生效的环境变量名和变量作用是固定的**
* **位置参数变量:这种变量主要是用来向脚本当中传递参数或数据的,变量名不能自定义,变量作用是固定的（是预定义变量中的一种，所以实际上只有三类）**
* **预定义变量:是Bash中已经定义好的变量,变量名不能自定义,变量作用也是固定的**

### 用户自定义变量

##### 定义变量

**变量名=变量值(等号两侧不能有空格)**

**例如**

**x=5**

**name=”jie cao”**

##### 变量调用

**$变量名**

**例如**

**echo $x**

**echo $name**

##### 变量叠加

**x=123**

**x="$x"456**

**x=${x}789 #不建议用**

##### 变量查看

**set**

**#查询系统下所有变量**

**选项:**

**-u如果设定些选项,调用未声明变量时会报错(默认无任何提示)**

##### 变量删除

**unset变量名**

### 环境变量

#### 环境变量

**环境变量与用户自定义变量的区别?**

* **环境变量是全局变量,用户自定义变量是局部变量，用户自定义变量只在当前的Shell中生效,环境变量在当前Shell和这个Shell的所有子Shell中生效**
* **变量可以自定义,对系统生效的环境变量名和变量作用是固定的**

#### 设置环境变量

**export 变量名=变量值**

**或**

**变量名=变量值**

**export 变量名**

#### 查看环境变量

**set**

**#查看所有变量**

**env**

**#查看环境变量**

#### 删除环境变量

**unset 变量名**

#### 常用环境变量

**HOSTNAME: 主机名**

**SHELL: 当前的shell**

**TERM: 终端环境**

**HISTSIZE: 历史命令亲数**

**SSH CLIENT: 当前架作环境是用ss连的,这里记录客户端ip**

**SSH TTY: ssh连的终端的pts/1**

**USER: 当前置录的用户**

#### PATH环境变量

**PATH变量:系统查找命令的路径**

**echo $PATH**

**#查看PATH环境变量**

**PATH="$PATH":/root/**

**#增加PATH变量的值**

#### PS1环境变量

**PS1变量:命令提示符设置**

* **\d:显示日期,格式为“星期 月 日**
* **\H:显示完整的主机名。如默认主机名“ localhost. localdomain**
* **\t:显示24小时制时间,格式为“HH:MM:Ss"**
* **\A:显示24小时制时间,格式为“HH:MM**
* **\u:显示当前用户名**
* **\w:显示当前所在目录的完整名称**
* **\W:显示当前所在目录的最后一个目录**
* **\$:提示符。如果是root用户会显示提示符为“#”,如果是普通用户会显示提示符为”$”**

**查看:**

**[root@localhost ~]# echo $PS1**

**[\u@\h \W]\$**

#### 当前语系查询

**locale**

**#查询当前系统语系**

* **LANG:定义系统主语系的变量**

#### 语系变量LANG

**echo $LANG**

**#查看系统当前语系**

**locale -a | more**

**#查看Linux支持的所有语系**

#### Linux中文支持

**前提条件,正确安装的中文字体和中文语系**

* **如果有图形界面,可以正确支持中文显示**
* **如果使用第三方远程工具,只要语系设定正确,可以支持中文显**
* **如果使用纯字符界面,必须使用第三方插件(如 zhcon等)**

### 位置参数变量

**位置参数变量也是预定义变量的一种**

|  |  |
| --- | --- |
| **位置参数变量** | **作用** |
| $n | n为数字,$0代表命令本身,$1-$9代表第一到第九个参数,十以上的参数需要用大括号包含,如${10} |
| $\* | 这个变量代表命令行中所有的参数,$\*把所有的参数看成一个整体 |
| $@ | 这个变量也代表命令行中所有的参数,不过$@把每个参数区分对待 |
| $# | 这个变量代表命令行中所有参数的个数 |

**例子1:**

**#!/bin/bash**

**num1=$1**

**num2=$2**

**sum1=$(($num1 + $num2))**

**echo $sum1**

**例子2:**

**#!/bin/bash**

**echo "A total of $# parameters"**

**#使用$#代表所有参数的个数**

**echo "The parameters is: $\*"**

**#使用$\*代表所有的参数**

**echo "The parameters is: $@"**

**#使用$@也代表所有参数**

**例子3:$\*与$@的区别**

**#!/bin/bash**

**for i in "$\*"**

**#$中的所有参数看成是一个整体,所以这个循环只会循环一次**

**do**

**echo "The parameters is: $i"**

**done**

**for y in "$@"**

**#$@中的每个参数部看成是抛立的,所以“$@”中有几个参数,就会循环几次**

**do**

**echo "Parameter:$y"**

**done**

### 预定义变量

#### 预定义变量

|  |  |
| --- | --- |
| **预定义变量** | **作用** |
| $? | **最后一次执行的命令的返回状态。如果这个变量的值为0,证**  **明上一个命令正确执行;如果这个变量的值为非0(具体是哪**  **个数,由命令自己来决定),则证明上一个命令执行不正确了。** |
| $$ | **当前进程的进程号(PID)** |
| $! | **后台运行的最后一个进程的进程号(PID)** |

**案例**

**[root@localhost ~]# ls**

**anaconda-ks.cfg a.sh a.txt b.sh hello.sh initial-setup-ks.cfg**

**[root@localhost ~]# echo $?**

**0**

**[root@localhost ~]# ls /abc**

**ls: 无法访问/abc: 没有那个文件或目录**

**[root@localhost ~]# echo $?**

**2**

**[root@localhost ~]# vim b.sh**

**#!/bin/bash**

**echo "current pid: $$"**

**find / -name hello.sh &**

**echo "backgroup pid: $!"**

#### 接收键盘输入

**read [选项] [变量名]**

**选项:**

**-p:“提示信息”:在等待read输入时,输出提示信息**

**-t:秒数:read命令会一直等待用户输入,使用此选项可以指定等待时间**

**-n:字符数:read命令只接受指定的字符数,就会执行**

**-s:隐藏输入的数据,适用于机密信息的输入**

**案例:**

**#!/bin/bash**

**read -p "please input your name: " -t 30 name**

**echo $name**

## Shell运算符

**shell变量的缺点?**

**弱类型**

**默认字符串型**

### declare命令

#### declare声明变量类型

**[root@ localhost~]# declare [+/-] [选项] 变量名**

**选项:**

**-:给变量设定类型属性**

**+:取消变量的类型属性**

**-a:将变量声明为数组型**

**-i:将变量声明为整数型( integer)**

**-x:将变量声明为环境变量**

**-r:将变量声明为只读变量**

**-p:显示指定变量的被声明的类型**

#### 把变量声明为数值型

**[root@localhost ~]# aa=11**

**[root@localhost ~]# bb=22**

**#给变量a和bb赋值**

**[root@localhost ~]# declare -i cc=$aa+$bb**

**#声明变量cc的类型是整数型,它的值是aa和bb的和**

#### 声明数组变量

**#定义数组**

**[root@localhost ~]# movie[0]=cang**

**[root@localhost ~]# movie[1]=bo**

**[root@localhost ~]# declare -a movie[2]=duo**

**#查看数组**

**[root@localhost ~]# echo ${movie}**

**#不加下标默认为第1个即movie[0]**

**[root@localhost ~]# echo ${movie[2]}**

**[root@localhost ~]# echo ${movie[\*]}**

#### 声明环境变量

**declare -x test=123**

**#和export作用相似,但其实是调用了declare命令的作用**

#### 声明变量只读属性

**[root@localhost ~]# declare -r test**

**#给test只读属性,但是请注意只读属性会让变量不能修改,不能删除,甚至不能取消只读属性**

#### 查询变量的属性

**declare -p**

**#查询所有变量的属性**

**declare -p 变量名**

**#查询指定变量的属性**

### 数值运算的方法

#### 方法1: 数值运算

**[root@localhost ~]# aa=11**

**[root@localhost ~]# bb=22**

**#给变量a和bb赋值**

**[root@localhost ~]# declare -i cc=$aa+$bb**

#### 方法2:expr或et数值运算工具

**[root@localhost ~]# aa=11**

**[root@localhost ~]# bb=22**

**#给变量aa和变量bb赋值**

**[root@localhost ~]# dd=$(expr $aa + $bb)**

**#dd的是aa和bb的和。**

#### 方法3:“$((运算式))”或“$[运算式]”

**强烈推荐方法 $((运算式))**

**[root@localhost ~]# aa=11**

**[root@localhost ~]# bb=22**

**[root@localhost ~]# cc=$(($aa+$bb))**

**[root@localhost ~]# dd=$[$aa+$bb]**

#### 运算符

**其实shell编程较少需要做数值运算，更多的是系统管理！**

|  |  |  |
| --- | --- | --- |
| **优先级** | **运算符** | **说明** |
| 13 | -,+ | 单目负、单目正 |
| 12 | !,~ | 逻辑非、按位取反或补码 |
| 11 | \*,/,% | 乘、除、取模 |
| 10 | +,- | 加、减 |
| 9 | <<, >> | 按位左移、按位右移 |
| 8 | <=, >=, <,> | 小于或等于、大于或等于、小于、大于 |
| 7 | ==, != | 等于、不等于 |
| 6 | & | 按位与 |
| 5 | ^ | 按位异或 |
| 4 | | | 按位或 |
| 3 | && | 逻辑与 |
| 2 | || | 逻辑或 |
| 1 | =, +=,\*=,/=,%=,&=,^=,|=,<<=,>>= | 赋值、运算且赋值 |

**例如：**

**[root@localhost ~]# aa=$(((11+3)\*3/2))**

**#虽然乘和除的优先级高于加,但是通过小括号可以调整运算优先级**

**[root@localhost ~]# bb=$((14%3))**

**#14不能被3整除,余数是2**

**[root@localhost ~]# cc=$((1&&0))**

**#逻辑与运算只有想与的两边部是1,与的结果才是1,否则与的结果是0**

### 变量测试

**只有shell有，只需要做了解，格式复杂，语法简单**

**变量测试在脚本优化时使用**

|  |  |  |  |
| --- | --- | --- | --- |
| **变量置换方式** | **变量y没有设置** | **变量y为空值** | **变量y设置值** |
| x=${y-新值} | x=新值 | x为空 | x=$y |
| x=${y:-新值} | x=新值 | x=新值 | X=$y |
| x=${y+新值} | x为空 | x=新值 | x=新值 |
| x=${y:+新值} | x为空 | x为空 | x=新值 |
| x=${y=新值} | x=新值  y=新值 | x为空  y值不变 | x=$y  y值不变 |
| X=${y:=新值} | x=新值  y=新值 | x=新值  y=新值 | x=$y  y值不变 |
| X=${y?新值} | 新值输出到标准错误输出(就是屏幕) | x为空 | x=$y |
| X=${y:?新值} | 新值输出到标准错误输出 | 新值输出到标准错误输出 | x=$y |

**案例:测试x=${y-新值}**

**[root@localhost ~]# unset y**

**[root@localhost ~]# x=${y-2}**

**[root@localhost ~]# echo $x**

**2**

**[root@localhost ~]# y=""**

**[root@localhost ~]# x=${y-2}**

**[root@localhost ~]# echo $x**

**[root@localhost ~]# y="aa"**

**[root@localhost ~]# x=${y-2}**

**[root@localhost ~]# echo $x**

**aa**

### $()、$(())与${}

* **$( )与` `即反引号在操作上，这两者都是达到相应的效果，但是建议使用$( )，**

**因为` `很容易与’ ’搞混乱，尤其对初学者来说，而$( )比较直观。**

**$( )的弊端是，并不是所有的类unix系统都支持这种方式，但反引号是肯定支持的**

**例如**

**[root@localhost ~]# echo today is $(date "+%Y-%m-%d")**

**today is 2018-08-31**

**[root@localhost ~]# echo today is `date "+%Y-%m-%d"`**

**today is 2018-08-31**

* **${ }为变量替换，一般情况下，$var与${var}是没有区别的，但是用${ }会比较精确的界定变量名称的范围**

**例如**

**[root@localhost ~]# A=B**

**[root@localhost ~]# echo $AB**

**[root@localhost ~]# echo ${A}B**

**BB**

* **$(( ))与用来进行运算**

**例如**

**[root@localhost ~]# aa=11**

**[root@localhost ~]# bb=22**

**[root@localhost ~]# cc=$(($aa+$bb))**

**[root@localhost ~]# echo $cc**

**33**

## 环境变量配置文件

### source命令

**[root@localhost ~]# source 配置文件**

**[root@localhost ~]# . 配置文件**

**修改配置文件后,必须注销重新登录才能生效,使用source命令可以不用重新登录**

**例如**

**[root@localhost ~]# . .bashrc**

**[root@localhost ~]# source .bashrc**

### 环境变量配置文件简介

**环境变量配置文件主要是定义对系统操作环境生效的默认环境变量。**

**PATH、HISTSIZE、PS1、HOSTNAME等环境变量需要写入对应的环境变量配置文件，**

### 环境变量配置文件的功能

#### 环境变量配置文件（主要的）

* **/etc/profile**
* **/etc/profile. d/\*.sh**
* **~/.bash\_profile**
* **~/.bashrc**
* **/etc/bashrc**

#### 环境配置过程

~/.bashrc

 /etc/profile

 /etc/bashrc

~/.bash\_profile

 命令提示符

/etc/profile.d/\*.sh

/etc/profile.d/lang.sh

#### /etc/profile的作用

**USER变量**

**LOGNAME变量:**

**MAIL变量:**

**PATH变量:**

**HOSTNAME变量:**

**HISTSIZE变量**

**umask**

**调用/etc/profile. d/\*.sh文件**

#### ~/bash\_profile的作用

**调用了~/.bashrc文件**

**在PATH变量后面加入了”:$HOME/bin”这个目录**

#### /etc/bashrc的作用

* **PS1变量**
* **umask**
* **PATH变量**
* **调用/etc/profile. d/\*.sh文件**

### 其他配置文件

#### 注销时生效的环境变量配置文件

**~/.bash\_logout**

#### Shell登录信息

**本地终端欢迎信息:/etc/issue**

|  |  |
| --- | --- |
| **转义符** | **作用** |
| \d | **显示当前系统日期** |
| \S | **显示操作系统名称** |
| \l | **显示登录的终端号,这个比较常用。** |
| \m | **显示硬件体系结构,如i386、i686等** |
| \n | **显示主机名** |
| \o | **显示域名** |
| \r | **显示内核版本** |
| \t | **显示当前系统时间** |
| \u | **显示当前登录用户的序列号** |

**远程终端欢迎信息:/etc/issue.net**

* **转义符在/etc/issue.net文件中不能使用**
* **是否显示此欢迎信息,由ssh的配置文件/etc/ssh/sshd\_config决定,加入”Banner /etc/ issue. net"行才能显示(记得重启SSH服务)**

**登陆后欢迎信息:/etc/motd**

**不管是本地登录,还是远程登录,都可以显示此欢迎信息**

## 正则表达式

### 正则表达式是什么

**正则表达式是用于描述字符排列和匹配模式的一种语法规则。它主要用于宇符串的模式分割、匹配、查找及替换操作。**

### 正则表达式与通配符

* **正则表达式用来在文件中匹配符合条件的字符串,正则是包含匹配。grep、awk、sed等命令可以支持正则表达式。**
* **通配符用来匹配符合条件的文件名,通配符是完全匹配。ls、find、cp这些命令不支持正则表达式,所以只能使用shell自己的通配符来进行匹配了**

**通配符**

**\*匹配任意内容**

**?匹配任意一个内容**

**[]匹配中括号中的一个字符**

### 基础正则表达式

#### 基础正则表达式

|  |  |
| --- | --- |
| **元字符** | **作用** |
| **\*** | **前一个字符匹配0次或任意多次。** |
| **.** | **匹配除了换行符外任意一个字符。** |
| **^** | **匹配行首。例如:^hello会匹配以hello开头的行。** |
| **$** | **匹配行尾。例如:hello&会匹配以hello结尾的行** |
| **[]** | **匹配中括号中指定的任意一个字符,只匹配一个字符。例如:[aoeiu]匹配任意一个元音字母,[0-9]匹配任意一位数字,[a-z][0-9] 匹配小写字和一位数字构成的两位字符。** |
| **[^]** | **匹配除中括号的字符以外的任意一个字符。例如:[^0-9]匹配任意一位非数字字符,[^a-z]表示任意一位非小写字母。** |
| **\** | **转义符。用于取消讲特殊符号的含义取消。** |
| **\{n\}** | **表示其前面的字符恰好出现n次。例如:[0-9]\{4\}匹配4位数字,[1][3-8]0-9]\{9\}匹配手机号码。** |
| **\{n,\}** | **表示其前面的字符出现不小于n次。例如:[0-9]\{2,\}表示两位及以上的数字。** |
| **\{n,m\}** | **表示其前面的字符至少出现n次,最多出现m次。例如:[a-z]\{6,8\}匹配6到8位的小写字母** |

**?与()是拓展正则，shell中不支持**

#### “\*”前一个字符匹配0次,或任意多次

* **“a\*”**

**#匹配所有内容,包括空白行**

* **“aa\*”**

**#匹配至少包含有一个a的行**

* **“aaa\*”**

**#匹配最少包含两个连续a的字符串**

* **“aaaaa\*”**

**#则会匹配最少包含四个个连续a的字符串**

#### “.”匹配除了换行符外任意一个字符

* **"s..d"**

**#“s..d"会匹配在和d这两个字母之一定有两个字的单词**

* **“s.\*d”**

**匹配在s和d字母之同有任意学符**

* **“.\*”**

**匹配所有内容**

#### “^”匹配行首,“$"匹配行尾

* **"^M”**

**#匹配以大写”M"开头的行**

* **"n$"**

**#匹配一小写"n”结尾的行**

* **“^$”**

**#会匹配空白行**

#### “[]”匹配中括号中指定的任意一个字符,只匹配一个字符

* **"s[ao]i"**

**#匹配s和i字字母中,要不是a、要不是o**

* **"[0-9]”**

**#匹配任意一个数字**

* **“^[a-z]”**

**#匹配用小写字受开义的行**

#### “[^]”匹配除中括号的字符以外的任意一个字符

* **“[^a-z]”**

**匹配不用小写字母开头的行**

* **“[^a-zA-Z]”**

**匹配不以字母受开头的行**

#### “\”转义符

* **"\.$”**

**#匹配使用”.”结尾的行**

#### “\{n\}”表示其前面的字符怡好出现n次

* **"a\{3\}"**

**#匹配a字母连续出现三次的字符,字母a至少重复3次**

* **“[0-9]\{3\}”**

**#匹配包含连续的三个数字的字符**

#### “\{n,\}”表示其前面的字符出现不小于n次

* **“^[0-9]\{3,\}[a-z]”**

**#匹配最少用连续三个数字开头的行**

#### “\{n,m\}”匹配其前面的字符至少出现n次,最多出现m次

* **"sa\{1,3\}i"**

**#匹配在字母s和字母i之间有最少一个a,最多三个a**

#### 几个例子

* **[0-9]\{4\}-[0-9]\{2\}-[0-9]\{2\}**

**#匹配日期格式 YYYY-MM-DD**

* **[0-9]\{1,3\}\.**

**#匹配IP地址**

### 字符串截取命令

**思考为什么需要进行字符串的截取？**

**当我们需要删除所有普通的用户，第一步需要去获取所有的普通用户的用户名，那么怎么去获取呢？**

#### cut字段提取命令

**[root@ localhost~]#cut [选项] 文件名**

**选项**

**-f列号:提取第几列**

**-d分隔符:按照指定分隔符分割列**

**[root@localhost ~]# vi student. txt**

**ID Name gender Mark**

**1 furon0 F 85**

**2 fengjie F 60**

**3 cang F 70**

**[root@localhost ~]# cut -f 2 student.txt**

**[root@localhost ~]# cut -f 2,4 student.txt**

**[root@localhost ~]# cut -f 2-4 student.txt**

**[root@localhost ~]# grep "/bin/bash" /etc/passwd | grep -v "root" | cut -f 1 -d ":"**

**如果我们想知道磁盘的占有情况，发现如果磁盘使用率超过80%则发短信提醒管理员，应该怎样实现？**

#### printf命令

**printf格式化输出命令，实际并不是字符串截取命令，但要在awk中用到**

**printf '输出类型输出格式’ 输出内容**

**输出类型:**

* **%-ns:输出字符串。n是数字指代输出几个字符**
* **%-ni:输出整数。n是数字指代输出几个字符**
* **%-m.nf:输出浮点数。m和n是数字,指代输出的整数位数和小数位数**

**如%8.2f代表共输出占8位,其中2位是小数。123456.78**

**输出格式:**

* **\a:输出警告声音**
* **\b:输出退格键,也就是 Backspace键**
* **\f:清除屏幕**
* **\n:换行**
* **\r:回车,也就是 Enter键**
* **\t:水平输出制表符,也就是Tab键**
* **\v:垂直输出制表符,也就是Tab键**

**例如：**

**[root@localhost ~]# printf 1 2 3 4 5**

**[root@localhost ~]# printf %s 1 2 3 4 5**

**[root@localhost ~]# printf %s %s %s 1 2 3 4 5 6**

**[root@localhost ~]# printf '%s %s %s' 1 2 3 4 5 6**

**[root@localhost ~]# printf '%s %s %s\n' 1 2 3 4 5 6**

**[root@localhost ~]# printf '%s' $(cat student.txt)**

**#不调整输出格式**

**[root@localhost ~]# printf '%s\t%s\t%s\t%s\n' $(cat student.txt)**

**#调整格式输出**

**[root@localhost ~]# printf "%-5s %-10s %-4s\n" NO Name Mark**

**[root@localhost ~]# printf "%-5s %-10s %-4.2f\n" 01 Tom 90.3456**

**[root@localhost ~]# printf "%-5s %-10s %-4.2f\n" 02 Jack 89.2345**

**[root@localhost ~]# printf "%-5s %-10s %-4.2f\n" 03 Jeff 98.4323**

**在awk命令的输出中支持print和 printf命令**

* **print: print会在每个输出之后自动加入一个换行符(Linux默认没有 print命令)**
* **printf: printf是标准格式输出命令,并不会自动加入换行符,如果需要换行,需要手工加入换行符**

#### awk命令

**#awk ‘条件1{动作1} 条件2{动作2}…’ 文件名**

**条件( Pattern)**

* **一般使用关系表达式作为条件**
* **x>10判断变量x是否大于10**
* **x>=10大于等于**
* **x<=10小于等于**

**动作( Action)**

* **格式化输出**
* **流程控制语句**

**如：**

**[root@localhost ~]# awk '{printf $2 "\t" $4 "\n"}' student.txt**

**[root@localhost ~]# awk '{print $2 "\t" $4}' student.txt**

**[root@localhost ~]# df -h | awk '{print $5}'**

**[root@localhost ~]# df -h | grep "/$" | awk '{print $5}' | cut -f 1 -d "%"**

**BEGIN**

**[root@localhost ~]# awk 'BEGIN{print "begin"}{print $2 "\t" $4}' student.txt**

**END**

**[root@localhost ~]# awk 'END{print "end"}{print $2 "\t" $4}' student.txt**

**FS内置变量**

**[root@localhost ~]# cat /etc/passwd | grep '/bin/bash' | awk 'BEGIN{FS=":"}{print $1 "\t" $3}'**

**关系运算符**

**[root@localhost ~]# cat student.txt | grep -v Name | awk '$4>=70{print $2}'**

#### sed命令

**sed命令**

**sed是一种几乎包括在所有UNIX平台(包括 Linux)的轻量级流编辑器。sed主要是用来将数据进行选取、替换、删除、新增的命令**

**sed [选项] ‘[动作]’ 文件名**

**选项:**

* **-n:一般sed命令会把所有数据都输出到屏幕,如果加入此选择则只会把经过sed命令处理的行输出到屏幕。**
* **-e:允许对输入数据应用多条sed命令编辑**
* **-i:用sed的修改结果直接修改读取数据的文件,而不是由屏幕输出**

**动作**

* **a:追加,在当前行后添加一行或多行**
* **c:行替换,用c后面的字符串替换原数据行**
* **i:插入,在当期行前插入一行或多行。**
* **d:删除,删除指定的行**
* **p:打印,输出指定的行。**
* **s:字串替换,用一个字符串替换另外一个字符串。格式为“行范围s/旧字串/新字串/g”(和vim中的替换格式类似)。**

**行数据操作**

**[root@localhost ~]# sed '2p' student.txt**

**[root@localhost ~]# sed -n '2p' student.txt**

**#查看文件的第二行**

**[root@localhost ~]# sed '2d' student.txt**

**[root@localhost ~]# sed '2,4d' student.txt**

**#删除第二行到第四行的数据,但不修改文件本身**

**[root@localhost ~]# sed '2a piaoliang jiushi renxing' student.txt**

**#在第二行后追加**

**[root@localhost ~]# sed '2i piaoliang jiushi renxing' student.txt**

**#在第二行前插入**

**[root@localhost ~]# sed '4c piaoliang jiushi renxing' student.txt**

**#数据替换**

**字符串替换**

**sed 's/旧字串/新字串/g’ 文件名**

* **[root@localhost opt]# sed '4s/70/100/g' student.txt**

**#在第四行中,把70换成100**

* **[root@localhost opt]# sed -i '3s/70/100/g' student.txt**

**#sed操作的数据直接写入文件**

* **[root@localhost opt]# sed 's/furon0/mm/g;s/fengjie/mm2/g' student.txt**

**#两条数据同时替换**

#### 排序命令sort

**sort [选项] 文件名**

**选项**

**-f:忽略大小写**

**-n:以数值型进行排序,默认使用字符串型排序**

**-r:反向排序**

**-t:指定分隔符,默认是分隔符是制表符**

**-k n[,m]:按照指定的字段范围排序。从第n字段开始,m字段结束(默认到行尾)**

**例如：**

**sort /etc/passwd**

**#排序用户信息文件**

**sort -r /etc/passwd**

**#反向排房**

**sort -t ":" -k 3,3 /etc/passwd**

**#指定分隔符是”:”,用第三个字段开头,第个三字段结尾排序,就是只用第个三字段排序**

**sort -n -t ":" -k 3,3 /etc/passwd**

#### 字符统计命令wc

**[root@ localhost ~]# wc [选项] [文件名]**

**选项:**

**-c统计字符数**

**-w统计单词数**

**-l统计行数**

**例如：**

**[root@localhost opt]# wc /etc/passwd**

## 条件判断与流程控制

’

**为什么要进行条件判断和流程控制？**

### 条件判断式

#### 按照文件类型进行判断

|  |  |
| --- | --- |
| **测试选项** | **作用** |
| **-b文件** | 判断该文件是否存在,并且是否为块设备文件(是块设备文件为真) |
| **-c文件** | 判断该文件是否存在,并且是否为字符设备文件(是字符设备文件为真) |
| **-d文件** | 判断该文件是否存在,并且是否为目录文件(是目录为真) |
| **-e文件** | 判断该文件是否存在(存在为真) |
| **-f文件** | 判断该文件是否存在,并且是否为普通文件(是普通文件为真) |
| **-L文件** | 判断该文件是否存在,并且是否为符号链接文件(是符号链接文件为真) |
| **-p文件** | 判断该文件是否存在,并且是否为管道文件(是管道文件为真) |
| **-s文件** | 判断该文件是否存在,并且是否为非空(非空为真) |
| **-S文件** | 判断该文件是否存在,并且是否为套接字文件(是套接字文件为真) |

**两种判断格式**

* **test -e /root/anaconda-ks.cfg**
* **[ -e /root/anaconda-ks.cfg ]**

**推荐使用第二种方式，注意空格不能省**

**[ -d /root ] && echo "yes" || echo "no"**

**#一个判断命令如果正确热行,则的印"yes",否则打印"no**

#### 按照文件权限进行判断

|  |  |
| --- | --- |
| **测试选项** | **作用** |
| **-r文件** | 判断该文件是否存在,并且是否该文件拥有读权限(有读权限为真) |
| **-w文件** | 判断该文件是否存在,并且是否该文件拥有写权限(有写权限为真) |
| **-x文件** | 判断该文件是否存在,并且是否该文件拥有执行权限(有执行权限为真) |
| **-u文件** | 判断该文件是否存在,并且是否该文件拥有SUD权限(有SUID权限为真) |
| **-g文件** | 判断该文件是否存在,并且是否该文件拥有SGID权限(有SGID权限为真) |
| **-k文件** | 判断该文件是否存在,并且是否该文件拥有SBit权限(有SBit权限为真) |

**[ -w student.txt ] && echo "yes" || echo "no"**

**#判断文件是拥有写权限的**

**思考：怎样判断是否同时拥有读和执行的权限？**

#### 两个文件之间进行比较

|  |  |
| --- | --- |
| **测试选项** | **作用** |
| **文件1 -nt 文件2** | 判断文件1的修改时间是否比文件2的新(如果新则为真) |
| **文件1 -ot 文件2** | 判断文件1的修改时间是否比文件2的旧(如果旧则为真) |
| **文件1 -ef 文件2** | 判断文件1是否和文件2的 Inode号一致,可以理解为两个文件是否为同一个文件。这个判断用于判断硬链接是很好的方法 |

**案例**

**[root@localhost ~]# ln /root/student.txt /tmp/**

**#创建一个硬链接**

**[root@localhost ~]# [ /root/student.txt -ef /tmp/student.txt ] && echo yes || echo no**

**yes**

**#用-ef判断结果为真**

#### 两个整数之间比较

|  |  |
| --- | --- |
| **测试选项** | **作用** |
| **整数1 -eq 整数2** | 判断整数1是否和整数2相等(相等为真) |
| **整数1 -ne 整数2** | 判断整数1是否和整数2不相等(不相等位置 |
| **整数1 -gt 整数2** | 判断整数1是否大于整数2(大于为真) |
| **整数1 -lt 整数2** | 判断整数1是否小于整数2(小于位置) |
| **整数1 -ge 整数2** | 判断整数1是否大于等于整数2(大于等于为真) |
| **整数1 -le 整数2** | 判断整数1是否小于等于整数2(小于等于为真) |

**几个单词**

**equal：等于**

**not equal： 不相等**

**greater：大于**

**less：小于**

**than: 比较**

**在进行比较的时候**

**案例**

**[ten@localhost opt]$ [ 23 -ge 22 ] && echo 'yes' || echo 'no'**

**yes**

**#判断23是否大于等于22**

**[ten@localhost opt]$ [ 23 -le 22 ] && echo 'yes' || echo 'no'**

**no**

**#判断23是否小于等于22**

#### 字符串的判断

|  |  |
| --- | --- |
| **测试选项** | **作用** |
| **-z 字符串** | 判断字符串**是否为空**(为空返回真) |
| **-n 字符串** | 判断字符串**是否为存在值**(非空返回真) |
| **字串1 == 字串2** | 判断字符串1是否和字符串2相等(相等返回真) |
| **字串1 != 字串2** | 判断字符串1是否和字符串2不相等(不相等返回真) |

**案例:**

**[root@localhost ~]$ name=fengjie**

**#给name变量赋值**

**[root @localhost ~]$ [ -z "$name" ] && echo "yes" || echo "no"**

**no**

**#判断name变星是否为空,因为不为空,所以返回no**

**[root @localhost ~]$ aa=11**

**[root @localhost ~]$ bb=22**

**#给变量aa和变量bb赋值**

**[root @localhost ~]$ [ $aa == $bb ] && echo "yes" || echo "no"**

**no**

**#判断两个变量是否等,明显不相等,所以返回no**

#### 多重条件判断

|  |  |
| --- | --- |
| **测试选项** | **作用** |
| **判断1 -a 判断2** | 逻辑与,判断1和判断2都成立,最终的结果才为真 |
| **判断1 -o 判断2** | 逻辑或,判断1和判断2有一个成立,最终的结果就为真 |
| **!判断** | 逻辑非,使原始的判断式取反 |

**案例:**

**[root@localhost ~]# aa=11**

**[root@localhost ~]# [ -n "$aa" -a "$aa" -gt 23 ] && echo "yes" || echo "no"**

**no**

**#判断变量aa是否有值,同的判断变量aa的是否大于23**

**#因为变星aa值不大于23,所以虽然第一个判为真,返回的结果也是假**

### 单分支if语句

#### 单分支条件语句

**if [ 条件判断式 ];then**

**程序**

**fi**

**或者**

**if [ 条件判断式 ]**

**then**

**程序**

**fi**

#### 单分支条件语句需要注意几个点

* **if语句使用fi结尾,和一般语言使用大括号结尾不同**
* **[ 条件判断式 ]就是使用test命令判断,所以中括号和条件判断式之间必须有空格**
* **then后面跟符合条件之后执行的程序,可以放在[]之后;用分割。也可以换行写,就不需要了**

#### 例子1:判断登陆的用户是否root

**[root@localhost ~]# vim is\_user\_root.sh**

**#!/bin/bash**

**test=$(env | grep "USER" | cut -d "=" -f 2)**

**if [ $test=="root" ];then**

**echo "current user is root"**

**fi**

#### 例子2:判断分区使用率

**#/bin/bash**

**#判断分区使用率是否超过80%，假如超过则报警**

**rate=$(df -h | grep '/$' | awk '{print $5}' | cut -f 1 -d '%')**

**#把根分区使用率作为变量值赋予变量rate**

**if [ $rate -ge 80 ]**

**then**

**echo '/目录磁盘使用率达到或超过80%，请留意'**

**fi**

### 双分支if语句

#### 双分支if条件语句

**if [ 条件判断式 ]**

**then**

**条件成立时,执行的程序**

**else**

**条件不成立时,执行的另一个程序**

**fi**

#### 判断Nginx服务是否启动

**#!/bin/bash**

**#判断nginx是否启动，记录日志，没启动的话启动nginx服务器**

**test=$(ps aux | grep 'nginx' | grep -v 'grep')**

**if [ -n "$test" ]**

**then**

**echo "$(date) nginx is ok" >> /tmp/nginx\_check.log**

**else**

**systemctl start nginx > /dev/null**

**echo "$(date) nginx restart" >> /tmp/nginx\_restart.log**

**fi**

### 多分支if语句

#### 多分支if条件语句

**if [ 条件判断式1 ]**

**then**

**当条件判断式1成立时,执行程序1**

**elif [ 条件判断式2 ]**

**then**

**当条件判断式2成立时,执行程序2**

**…省略更多条件…**

**else**

**当所有条件都不成立时,最后执行此程序**

**fi**

#### 判断用户输入的是什么文件

**#!/bin/bash**

**read -p "Please input a filename:" file**

**#接收键盘的输入,并赋予变量file**

**if [ -z "$file" ]**

**#判断加变量是否为空**

**then**

**echo "Error, please input a filename"**

**exit 1**

**elif [ ! -e "$file" ]**

**#判断file的值是否存在**

**then**

**echo "Your input is not a file!"**

**exit 2**

**elif [ -f "$file" ]**

**#判断的file的值是否为普通文件**

**then**

**echo "$file is a regulare file!"**

**elif [ -d "$file" ]**

**#判断file的值是否为目录文件**

**then**

**echo "$file is a directory!"**

**else**

**echo "$file is an other file!"**

**fi**

### case语句

**case语句和if…elif...else语句一样都是多分支条件语句,不过和多分支条件语句不同的是,case语句只能判断一种条件关系,而if语句可以判断多种条件关系。**

#### case语句

**case $变量名 in**

**“值1”)**

**如果变量的值等于值1,则执行程序1**

**;;**

**“值2”)**

**;;**

**如果变量的值等于值2,则执行程序2**

**…省略其他分支…**

**\*)**

**如果变量的值都不是以上的值,则执行此程序**

**;;**

**esac**

#### 判断用户输入

**#!/bin/bash**

**#判断用户输入**

**read -p "Please choose yes/no:" -t 30 cho**

**case $cho in**

**"yes")**

**echo "Your choose is yes!"**

**;;**

**"no")**

**echo "Your choose is no!"**

**;;**

**\*)**

**echo "Your choose is error!"**

**;;**

**esac**

### for循环

#### 语法一

**for 变量 in 值1 值2 值3…**

**do**

**程序**

**done**

**如**

**#!/bin/bash**

**for i in 1 2 3 4 5**

**do**

**echo $i**

**done**

#### 批量解压缩脚本

**#!/bin/bash**

**#批量解压/root/下所有tar.gz的压缩文件**

**cd /root/**

**for i in $(ls \*.tar.gz)**

**do**

**tar -zxf $i &> /dev/null**

**done**

#### 语法二

**for ((初始值;循环控制条件;变量变化))**

**do**

**程序**

**done**

#### 从1加到100

**#!/bin/bash**

**#从1加到100**

**s=0**

**for((i=1;i<=100;i=i+1))**

**do**

**s=$(($s+$i))**

**done**

**echo "The sum of 1+2+...+100 is: $s"**

#### 批量添加删除指定数量的用户

**#!/bin/bash**

**#批量添加指定数量的用户**

**read -p "Please input user name: " -t 30 name**

**read -p "Please input the number of users: " -t 30 num**

**read -p "Please input the password of users: " -t 30 pass**

**if [ ! -z "$name" -a ! -z "$num" -a ! -z "$pass" ]**

**then**

**y=$(echo $num | sed 's/[0-9]//g')**

**if [ -z "$y" ]**

**then**

**for ((i=1;i<=$num;i=i+1))**

**do**

**/usr/sbin/useradd $name$i &> /dev/null**

**echo $pass | /usr/bin/passwd --stdin $name$i &> /dev/null**

**done**

**fi**

**fi**

### while循环和until循环

#### while循环

**while循环是不定循环,也称作条件循环。只要条件判断式成立,循环就会一直继续,直到条件判断式不成立,循环才会停止。这就和for的固定循环不太一样了**

#### while循环语法

**while [条件判断式]**

**do**

**程序**

**done**

#### while实现从1加到100

**#!/bin/bash**

**#从1加到100**

**i=1**

**s=0**

**while [ $i -le 100 ]**

**#如果变量i的值小于等于100,则执行循环**

**do**

**s=$(($s+$i))**

**i=$(($i+1))**

**done**

**echo “The sum is: $s”**

#### until循环

**until循环,和 while循环相反, until循环时只要条件判断式不成立则进行循环,并执行循环程序。一旦循环条件成立,则终止循环。**

#### until循环语法

**until [ 条件判断式 ]**

**do**

**程序**

**done**

#### until实现从1加到100

**#!/bin/bash**

**#从1加到100**

**i=1**

**s=0**

**until [ $i -gt 100 ]**

**#循环直到变量i的值大于100,就停止循环**

**do**

**s=$(($s+$i))**

**i=$(($i+1))**

**done**

**echo “The sum is: $s”**

# 系统管理

**程序、进程、服务、线程还有python里的协程，大家都知道是什么吗？**

## 进程管理

### 进程管理简介

#### 进程简介

**进程是正在执行的一个程序或命令,每一个进程都是一个运行的实体,都有自己的地址空间,并占用一定的系统资源。**

#### 进程管理的作用

* **判断服务器健康状态**
* **查看系统中所有进程**
* **杀死进程**

### 进程查看ps命令和 pstree命令

#### 查看所有进程

**ps -aux**

**#主要作用：查看系统中所有进程**

**ps -le**

**#查看系统中所有进程,使用Linux标准命令格式**

**选项**

**a:显示一个终端的所有进程,除了会话引线**

**u:显示进程的归属用户及内存的使用情况**

**x:显示没有控制终端的进程**

**-l:长格式显示。显示更加详细的信息**

**-e:显示所有进程,和-A作用一致**

#### ps命令的输出

* **USER:该进程是由哪个用户产生的;**
* **PID:进程的ID号;**
* **%CPU:该进程占用CPU资源的百分比,占用越高,进程越耗费资源;**
* **%MEM:该进程占用物理内存的百分比,占用越高,进程越耗费资源;**
* **VSZ:该进程占用虚拟内存的大小,单位KB;**
* **RSS:该进程占用实际物理内存的大小,单位KB;**
* **TTY:该进程是在哪个终端W中运行的。其中tty1-tty7代表本地控制台终端,tty1-tty6是本地的字符界面终端,tty7是图形终端。pts/0-255(65355) 代表虚拟终端。**
* **STAT:进程状态。常见的状态有**
  + **R:运行**
  + **S:睡眠**
  + **T:停止状态**
  + **Z:僵尸(zombie)进程**
  + **s:包含子进程**
  + **+:位于后台**
* **START:该进程的启动时间**
* **TIME:该进程占用CPU的运算时间,注意不是系统时间**
* **COMMAND:产生此进程的命令名**

#### 查看进程树

**[root@ localhost ~]# pstree [选项]**

**选项:**

**-p:显示进程的PID**

**-u:显示进程的所属用户**

### 进程的查看top命令

#### 查看系统健康状态

**[root@localhost ~]# top [选项]**

**选项**

* **-d秒数:指定top命令每隔几秒更新。默认是3秒**
* **-b:使用批处理模式输出。一般和"-n"选项合用**
* **-n次数:指定top命令执行的次数。一般和“-b”选项合用**

**在top命令的交互模式当中可以执行的命令**

* **?或h:显示交互模式的帮助**
* **P:以CPU使用率排序,默认就是此项**
* **M:以内存的使用率排序**
* **N:以PID排序**
* **q:退出top**

##### 第一行信息为任务队列信息

|  |  |
| --- | --- |
| **内容** | **说明** |
| **12:26:46** | 系统当前时间 |
| **up 1 day, 13: 32** | 系统的运行时间,本机已经运行1天13小时32分钟 |
| **2 users** | 当前登录了两个用户 |
| **load average: 0.00, 0.00,0.00** | **系统在之前1分钟,5分钟,15分钟的平均负载,一般认为小于1时,负载较小。如果大于1,系统已经超出负荷。** |

**PS: 有计划的重启是服务器维护的基本规则**

##### 第二行为进程信息

|  |  |
| --- | --- |
| **内容** | **说明** |
| **Tasks: 95 total** | 系统中的进程总数 |
| **1 running** | 正在运行的进程数 |
| **94 sleeping** | 睡眠的进程 |
| **0 stopped** | 正在停止的进程 |
| **0 zombie** | 僵尸进程。如果不是0,需要手工检查僵尸进程 |

##### 第三行为CPU信息

|  |  |
| --- | --- |
| **内容** | **说明** |
| **Cpu(s): 0.1%us** | 用户模式占用的CPU百分比 |
| **0.1%sy** | 系统模式占用的CPU百分比 |
| **0.0‰ni** | 改变过优先级的用户进程占用的CPU百分比 |
| **99.7%id** | **空闲CPU的CPU百分比** |
| **0.1%wa** | 等待输入/输出的进程的占用CPU百分比 |
| **0.0%hi** | 硬中断请求服务占用的CPU百分比 |
| **0.1%si** | 软中断请求服务占用的CPU百分比 |
| **0.0%st** | st( Steal time)虚拟时间百分比。就是当有虚拟机时,虚拟CPU等待实际CPU的时间百分比。 |

##### 第四行为物理内存信息

|  |  |
| --- | --- |
| **内容** | **说明** |
| **Mem: 625344k total** | 物理内存的总量,单位KB |
| **571504k used** | 已经使用的物理内存数量 |
| **53840k free** | 空闲的物理内存数量,我们使用的是虚拟机,总共只分配了628MB内存,所以只有53MB的空闲内存了 |
| **65800k buffers** | 作为**缓冲**的内存数量 |

##### 第五行为交换分区(swap)信息

|  |  |
| --- | --- |
| **内容** | **说明** |
| **Swap: 524280k total** | 交换分区(虚拟内存)的总大小 |
| **Ok used** | 已经使用的交互分区的大小 |
| **524280k free** | 空闲交换分区的大小 |
| **409280k cached** | 作为**缓存**的交互分区的大小 |

**CPU >> 内存 >> 硬盘的速度**

**100G/s 2.4G/s 0.5G/s**

**速度数据为约等于数据**

**缓存和缓冲的区别:**

* **缓存(cache)是用来加速数据从硬盘中“读取”**
* **缓冲( buffer)是用来加速数据“写入”硬盘**

#### 查看所有进程

**[root@localhost ~]# top -b -n1 > /root/top.log**

### 杀死进程

#### kill命令

**[root@localhost ~]# kill -l**

**#查看可用进程信号**

|  |  |  |
| --- | --- | --- |
| **信号代号** | **信号名称** | **说明** |
| **1** | **SIGHUP** | 该信号让进程立即关闭,然后重**新读取配置文件之后重启**。 |
| 2 | SIGINT | 程序终止信号,用于终止前台进程。相当于输出 ctrl+c快捷键。 |
| 8 | SIGFPE | 在发生致命的算术运算错误时发出.不仅包括浮点运算错误,还包括溢出及除数为0等其它所有的算术的错误。 |
| **9** | **SIGKILL** | 用来立即结束程序的运行.本信号不能被阻塞、处理和忽略。一般用于**强制终止进程**。 |
| 14 | SIGALRM | 时钟定时信号,计算的是实际的时间或时钟时间.alarm函数使用该信号。 |
| **15** | **SIGTERM** | 正常结束进程的信号,**kill命令的默认信号**。有时**如果进程已经发生问题,这个信号是无法正常终止进程的**,我们才会尝试SIGKILL信号,也就是信号9。 |
| 18 | SIGCONT | 该信号可以让暂停的进程恢复执行,本信号不能被阻断。 |
| 19 | SIGSTOP | 该信号可以暂停前台进程,相当于输入ctr快捷键。本信号不能被阻断。 |

**例如**

* **kill -1 2235**

**#重启进程**

* **kill 2235**

**#正常结束进程**

* **kill -9 2236**

**#强制杀死进程**

#### killall命令

**killall [选项] [信号] 进程名**

**#按照进程名杀死进程**

**选项:**

**-i:交互式,询问是否要杀死某个进程**

**-I:忽略进程名的大小写**

#### pkill命令

**pkill [选项] [信号] 进程名**

**#按照进程名终止进程**

**选项:**

**-t终端号:按照终端号踢出用户**

**案例:按照终端号踢出用户**

**w**

**使用w命令查询本机已经登录的用户**

**pkill -9 -t pts/1**

**#强制从pts/1虚拟终端登录的进程**

### 修改进程优先级

#### 进程优先级简介

**Linux操作系统是一个多用户、多任务的操作系统, Linux系统中通知运行着非常多的进程。但是CPU在同一个时钟周期内只能运算一个指令。进程优先级决定了每个进程处理的先后顺序**

#### 修改进程优先级

**ps -le**

**PRI代表 Priority,NI代表Nice。这两个值都是优先级,数字越小代表该进程优先级越高**

**修改N值时的几个注意事项:**

* **NI的值的范围是-20到19;**
* **普通用户调整NI值的范围是0到19,而且只能调整自己的进程**
* **普通用户只能调高NI值,而不能降低,如原本NI值为0,则只能调整为大于0;**
* **root用户才能设定进程NI值为负值,而且可以调整任何用户的进程**
* **PRI(最终值) = PRI(原始值) + NI**
* **用户只能修改NI的值,不能直接修改PRI**

#### ~~nice命令~~

**~~nice [选项] 命令~~**

**~~#nice命令可以给新执行的命令直接赋予NI值,但是不能修已经存在进程的NI值~~**

**~~选项:~~**

**~~-n NI值:给命令赋予N值。~~**

**~~例如~~**

**~~nice -n -5 systemctl start nginx~~**

#### renice命令

**renice [优先级] PID**

**#renice命令是修改已经存在进程的NI值的命令**

**例如**

**renice -10 2125**

## 工作（jobs）管理

**工作管理指的是在单个登录终端中(也就是登录的shell界面中)同时管理多个工作的行为。**

**Demo代码，暂时不要求掌握！！！:**

**import** logging  
  
**import** tornado.httpserver  
**import** tornado.ioloop  
**import** tornado.options  
**import** tornado.web  
  
**from** tornado.options **import** define, options  
  
define(**"port"**, default=8080, help=**"run on the given port"**, type=int)  
  
  
**class** MainHandler(tornado.web.RequestHandler):  
 **def** get(self):  
 self.write(**"Hello, world"**)  
  
  
**def** main():  
 tornado.options.parse\_command\_line()  
 application = tornado.web.Application([  
 (**r"/"**, MainHandler),  
 ])  
 http\_server = tornado.httpserver.HTTPServer(application)  
 http\_server.listen(options.port)  
 logging.info(**f'start tornado server on {options.port} port, stop with Ctrl + c'**)  
 **try**:  
 tornado.ioloop.IOLoop.current().start()  
 **except** KeyboardInterrupt:  
 logging.info(**'stop tornado server...'**)  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 main()

### 查看工作列表

**[root@localhost ~]# jobs**

**#查看所有工作列表**

**注意事项：**

* **当前的登录终端,只能管理当前终端的工作,而不能管理其他登录终端的工作**
* **放入后台的命令必须可以持续运行一段时间,这样我们才能扑捉和操作这个工作**
* **放入后台执行的命令不能和前台用户有交互或需要前台输入,否则放入后台只能暂停,而不能执行**

### 工作管理

#### 把进程放入后台

* **find / -name hello.sh &**

**#把命令放入后台,并在后台执行**

* **top**

**#按下ctrl+z快捷键,放在后台暂停**

#### 查看后台的工作

**jobs [-l]**

**选项**

**-l:显示工作的PID**

**注:“+”号代表最近一个放入后台的作,也是工作恢复的,默认恢复的工作。“-”号代表倒数第二个放入后台的工作**

#### 将后台暂停的工作恢复到前台执行

**[root@localhost ~]# fg %工作号**

**参数:**

**%工作号: %号可以省略,但是注意工作号和PID的区别,**

**此处用%号标识的目的也有强调这不是PID的**

#### 终止后台工作

**[root@localhost ~]# kill %工作号**

#### 把后台暂停的工作恢复到后台执行

**root@ localhost~]# bg %工作号**

**注:后台恢复执行的命令,是不能和前台有交互的,否则不能恢复到后台执行**

**可以这样理解 & = (Ctrl + z) + bg**

### 后台命令脱离登陆终端执行

**把命令放入后台,只能在当前登录终端执行。一旦退出或关闭终端,后台程序就会停止。**

#### 后台命令脱离登陆终端执行的方法

* **第一种方法是把需要后台执行的命令加入/etc/rc.local文件**
* **第二种方法是使用系统定时任务,让系统在指定的时间执行某个后台命令（可能存在问题，不推荐使用）**
* **第三种方法是使用nohup命令（标准和推荐方法）**

#### nohup命令

**[root@ localhost ~]# nohup [命令] &**

**案例**

**[root@localhost ~]# vim for.sh**

**#!/bin/bash**

**for((i=1;i<=100;i=i+1))**

**#循环1000次**

**do**

**echo $i >> /root/for.log**

**sleep 2s**

**done**

**[root@localhost ~]# nohup ./for.sh &**

## Systemd

**说明1：本小节也是依据参考阅读编写，可以直接阅读参考阅读原文，而不阅读本章节！**

**说明2：Linux优化可以按照《Linux服务列表.pdf》文档中的建议，关闭没有必要开启的服务，这是服务器优化中很重要的举措，详见《Linux服务列表.pdf》**

**参考阅读：**

**<http://www.ruanyifeng.com/blog/2016/03/systemd-tutorial-commands.html>**

**<http://www.ruanyifeng.com/blog/2016/03/systemd-tutorial-part-two.html>**

### Systemd简介

**Systemd是Linux系统工具，用来启动[守护进程](http://www.ruanyifeng.com/blog/2016/02/linux-daemon.html" \t "_blank)，已成为大多数发行版的标准配置。**

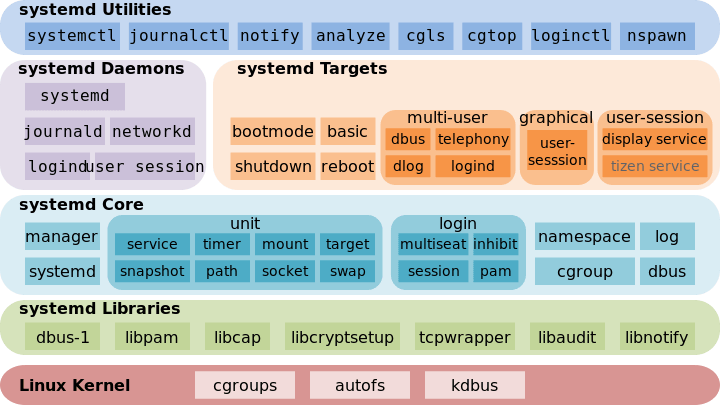
**它的设计目标是，为系统的启动和管理提供一套完整的解决方案。**

**字母d是守护进程（daemon）的缩写。 Systemd 这个名字的含义，就是它要守护整个系统。**

**使用了 Systemd，就不需要再用init了。Systemd 取代了initd，成为系统的第一个进程（PID 等于 1），其他进程都是它的子进程。**

**Systemd 的优点是功能强大，使用方便，缺点是体系庞大，非常复杂。事实上，现在还有很多人反对使用 Systemd，理由就是它过于复杂，与操作系统的其他部分强耦合，违反"keep simple, keep stupid"的[Unix 哲学](http://www.ruanyifeng.com/blog/2009/06/unix_philosophy.html" \t "_blank)。**

**Systemd 架构图**

****

### 管理

**Systemd 并不是一个命令，而是一组命令，涉及到系统管理的方方面面。**

* **systemctl是Systemd的主命令，用于管理系统。**
* **systemd-analyze命令用于查看启动耗时。**
* **hostnamectl命令用于查看和设置当前主机的信息。**
* **localectl命令用于查看本地化设置。**
* **timedatectl命令用于查看当前时区设置。**
* **loginctl命令用于查看当前登录的用户。**

### Unit单位

**Systemd 可以管理所有系统资源。不同的资源统称为 Unit（单位）。**

#### Unit一共分成12种

* **Service unit：系统服务**
* **Target unit：多个 Unit 构成的一个组**
* **Device Unit：硬件设备**
* **Mount Unit：文件系统的挂载点**
* **Automount Unit：自动挂载点**
* **Path Unit：文件或路径**
* **Scope Unit：不是由 Systemd 启动的外部进程**
* **Slice Unit：进程组**
* **Snapshot Unit：Systemd 快照，可以切回某个快照**
* **Socket Unit：进程间通信的 socket**
* **Swap Unit：swap 文件**
* **Timer Unit：定时器**

#### Unit 的状态

**systemctl status命令用于查看系统状态和单个 Unit 的状态。**

* **$ systemctl status**

**# 显示系统状态**

* **$ systemctl status nginx.service**

**# 显示单个 Unit 的状态，.service可以不写**

* **$ systemctl -H root@192.168.222.62 status sshd.service**

**# 显示远程主机的某个 Unit 的状态**

**除了status命令，systemctl还提供了三个查询状态的简单方法，主要供脚本内部的判断语句使用。**

* **$ systemctl is-active application.service**

**# 显示某个 Unit 是否正在运行**

* **$ systemctl is-failed application.service**

**# 显示某个 Unit 是否处于启动失败状态**

* **$ systemctl is-enabled application.service**

**# 显示某个 Unit 服务是否建立了启动链接**

#### Unit 管理命令

**对于用户来说，最常用的是下面这些命令，用于启动和停止 Unit（主要是 service）。**

* **$ sudo systemctl start apache.service**

**# 立即启动一个服务**

* **$ sudo systemctl stop apache.service**

**# 立即停止一个服务**

* **$ sudo systemctl restart apache.service**

**# 重启一个服务**

* **$ sudo systemctl kill apache.service**

**# 杀死一个服务的所有子进程**

* **$ sudo systemctl reload apache.service**

**# 重新加载一个服务的配置文件**

* **$ sudo systemctl daemon-reload**

**# 重载所有修改过的配置文件**

* **$ systemctl show httpd.service**

**# 显示某个 Unit 的所有底层参数**

* **$ systemctl show -p CPUShares httpd.service**

**# 显示某个 Unit 的指定属性的值**

* **$ sudo systemctl set-property httpd.service CPUShares=500**

**# 设置某个 Unit 的指定属性**

#### 依赖关系

**Unit 之间存在依赖关系：A 依赖于 B，就意味着 Systemd 在启动 A 的时候，同时会去启动 B。**

**systemctl list-dependencies命令列出一个 Unit 的所有依赖。**

**$ systemctl list-dependencies nginx.service**

**上面命令的输出结果之中，有些依赖是 Target 类型，默认不会展开显示。如果要展开 Target，就需要使用--all参数。**

**$ systemctl list-dependencies --all nginx.service**

### Unit的配置文件

#### 简介

**每一个 Unit 都有一个配置文件，告诉 Systemd 怎么启动这个 Unit ，Systemd 默认从目录/etc/systemd/system/读取配置文件。里面存放的大部分文件都是符号链接，指向目录/usr/lib/systemd/system/，真正的配置文件存放在那个目录**

**例如**

**使用rpm方式安装完nginx后，会生成**

**/etc/systemd/system/multi-user.target.wants/nginx.service的配置文件**

#### 设置开机启动

* **[root@localhost ~]# systemctl enable nginx**

**# 设置开机启动nginx**

**Created symlink from /etc/systemd/system/multi-user.target.wants/nginx.service to /usr/lib/systemd/system/nginx.service.**

* **[root@localhost ~]# systemctl disable nginx**

**# 设置开机不启动nginx**

**Removed symlink /etc/systemd/system/multi-user.target.wants/nginx.service.**

#### 配置文件状态

**[root@localhost ~]# systemctl list-unit-files**

**#列出所有配置文件**

**这个列表显示每个配置文件的状态，一共有四种。**

* **enabled：已建立启动链接**
* **disabled：没建立启动链接**
* **static：该配置文件没有[Install]部分（无法执行），只能作为其他配置文件的依赖**
* **masked：该配置文件被禁止建立启动链接**

#### 配置文件格式

**配置文件就是普通的文本文件，可以用文本编辑器打开。**

**systemctl cat命令可以查看配置文件的内容。**

**如：**

**[root@localhost ~]# systemctl cat nginx**

**# /usr/lib/systemd/system/nginx.service**

**[Unit]**

**Description=nginx - high performance web server**

**Documentation=http://nginx.org/en/docs/**

**After=network-online.target remote-fs.target nss-lookup.target**

**Wants=network-online.target**

**[Service]**

**Type=forking**

**PIDFile=/var/run/nginx.pid**

**ExecStart=/usr/sbin/nginx -c /etc/nginx/nginx.conf**

**ExecReload=/bin/kill -s HUP $MAINPID**

**ExecStop=/bin/kill -s TERM $MAINPID**

**[Install]**

**WantedBy=multi-user.target**

**字段意思详情可以参考**

**http://www.ruanyifeng.com/blog/2016/03/systemd-tutorial-commands.html**

**思考一个问题，源码方式安装的nginx怎样让systemd管理起来？**

### Target

**启动计算机的时候，需要启动大量的 Unit。如果每一次启动，都要一一写明本次启动需要哪些 Unit，显然非常不方便。Systemd 的解决方案就是 Target。**

**简单说，Target 就是一个 Unit 组，包含许多相关的 Unit 。启动某个 Target 的时候，Systemd 就会启动里面所有的 Unit。从这个意义上说，Target 这个概念类似于"状态点"，启动某个 Target 就好比启动到某种状态。**

**传统的init启动模式里面，有 RunLevel 的概念，跟 Target 的作用很类似。不同的是，RunLevel 是互斥的，不可能多个 RunLevel 同时启动，但是多个 Target 可以同时启动。**

#### Target命令

**$ systemctl list-unit-files --type=target**

**# 查看当前系统的所有 Target**

**$ systemctl list-dependencies multi-user.target**

**# 查看一个 Target 包含的所有 Unit**

**$ systemctl get-default**

**# 查看启动时的默认 Target**

**$ sudo systemctl set-default multi-user.target**

**# 设置启动时的默认 Target**

**$ sudo systemctl isolate multi-user.target**

**# 切换 Target 时，默认不关闭前一个 Target 启动的进程，**

**# systemctl isolate 命令改变这种行为，**

**# 关闭前一个 Target 里面所有不属于后一个 Target 的进程**

#### Target 与 传统 RunLevel 的对应关系

**Traditional runlevel New target name Symbolically linked to...**

**Runlevel 0 | runlevel0.target -> poweroff.target**

**Runlevel 1 | runlevel1.target -> rescue.target**

**Runlevel 2 | runlevel2.target -> multi-user.target**

**Runlevel 3 | runlevel3.target -> multi-user.target**

**Runlevel 4 | runlevel4.target -> multi-user.target**

**Runlevel 5 | runlevel5.target -> graphical.target**

**Runlevel 6 | runlevel6.target -> reboot.target**

### 日志管理

**Systemd 统一管理所有 Unit 的启动日志。带来的好处就是，可以只用journalctl一个命令，查看所有日志（内核日志和应用日志）。日志的配置文件是/etc/systemd/journald.conf。**

**$ sudo journalctl**

**# 查看所有日志（默认情况下 ，只保存本次启动的日志）**

## 系统资源查看

### vmstat命令监控系统资源

**vmstat [刷新延时 刷新次数]**

**例如**

**vmstat 1 3**

**procs:进程信息字段**

**r:等待运行的进程数,数量越大,系统越繁忙。**

**b:不可被唤醒的进程数量,数量越大,系统越繁忙。**

**memory:内存信息字段**

**swpd:虚拟内存的使用情况,单位KB**

**free:空闲的内存容量,单位KB。**

**buff:缓冲的内存容量,单位KB。**

**cache:缓存的内存容量,单位KB。**

**swap:交换分区的信息字段**

**si:从磁盘中交换到内存中数据的数量,单位KB**

**so:从内存中交换到磁盘中数据的数量,单位KB。**

**此两个数越大,证明数据需要经常在磁盘和内存之间交换,系统性能越差。**

**io:磁盘读写信息字段**

**bi:从块设备读入数据的总量,单位是块。**

**bo:写到块设备的数据的总量,单位是块。**

**此两个数越大,代表系统的I/O越繁忙。**

**systen:系统信息字段:**

**in:每秒被中断的进程次数。**

**cs:每秒钟进行的事件切换次数。此两个数越大,代表系统与接口设备的通信非常繁忙**

**CPU:CPU信息字段:**

**us:非内核进程消耗CPU运算时间的百分比**

**sy:内核进程消耗CPU运算时间的百分比。**

**id:空闲CPU的百分比。**

**wa:等待/O所消耗的cPU百分比**

**st:被虚拟机所盗用的cPU占比。**

### dmesg开机时内核检测信息

**dmesg**

**dmesg | grep CPU**

**思考一下：假如面试官问你怎样查看linux服务器的硬件状态，怎么回答？**

### free命令查看内存使用状态

**free [-b|-k|-m|-g]**

**选项**

**-b:以字节为单位显示**

**-k:以KB为单位显示,默认就是以KB为单位显示**

**-m:以MB为单位显示**

**-g:以GB为单位显示**

* **第一行:total是总内存数,used是已经使用的内存数,free是空闲的内存数, shared是多个进程共享的内存总数,buffers是缓冲内存数, cached是缓存内存数。默认单位是KB**
* **第二行:-/ buffers/ cache的内存数,相当于第一行的used-buffers- cached。+/ buffers/ cache的内存数,相当于第一行的free+ buffers+ cached。**
* **第三行:total是swap的总数;used是已经使用的swap数,free是空闲的Wap数。默认单位是KB**

### 查看CPU信息

**cat /proc/cpuinfo**

### uptime命令

**uptime**

**#显示系统的启动的间和平均负载,也就是top命令的第一行W命令也可以看到这个数据**

### 查看系统与内核相关信息

**uname [选项]**

**选项:**

**-a:查看系统所有相关信息**

**-r:查看内核版本;**

**-s:查看内核名称**

**判断当前系统的位数**

**file /bin/ls**

**getconf LONG\_BIT**

**查询当前Lnux系统的发行版本**

**~~Isb\_release -a~~**

**ll /etc/\*-release**

**cat /etc/centos-release**

### 列出进程打开或使用的文件信息

**lsof [选项]**

**# list open files列出进程凋用成打开的文件的信启**

**选项:**

**-c字符串:只列出以字符串开头的进程打开的文件**

**-u用户名:只列出某个用户的进程打开的文件**

**-p pid:列出某个PID进程打开的文件**

**-i:端口号**

**lsof -i:8080**

**#查询特定端口号的占用情况，可配合kill使用，杀死进程**

**Isof | more**

**#查询系统中所有进程调用的文件**

**Isof /sbin/init**

**#查询某个文件被哪个进程调用**

**Isof -c httpd**

**#查着httpd进程调用了哪些文件**

**Isof -u root**

**#按照用户名,查询某用户的进程调用的文件名**

### 将指定的文件输出到标准设备

**tail [选项] 文件名**

**选项**

**-f 循环读取**

**-n<行数> 显示行数**

**tail -f access.log**

**#此命令显示 access.log 文件的最后10(默认)行。当将某些行添加至 access.log 文件时，tail 命令会继续显示这些行。 显示一直继续，直到您按下（Ctrl-C）组合键停止显示。**

## 系统定时任务

### at一次性定时任务

#### 确定at安装

**systemctl list-unit-files | grep atd**

**#at服务是否安装**

**systemctl start atd**

**#at服务的启动**

#### at的访向问控制

* **如果系统中有/etc/at.alloy文件,那么只有写入/etc/at.allow文件(白名单)中的用户可以使用at命令(/etc/at.deny文件会被忽略)**
* **如果系统中没有/etc/ at.allow文件,只有/etc/at.deny文件那么写/etc/at.deny文件(黑名单)中的用户不能使用at命令。对root不起作用**
* **如果系统中这两个文件都不存在,那么只有root用户可以使用at命令**

#### at命令

**at [选项] 时间**

**选项:**

**-m:当at工作完成后,无论是否命令有输出,都用emai通知执行at命令的用户**

**-c工作号:显示该a工作的实际内容**

**时间:**

**HH:MM 例如:02:30**

**HH:MM YYYY-MM-DD 例如02:30 2018-07-25**

**HH:MM[am|pm] [month] [date] 例如02:30 July 25**

**HH:MM[am|pm] + [minutes|hours|days|weeks] 例如now + 5minutes**

#### 其他at管理命令

**atq**

**#查询当前服务器上的at工作**

**atrm [工作号]**

**#删除指定的at任务**

#### 例子1

**at now + 2 minutes**

**#在两分钟之后执行helo.sh脚本**

**at> /root/hello. sh >> /root/hello.log**

**按Ctrl+D组合键正确结束**

**按Ctrl+C 取消**

#### 例子2

**at 02:00 2018-07-26**

**#在指定的时间重启**

**at> /bin/sync**

**#数据同步，把内存里的数据往硬盘上转移**

**at> /sbin/shutdown -r now**

### crontab循环定时任务

**务必掌握！**

#### crond服务管理与访问控制

**[root@localhost ~]# systemctl list-unit-files | grep crond**

**[root@localhost ~]# systemctl start crond**

**[root@localhost ~]# systemctl enable crond**

#### 访问控制

* **当系统中有/etc/cron.allow文件时,只有写入此文件的用户可以使用 crontab命令,**

**没有写入的用户不能使用 crontab命令。同样如果有此文件,/etc/cron.deny文件会被忽略,/etc/cron.allow文件的优先级更高**

* **当系统中只有/etc/cron.deny文件时,则写入此文件的用户不能使用 crontab命令,没有写入文件的用户可以使用 crontab命令**
* **如果系统中这两个文件都不存在,那么只有root用户可以使用crontab命令**

#### 用户的 crontab设置

**[root@ localhost~]# crontab [选项]**

**选项**

**-e:编辑 crontab定时任务**

**-l:查询 crontab任务**

**-r:删除当前用户所有的 crontab任务**

#### crontab -e

**[root@localhost -]# crontab -e**

**进入 crontab编辑界面。会打开vim编辑你的工作。**

**\* \* \* \* \* 执行的任务**

**位置含义**

|  |  |  |
| --- | --- | --- |
| **项目** | **含义** | **范围** |
| **第一个 \*** | **一小时当中的第几分钟** | **0-59** |
| **第二个 \*** | **一天当中的第几小时** | **0-23** |
| **第三个 \*** | **一个月当中的第几天** | **1-31** |
| **第四个 \*** | **一年当中的第几月** | **1-12** |
| **第五个 \*** | **一周当中的星期几** | **0-7(0和7都代表星期日)** |

**思考一下，如果想每隔10分钟执行一次怎么办？**

**特殊符号**

|  |  |
| --- | --- |
| **特殊符号** | **含义** |
| **\*** | **代表任何时间。比如第一个“\*”就代表一小时中每分钟都执行次的意思。** |
| **,** | **代表不连续的时间。比如“0 8,12,16 \* \* \*命令”,就代表在每天的8点0分,12点0分,16点0分都执行一次命令** |
| **-** | **代表连续的时间范围。比如“0 5 \* \* 1-6命令”,代表在周一到周六的凌晨5点0分执行命令** |
| **\*/n** | **代表每隔多久执行一次。比如“\*/10 \* \* \* \*命令”,代表每隔10分钟就执行一遍命令** |

**案例**

|  |  |
| --- | --- |
| **时间** | **含义** |
| **45 22 \* \* \*命令** | **在22点45分执行命令** |
| **0 17 \* \* 1命令** | **每周1的17点0分执行命令** |
| **0 5 1,15 \* \*命令** | **每月1号和15号的凌晨5点0分执行命令** |
| **40 4 \* \* 1-5命令** | **每周一到周五的凌晨4点40分执行命令** |
| **\*/10 4 \* \* \*命令** | **每天的凌晨4点,每隔10分钟执行一次命令** |
| **0 0 1,15 \* 1命令** | **每月1号和15号,每周1的0点0分都会执行命令。**  **注意:星期几和几号最好不要同时出现,因为他**  **们定义的都是天。非常容易让管理员混乱。** |

#### crontab注意事项

* **六个选项都不能为空,必须填写。如果不确定使用“\*”代表任意时间**
* **crontab定时任务,最小有效时间是分钟,最大时间范围是月。像2018年某时执行,3点30分30秒这样的时间都不能识别**
* **在定义时间时,日期和星期最好不要在一条定时任务中出现，因为它们都是以天作为单位,非常容易让管理员混乱。**
* **在定时任务中,不管是直接写命令,还是在脚本中写命令，最好都是用绝对路径**

**案例**

* **\*/5 \* \* \* \* /bin/echo “11” >> /tmp/test**
* **5 5 \* \* 2 /sbin/shutdown -r now**
* **\* \*/1 \* \* \* curl https://www.baidu.com**
* **0 5 1,10,15 \* \* /root/sh/autobak sh**

**题目1: \* \* \* \* \* 代表什么？**

**题目2:怎样在2017年9-10月间,每天的05点05分30秒执行一条命令。**

**大家能找出题目中的问题吗？**

### 系统的 crontab设置

#### 系统的定时任务

**“crontab-e”是每个用户执行的命令,也就是说不同的用户身份可以执行自己的定时任务。可是有些定时任务需要系统执行,这时我们就需要编辑/etc/crontab这个配置文件了**

#### CentOS5中的配置文件

**[root@localhost ~]# vi / etc/crontab**

**SHELL=/bin/bash**

**PATH=/sbin bin: /usr/sbin /usr/bin**

**MAILTO=root**

**HOME=/**

**run-parts**

**01 \* \* \* \* root run-parts /etc/cron.hourly**

**02 4 \* \* \* root run-parts /etc/cron.daily**

**22 4 \* \* 0 root run-parts /etc/cron.weekly**

**42 4 1 \* \* root run-parts /etc/cron.monthly**

#### 执行系统的定时任务的方法

* **手工执行定时任务**
* **系统定时任务**
  1. **第一种是把需要定时执行的脚本复制到/etc/cron.{daily, weekly,monthly}目录中的任意一个**
  2. **第二种是修改/etc/crontab配置文件**

### anacron配置

#### anacron是什么

**anacron是用来保证在系统关机的时候错过的定时任务,可以在系统开机之后再执行**

#### anacron检测周期

* **anacron:会使用一天,七天,一个月作为检测周期**
* **在系统的/var/spool/anacron/目录中存在cron.{daily, weekly, monthly}文件,用于记录上次执行cron的时间**
* **和当前时间做比较,若果两个时间的差值超过了 anacron的指定时间差值,证明有cron任务被楼执行**

#### 不同版本间区别

* **在老的 Centos5和以前版本中,/ etc/cron{ daily, weekly, monthly}这些目录即会被cron调用,也会被 anacron调用,容易重复执行**
* **在 Centos6-7x中则只会被 anacron调用,避免了重复执行**
* **在 Cnetos6-7x中, anacron不再是服务,而是系统命令**

#### anacron配置文件

**[root@localhost ~]# vi /etc/anacrontab**

* **RANDOM DELAY=45**

**#最大随机延迟**

* **START HOURS RANGE=3-22**

**#anacron执行的间范围是3:00-22:00**

* **1 5 cron.daily nice run-parts /etc/cron.daily**
* **7 25 cron.weekly nice run-parts /etc/cron.weekly**
* **@monthly 45 cron.monthly nice run-parts /etc/cron.monthly**

**#天数 强制延迟(分) 工作名称 实际执行的分令**

#### cron.daily工作来说明执行过程

* **首先读取/var/spool/anacron/cron.daily中的上一次anacron执行的时间**
* **和当前时间比较,如果两个时间的差值超过1天,就执行cron.daily工作**
* **执行这个工作只能在03:00-2200之间**
* **执行工作时强制延迟时间为5分钟,再随机延迟0-45分钟时间**
* **使用nice命令指定默认优先级,使用 run-parts脚本执行/etc/cron.daily目录中的所有可执行文件**

# 写在最后

**致敬Linus Torvalds先生和为Linux作出贡献的所有组织和开发者。**

**致敬伟大的开源共享的精神。**

****

**Linus Torvalds：Linux 之旅既有趣又幸运，我不敢奢望精通内核的全部**

**https://www.linuxidc.com/Linux/2017-07/145456.htm**

**《Linux 系统管理的面试题》这个仓库收集了200多道服务器管理的面试题，涉及的面很广，着重考察基础知识:**

**https://github.com/trimstray/test-your-sysadmin-skills**