# 云计算过程

您可能需要再了解原理的情况下使用这个文档，因为有许多地方没有（或至少没有足够简洁地）指出这么写的原因。这个文件实际上帮了我很多。

## 网络（不要参考此处）

太古之初，有终端机。终端机功能强大，故人乐在其中。然独乐乐非真乐也！继而醒悟，疾需互联。

后来，网卡现，网线出，物理互联，不亦乐乎。

然多人互联，必添网卡，机箱过重，其能久乎！故而集线器、交换机横空出世，局域网继往开来。

人性也，有近有远，有喜有怖，vlan分区显神通。

计算机可以直接通过网线相连，其原理属于硬件设计，这里暂时不谈。

联网本质上是连接，通过网线将两台终端相连，这里终端一般是计算机和服务器。

用一根网线加上机箱内置的网卡就能让两台计算机联网，但只有两台计算机联网相对互联网来说格局还是太小了，所以需要想一种将更多计算机相连的方法。与此同时，机箱的容量的是有限的，为了方便个人使用，它的体积受到一定的限制，所以就需要一种专门用来连接多台终端的设备，它必须有许多接口，而且接口之间能够互联。人们据此造出了“交换机”。

交换机通过机器码（MAC码）分辨不同的终端，它用一种类似广播的泛洪方法将一段目的未知的信号发送给尽可能多的终端，当信号目的可辨识时，可以通过机器码转发到目的地，达到传送信息的目的。交换机会记住所有通过它的信号的源地址的机器码，在几分钟内不再因同样的信号触发泛洪，而是直接转发，这个是交换机的“学习”功能。

一个网段内连接的计算机一多，管理上的问题和机制本身的问题就不可忽视，人们设计了VLAN压制这个问题。为了方便管理，VLAN将连接在同一交换机的终端划分为多个分区，这些分区以数字命名，可以一次性管理一整个分区的终端的通断，而且为了避免大规模泛洪现象导致的风险，彼此之间信号并不互通。

用交换机实现小范围的网络互通以后，人们开始追求更大规模的联机，IP协议横空出世，它给每一台联网的计算机一个IP地址，在更大范围的网络连接中通过IP地址标识自身，而非MAC码。

##### 终端？

就计算机和服务器。或者叫计算机、服务器的网卡，连通的意义所在。

##### 交换机？

为了满足大量的终端互联，而制造出的可以插几十根网线、有这么多接口的设备。

##### 路由器？

拥有使不同网段ip互联的功能。至少有两个接口，用到的时候一出一入。

## Linux系统

### 系统部分

##### 根目录下各目录的作用

|  |  |
| --- | --- |
| 1 | bin #用户可执行目录 |
| 2 | sbin #系统可执行目录 |
| 3 | lib #32位库文件目录 |
| 4 | lib64 #64位库文件目录 |
| 5 | dev #设备文件目录 |
| 6 | usr #应用程序目录 |
| 7 | var #服务器数据目录，数据日志 |
| 8 | srv #服务器数据目录 |
| 9 | etc #配置文件目录 |
| 10 | tmp #临时文件目录 |
| 11 | boot #服务器启动目录 |
| 12 | media #媒介目录，U盘和光盘会在这里 |
| 13 | mnt #其他挂载点 |
| 14 | opt #第三方应用程序目录 |
| 15 | proc #伪文件系统（内核参数，进程信息，硬件信息） |
| 16 | sys #伪文件系统（配置文件目录、内核参数、进程信息、硬件信息） |
| 17 | run #进程锁目录 |
| 18 | root #root管理员目录 |
| 19 | home #普通用户家目录，普通用户登录终端一般在这 |

##### 7种文件类型

在ls的视角中，文件的类型会在它的左边第一位用一个符号表现，例如“-”是指普通文件，而“d”是指目录。

|  |  |
| --- | --- |
| 1 | - #普通文件——file |
| 2 | d #目录文件——directory |
| 3 | c #字符设备文件——character |
| 4 | b #块设备文件——block |
| 5 | s #套接字文件——socket |
| 6 | p #管道文件——pipe |
| 7 | l #符号链接文件——symbolic |

##### 重置root密码

重启，在选择进入哪个系统的时候按e。找到开头是linux16的一行，将ro后面的文字删去，然后在相同位置写上rd.break，之后ctrl+x启动虚拟操作系统，进入命令行模式。

|  |  |
| --- | --- |
| 1 | mount -o remount,rw /sysroot/ #由于虚拟操作系统的根目录是只读权限，所以这里要专门给根目录设置读写权限。 |
| 2 | chroot /sysroot #算是紧急模式的cd /，从虚拟操作系统进入真实的根目录。 |
| 3 | touch /.autorelabel #刷新selinux权限。 |
| 4 | echo ［新密码］ | passwd --stdin root |
| 5 | exit #退出真实根目录 |
| 6 | exit |

#### 进程与程序

进程与程序没有一一对应关系，因为一个程序在被执行过程中可能依次产生多个进程，另一方面一个进程在其生命周期的中间阶段也可能执行多个程序。

举例而言，终端是一个进程，在终端页面未关闭的时候，就是终端这个进程生命周期的中间阶段，在终端中使用命令就是在进程中执行程序。

##### 进程

一个程序的执行过程，一个程序在被执行的时候，就会产生一个进程，进程在程序中的指令执行完毕后就会消亡。

进程需要进行资源申请、会被系统调度，也能作为独立运行的单位，他会占用系统中的运行资源。

进程有父进程和子进程之分，子进程在父进程之下，杀死父进程会影响子进程，杀死子进程不会影响父进程。

##### 程序

一个静态的指令集合，它的存在本身不会占用系统的运行资源。在不被执行的时候，程序和没人读的文字一样无意义也无危害。

##### 大进程systemd

systemd是系统开启的第一个进程，也是所有进程的父进程。systemd无法被杀死，除非关机或重启。如果其他父进程被杀死，其下的子进程就会被systemd接收，这之后的子进程可能会继续生命周期，但是也有可能被重置或停止。

#### Unit 系统资源

Systemd可以管理所有系统资源，不同的资源统称为Unit。

unti共有12种类型：

|  |  |
| --- | --- |
| 1 | Service Unit #系统服务。 |
| 2 | Target Unit #多个Unit构成的一个组。 |
| 3 | Device Unit #硬件设备。 |
| 4 | Mount Unit #文件系统的挂载点。 |
| 5 | Automount Unit #自动挂载点。 |
| 6 | Path Unit #文件或路径。 |
| 7 | Scope Unit #不是Systend启动的外部进程。 |
| 8 | Slice Unit #进程组 |
| 9 | Snapshot Unit #Systemd快照，可以切回某个快照。 |
| 10 | Socket Uint #进程间通信的socket。 |
| 11 | Swap Unit #swap文件。 |
| 12 | Timer Unit #定时器。 |

#### 服务

服务（Service）是由守护进程（Daemon）提供的。

|  |  |
| --- | --- |
| Service服务 | Daemon守护进程 |
| ssh | sshd |
| web | httpd |
| dns | named |

##### 服务的类型

|  |  |
| --- | --- |
| 按功能分类 | 按安装方式分类 |
| 系统服务 | 使用rpm包安装的服务 |
| 网络服务 | 使用源码包编译安装的服务 |

##### 服务和端口的关系

启动一个服务都会启动一个或多个对应的端口，这些端口大多数是协议规定好的。

|  |  |
| --- | --- |
| 端口号 | 服务 |
| 80 | http |
| 23 | telnet |
| 22 | ssh |
| 21 | vsftpd |
| 53 | named（dns） |
| 67 | dhcp |
| 25 | smtp（mail） |

端口号可以通过/etc/services查看。

##### 动态和静态网页

静态网页和动态网页可能是比较过时的描述，现在通常都是混合的，一个网页内部同时有静态元素和动态元素。

静态元素通常是写死的、固定的，是纯粹的文本、图片或其他媒体。

动态元素会因为背后程序的变量变化而变化，它本质是个程序，而网页上显示的是程序的输出。

##### cookie令牌机制

客户端向服务器发信息，通常会得到服务器的回信，服务器给没见过的用户回信时总会生成一串用于识别客户端的随机数，在回信时一并回过去。这一串随机数就是cookie，cookie机制就是使用cookie识别客户端的机制。

客户端接收到含有cookie的回信后，往后给服务器发的信息都会带有这一串随机数，供服务器快速识别身份。

cookie被服务器生成，然后被服务器塞给客户端。

##### session会话

session会话是一种监测机制，表示web服务器会分配给来访用户一个专属的ID，并基于这个ID给用户提供个性化服务。网站给用户的ID储存在用户客户端的浏览器中，当客户端用同一个浏览器访问访问过的网站时，网站会根据ID确认用户的身份。

##### http服务

http服务是apachi发布的网络传输服务，可以用来浏览网页。

版本

最初的或许是可能的最初的0.9版本只支持纯文本数据的传输，从1.0开始，http可以传输非文本数据，例如图片、视频、音频、动画等，同时它还有缓存机制，这可能减少下次访问相同网站所需的加载时间。缓存分服务器缓存和客户端缓存，分别保存在服务器内存和客户端的浏览器。

从1.1开始，http加入了长连接机制和管道机制，前者允许在一条链接上发送多次请求，价值可能在省去了频繁连、断的麻烦。此前可能由于连接是一次性的，所以长链接机制也加上了超时时间和最大请求数两个子机制来限制资源占用。后者允许一次性发送多个请求、并行发送，就好像“粗链接机制”，加快了传输效率。1.1也优化了缓存的管理，这可能包括两个方面——缓存静态数据和过期时间。

从2.0版本开始，http将以并行的方式发送请求和回应。提高效率。

状态码

通过http协议向web网站发送请求时，通常服务器会向浏览器传回状态码。这些状态码一般只有3位，最开头的那一位断定它的性质。

2开头的状态码，如200，就是成功的信号。

3开头的状态码，如301/3023304，就是成功响应但因故而重定向的信号。

4开头的状态码是错误的含义，其中403是权限拒绝，404是找不到对应的网站。

5开头的状态码也是错误响应，一般是服务器端的配置错误。

请求方法

用户使用浏览器与网站交互时，浏览器自动应用一些请求方法，常见的比如GET获取内容，下载和POST上传等。一般不是由用户亲自输入，而是写在html或php文件里通过按钮或超链接实现的。

### 指令部分

#### 一般常用指令——————

##### 基本语法

Linux语法由三部分组成：

1 命令具体的功能

2 命令的扩展功能

3 参数

1. 语法：命令 选项 参数1 参数2 参数N

2

3 语法：命令 选项1 参数 选项2 参数 选项N 参数

##### su 切换用户

“su ［用户名］”和“su - ［用户名］”效果看似相同，实际上用“env”命令查看可以发现前者并没有将另一用户的全部信息都切换过来，只套了个皮，短期使用并不大碍，长期使用恐有祸患！

1 su – root 从当前登录的用户切换到root超级用户，可解锁更多权限。

2 su – 和上一条功能相等。

3 su – ［用户名］ 从当前登录的用户切换到目标用户。

4

5 exit 从当前登录的用户中登出。ctrl+d相同效果。

##### cd 进入目录

简单来说，可以用cd进入任意知道目的地的目录。cd可以在ls的帮助下找到下一步。

1 cd ~ #进入家目录

2 cd / #进入根目录

3 cd ［地址］ #进入目标目录

4 cd #进入家目录

##### ls 查看目录中的内容［1］

在终端的任意区域使用［ls］都可以列出当前目录下所有的目录和文件。

1 ls #查看当前目录下的所有内容。

2 ls ［目录名］ #查看目标目录下的可见内容。

3

4 -l #使信息以更详细的方式列出。

5 -a #查看目录下的隐藏文件。

6 -d #查看目录本身。

7 -A #查看包括隐藏文件在内的所有文件。

8 -h #显示文件大小并且按最大单位显示

9 -t #按时间排序（降序）

10 -r #逆序输出

11 -S #按大小排序（降序）

12 -R #递归显示

##### du 查看文件大小

du -h 用来查看文件占用空间的大小，而ls -h与之相似，但是用来查看文件本身的大小。

|  |  |
| --- | --- |
| 1 | du #显示当前目录下所有文件占用空间的大小，但不显示单位。 |
| 2 | -s #显示当前目录占用空间的大小。 |
| 3 | -h #显示大小的同时显示单位。 |

##### 查看文件

1.more［长］

1 more ［文件路径］ #浏览稍长文件使用，局内显示阅读百分比。

2

3 空格 #下一页

4 b #上一页

5 回车 #下一行

2.less［多页］

1 less ［文件路径］ #浏览多页文档时使用，可以翻页。

2

3 上键 #上一行

4 下键 #下一行

5 pageup #上一页

6 pagedown #下一页

7 q #退出

3.head［2］［头］

1 head ［文件路径］ #从文件头部开始，默认显示前十行。

2

3 -n ［行数］ #从头部开始显示特定行。

4 -［行数］ #同上

4.tail［2］［尾］

|  |  |
| --- | --- |
| 1 | tail ［文件路径］ #从文件末尾开始，默认显示后十行。 |
| 2 | -n ［行数］ #从尾部开始显示特定行。 |
| 3 | -［行数］ #同上 |
|  |  |
| 4 | tail -n 0 -f /var/log/messages #查看动态监控日志文件新更新的部分 |

/var/log/messages是动态监控日志文件，它的作用是记录系统启动、关闭、网络、文件系统、安全、硬件和错误信息等内容。通过-n 0，可以查询

5.cat［1］

1 cat ［文件路径］ #查看较短的文件，直白地列出。

2

3 -n #列出每一行的行号，包括空行。

4 -b #同上，但不列出空行的行号。

这些查看指令都可以用“|”互相嵌套。例如：

Less ［文件名］| head -100 | tail -30 | cat -n

##### touch 创建新文件

创建空的新文件。

1 touch ［文件路径1］［文件路径2］ … #创建新的空文件。

2

3 touch ［{第一部分}{数字1..数字2}.扩展名］ #创建以{第一部分}和连续数字命名的多个文件，适用于批量创建连续的文件。

5

6 touch ［{第一部分1,第一部分2,第一部分3}{数字1..数字2}.扩展名］ #创建以第一部分和连续数字部分的每一种组合的文件。

以下为部分举例：

1 touch a.txt b.txt c.txt #创建a.txt，b.txt，c.txt，参考第1行。

2

3 touch a{1..5}.txt #创建a1.txt，a2.txt，a3.txt，a4.txt，a5.txt，参考第3行。

5

6 touch {a,b}{2..3}.txt #创建a2.txt，a3.txt，b2.txt，b3.txt，参考第6行。

##### mkdir 创建目录［1］

mkdir相当于创造目录版的touch，创建多个目录的规则可以完全参考touch的举例。

1 mkdir ［目录路径］ #创建空目录。

2

3 -p #递归创建。假如要在一个不存在的目录中创建目录，使用这个选项可以先创建那个不存在的目录，从而实现目的、节省时间。

4

5 -v #显示创建目录的过程。

##### rmdir 删除空目录

很可能不常用。但或许有用。

1 rmdir ［目录路径］ #删除空目录。

##### rm 删除文件和目录［1］

1 rm ［文件名］ #删除空文件和普通文件（有内容的文件）。

2

3 -f #强制删除，不会进一步询问。

4 -r #递归删除，可以用来删除目录。

可以用“\*”表示多个字符用于批量删除文件和目录，例如rm -rf /\*可以干掉整个根目录。

##### cp 拷贝命令

1 cp ［文件1］ ［文件2］ ［目标路径］ #将文件复制到目标路径。

2

3 -r #递归复制，用于复制目录。

4 -p #保留被复制文件的权限。

5 -a #尽可能保持文件的结构和属性。相当于-dpR。

##### mv 剪切命令

1 mv ［文件］ ［目标路径］ #将文件移动到目标路径，如果最后一个参数为本目录的另一个文件名，则视为重命名。

批量剪切时，［文件］部分可以写星号\*。如：mv \* ［目标路径］。如果是将全部文件批量剪切到同目录下的某个目录里，那么系统提示无法移动某个目录是正常的。

##### vim文本编辑

用来编辑文件。有输入模式、命令模式和末行模式。其中输入模式可以修改文件内字符，命令模式可以输入快捷键方便跳行、跳页查阅，末行模式可以对文件发出各种指令。

|  |  |
| --- | --- |
| 1 | vim ［文件名］ #进入一个文件并进入命令模式。 |
| 2 | -o #以横排模式同时显示多个文件。 |
| 3 | -O #以竖排模式同时显示多个文件。 |
| 4 | a #从命令模式进入输入模式。 |
| 5 | esc #esc键。从输入模式和末行模式退出到命令模式。 |
| 6 | shift + : #从命令模式进入末行模式。 |

在同时打开多个窗口的时候，可以用ctrl+ww，在不同窗口间移动光标。

命令模式

1 a #在当前字符后一位开始输入。

2 A #在当前行的行尾开始输入。

3 i #在当前字符开始输入。

4 I #在当前行的行首开始输入。

5 o #在当前行的下一行开始输入，并空一行。

6 O #在当前行的上一行开始输入，并空一行。

7 s #删除光标所在的字符后进入输入状态。

8 S #删除光标所在的一行后进入输入状态。

定位

1 ↑↓←→ #控制光标上下左右移动。可以用数字+输入单键的方式跳过指定数目的字符。

3

4 hjkl #和箭头功能相似，分别为左上下右。

5 0 #将光标定位到行首。同功能：^。

6 $ #将光标定位到行尾。

7 gg #将光标定位到文章首部。

8 G #将光标定位到文章尾部。

9 ［数字］G #将光标定位到文章的指定行。

10 /［内容］ #从上往下搜索指定内容。n，下一个；N，上一个。

11 ?［内容］ #从下往上搜索指定内容。

操作

1 y #复制。

2 yw #复制一个单词。

3 yy #复制一行。［数字］yy可以复制多行。

4 p #粘贴在下一行。

5 P #粘贴在上一行。

6 u #撤回一步。

7 d #删除。

8 dw #删除一个单词。

9 dd #剪切。或者以剪切的方式删除，主要取决于按不按p/P粘贴。

10 ［数字］dd #剪切/删除指定行。

11 d^ #从光标处删到本行行首，同功能指令：d0。

12 D #从光标处删到本行行尾，同功能指令：d$。

13 dgg #从当前行删到首行。

14 dG #从当前行删到尾行。

15 . #重复上一步。

末行模式

1 w #保存。w!强制保存。

2 q #退出。q!强制退出。

3 wq #保存退出。wq!强制保存退出。

4 w ［文件名］ #另存为。

5 x #退出，如果修改过，则保存退出。

6 r ［文件名］ #将［文件名］的内容读取到当前文件。

7 ! ［linux命令］ #临时输入linux命令，输命令后临时跳到终端界面。

8 e ［文件名］ #打开另一个文件。

9 e! #重新打开当前文件。这是撤销保存前所有操作的最快方法之一。

10 X ［密码］ #给当前文件设密码，如果X后面不填，则视为删除密码。

环境设置，这些可以写在配置文件的最末尾保证自动、默认生效，也可以在vim的末行模式中进行临时设定。单用户的配置文件在~/.vimrc，全局的配置文件在/etc/vimrc。

1 set all

2 set nu #显示行号。

3 set nonu #隐藏行号。

4 set list #显示空格和制表符。

5 set nolist #隐藏空格和制表符。

6 set tabstop=［数字］ #设置文件内所有的制表位的宽度。

7 set softtabstop=［数字］ #设置此命令发布以后的制表位的宽度。

8 set autoindent #设置自动缩进。

9 set noautoindent #关闭自动缩进。

10 set ignorecase smartcase #使搜索功能忽略大小写。

替换功能：

|  |  |
| --- | --- |
| 1 | ［前缀］/［旧字符］/［新字符］/［后缀］ #替换的基本语法 |
| 2 | 前缀： |
| 3 | s #只输入前缀s，表示替换操作对当前行的第一个匹配字符生效。 |
| 4 | ［数字］s 只输入前缀［数字］s，表示替换第［数字］行的第一个匹配字符。 |
| 5 | ［数字1］，［数字2］s #只输入前缀［数字1］,［数字2］s，表示替换第［数字1］行到第［数字2］行的第一个匹配字符。 |
| 6 | %s #只输入前缀%s，表示替换操作对全文的第一个匹配字符生效。 |
| 7 | 后缀： |
| 8 | g #表示将替换操作应用在该行的每一个字符。 |
| 9 | c #交互式地替换字符。 |
| 10 | 注意： |
| 11 | @ #在vim的替换命令中，@可以匹配任何字符，它不仅可以当做“\*”用，也可以在需要替换文本中的“/”的时候，代替“/”打出。 |

例如：%s@/@n@g命令中，用@代替了/的功能，将全文的/替换成了n。在这个命令中，@和/不能同时具有一样的功能。

vimdiff 文件比较

|  |  |
| --- | --- |
| 1 | vimdiff ［文件名1］ ［文件名2］ #比较两个文件。这里可以比较更多文件但是我只写了两个。 |

wqa 全部保存并退出

在末行模式输入wqa可以保存并退出目前打开的所有文件。

#### 用户——————

##### useradd 添加用户［2］

在系统中添加一个新用户，用户配置文件在/etc/passwd。

1. useradd ［用户名］ 添加一个名为［用户名］的用户。

它有一系列选项和参数如下：

1. -u ［uid］ #UID选项，选项后接用户ID，取1000以上的任意值。新的用户ID不能和已有的重复。

3

4 -g ［gid］ #主要组选项，选项后接组ID，表示将新建用户归入选定的组，如果选定的组不存在，则自动生成一个组。

6

7 -c ［comment］ #［描述］中填入无关紧要的用户描述，一般来说是空的。

9

10 -d ［home］ #用户目录选项，［home］中填入用户目录名，即创建一个同名的用户目录。默认生成的用户目录名字与用户名相同。

12

13 -s ［shell］ #有两个主要参数，/bin/bash为默认参数，可以登录；/bin/nobash为非交互账号的参数，不能登录。非交互式还有/sbin/nologin，此外无权访问的/bin/false。

15

16 -G ［gid1,gid2,gid3］ #附加组。参数中填入已有的组的组名，表示将用户加入其他组中。

18

19 -o #与-u同时使用，允许新用户和其他用户共用一个uid。

##### usermod 修改用户［2］

修改已有用户的信息。

|  |  |
| --- | --- |
| 1 | usermod ［用户名］ #修改用户名为［用户名］的用户（这里没效果）。 |
| 2 | -u #修改uid。linux中，以uid的数字决定用户的权限，其中0是root权限。 |
| 3 | -g #修改所属主要组。 |
| 4 | -c #修改用户描述。 |
| 5 | -d #修改用户目录（一般不用，因为该新的目录不会自动创建，需要自行创建）。 |
| 6 | -s #修改shell参数。 |
| 7 | -G #修改附加组。 |
| 8 | -aG #添加附加组。 |
| 9 | -l #修改用户名。 |
| 10 | -L #锁定账号，使之不能被普通用户转登或直接登录。 |
| 11 | -U #解锁账号。 |

##### userdel 删除用户［1］

1. userdel ［用户名］ #删除用户（只删除用户，不常用）。

2

3 -r #删除用户，同时删除用户目录和邮箱（常用）。

##### passwd 修改用户密码

密码保存文件：/etc/shadow。

|  |  |
| --- | --- |
| 1 | passwd #交互式地改变当前用户的密码，需要连输两次新密码。 |
| 2 | passwd ［用户名］ #交互式地为选定用户修改密码。 |
| 3 | passwd –stdin ［用户名］ #用标准输入为选定用户修改密码，这个选项必须指定用户名。 |
| 4 | echo ［新密码］ | passwd --stdin ［用户名］ #用echo（一种标准输出）和管道符“|”作标准输入修改选定用户的密码。这是非交互式的。 |

除了修改密码，passwd还有如下应用：

|  |  |
| --- | --- |
| 1 | passwd -l ［用户名］ #锁定用户的密码，目标用户不能再通过密码登录。 |
| 2 | -S #输出目标用户账户密码的状态。 |
| 3 | -u #解锁目标用户的密码。和usermod -l的作用是一样的。 |

##### groupadd 添加组［2］

1 groupadd ［组名］ #添加名为［组名］的新组。

2

3 -g ［gid］ #添加自定义组ID的组。

##### groupmod 修改组［2］

［组］的面板比［用户］简单一些。

1 groupmod ［组名］ #修改名为［组名］的组（无效果）。

2

3 -g ［gid］ #修改组ID。

4 -n ［组名］ #修改组名。

##### groupdel 删除组

1 groupdel ［组名］ #删除组名为［组名］的组。

##### gpasswd 修改组密码及其他［2］

1 gpasswd ［组名］ #修改［组名］的组密码。

2

3 -a ［用户名］ #将用户加入到组中。

4 -d ［用户名］ #将用户从组中删除。

#### 安装软件包——————

##### rpm 安装软件/查看安装情况［1］

rpm包是红帽公司开发维护的软件包管理工具，安装软件时，需要提前解决依赖项。rpm包名的结构如下：

包名-版本.系统版本.平台.rpm

nmap-6.40-7.el7.x86\_64.rpm

rpm包默认安装路径

|  |  |
| --- | --- |
| 1 | /etc/ #配置文件安装目录。 |
| 2 | /usr/bin/ #可执行的命令安装目录。 |
| 3 | /usr/lib/ #程序所使用的函数库保存装置。 |
| 4 | /usr/share/doc/ #基本的软件使用手册保存位置。 |
| 5 | /usr/share/man/ #帮助文件保存位置。 |

软件安装和查询

安装和删除：

|  |  |
| --- | --- |
| 1 | rpm ［选项］ ［软件包全名］ #rpm语法。如果通过iso镜像安装软件应从光盘的挂载点寻找软件包。 |
| 2 | -i #安装软件。 |
| 3 | -v #显示安装过程。 |
| 4 | -h #显示安装百分比。 |
| 5 | -e #卸载软件。 |
| 6 | --nodeps #安装和卸载时忽略依赖。有可能导致安装的软件功能不全和卸载残留。 |
| 7 | --import #导入密钥专用的选项。 |

关于--import，常用的指令是：rpm --import ［公钥文件名］，即导入公钥，关于公钥文件的位置，可以用指令cd/;find / -name \*KEY\*查找到。除此之外，公钥也可以在服务器端找到，此时用

查询：

|  |  |
| --- | --- |
| 1 | rpm -q ［软件包名］ #查询软件是否安装。 |
| 2 | -qi #将效果改为查询软件包的信息。 |
| 3 | -ql #将效果改为查询安装的位置，包括软件包所有文件的安装位置。 |
| 4 | -qc #将效果改为查看配置文件。 |
| 5 | -qd #将效果改为查看帮助文档的位置。 |
| 6 | rpm -qa #查询已安装的软件包，在绝多数情况下都会列出一大条信息，所以一般会用第7行的精准查询。 |
| 7 | rpm -qa | grep ［软件包名］ #查询已安装的软件。 |
| 8 | rpm -qf ［文件名］ #查询文件所属的软件包。 |

关于上文第8行的查询方法，可以用which ［软件包名］找到一个软件包的可执行文件的路径（这是个学习方法，实际上多此一举）。

##### yum 批量安装软件包

yum需要配置仓库和源文件之后才能正常使用。其中yum仓库一般挂载到/mnt/cdrom，源文件一般放在/etc/yum.repos.d/目录下，其文件扩展名为.repo。

yum可以有多个源文件，但这些源文件的配置必须都是规范、有效的，否则一个出了问题其他的都无法使用。

挂载仓库（非本地源无需挂载仓库）

|  |  |
| --- | --- |
| 1 | 首先将光盘（或镜像）放入系统。 |
| 2 | df -h #检查系统中的光盘名称和位置，光盘这类设备进入系统后都是自动挂载到某个空目录里的，如果没有自动挂载，需要手动找到光盘的位置。 |
| 3 | umount ［光盘名］ #解除光盘自动挂载的关系，以便重新挂载。 |
| 4 | mkdir /mnt/cdrom #创建目标挂载点的目录。 |
| 5 | mount ［光盘名］ /mnt/cdrom #将光盘挂载到/mnt/cdrom目录，完成。 |

配置源文件

1 cd /etc/yum.repos.d/ #抵达源文件目录。

2 mkidr ［目录名］ #在源文件目录内创建新目录。

3 mv \* ［目录名］/ #将源文件目录中所有的文件都移动到新目录内。

4

5 vim iso.repo #在源文件目录内创建名为iso的源文件并进入编辑状态。

源文件内的配置内容：

1 [［池名称］] #表示软件源的名称。

2 name=［描述］ #软件源的描述，不重要。

3 baseurl=［仓库地址］ #表示仓库的地址。其中本地仓库以file://开头，网络仓库以http://开头。

5

6 enabled=1 #表示开启本仓库，默认（不写）为开启。设0为关闭。

7 gpgcheck=0 #设定为无需密钥。设1为需要密钥。其中本地源一般不需要密钥，网络源需要密钥。

以上第三行的举例为：Baseurl=file:///mnt/cdrom。

测试

|  |  |
| --- | --- |
| 1 | yum clean all #清除yum缓存。 |
| 2 | yum makecache fast #建立快速缓存。测试设置是否成功。 |
| 3 | yum repolist all #检查仓库是否可用。 |

常用指令

|  |  |
| --- | --- |
| 1 | yum search ［软件包名1］ ［软件包名2］ #列出所有与输入软件包名有关的软件包。 |
| 2 | yum install ［软件包名1］ ［软件包名2］ #安装列出的软件包。 |
| 3 | yum reinstall ［软件包名］ #重装软件包。 |
| 4 | yum provides ［软件包名］ #查看已安装软件包有无新版本。 |
| 5 | yum remove ［软件包名］ #卸载软件。 |
| 6 | yum info ［软件包名］ #显示可安装和已安装的软件包的详细信息。 |
| 7 | yum list ［软件包名］ #显示可安装和已安装的软件包的简短信息。 |
| 8 | yum upgrade ［软件包名］ #更新软件 |
| 9 | -y #在安装的过程中自动选择y。 |

安装软件包组

|  |  |
| --- | --- |
| 1 | yum group list #列出可安装的软件包组，包括环境包组和其他包组。跟安装系统时可以选择安装的系统软件包组是同一类的。 |
| 2 | yum groups info ［'软件包组名’］ #查看一个软件包组里面含有哪些软件。 |
| 3 | yum groups install ［'软件包组名’］ #安装软件包组。 |

常用软件（常测软件）

|  |  |
| --- | --- |
| 1 | dhcp-server #dhcp服务。 |
| 2 | wireshark\* #抓包工具。 |
| 3 | autofs #实现自动挂载。 |
| 4 | bind #dns服务。 |
| 5 | httpd #web服务。 |
| 6 | mariadb-server\* #数据库服务。 |
| 7 | vsftp #文件传输服务器。 |

##### 下载网络yum源

腾讯CentOS7源

|  |  |
| --- | --- |
| 1 | wget -O /etc/yum.repos.d/CentOS-Base.repo 3 |

阿里CentOS7源

|  |  |
| --- | --- |
| 1 | wget -O /etc/yum.repos.d/CentOS-Base.repo https://mirrors.aliyun.com/repo/Centos-7.repo |

或

|  |  |
| --- | --- |
| 1 | curl -o /etc/yum.repos.d/CentOS-Base.repo https://mirrors.aliyun.com/repo/Centos-7.repo |

华为CentOS7源

|  |  |
| --- | --- |
| 1 | wget -O /etc/yum.repos.d/CentOS-Base.repo https://mirrors.huaweicloud.com/repository/conf/CentOS-7-anon.repo |

配置网络镜像站

流程如下：

|  |  |
| --- | --- |
| 1 | 虚拟机联外网。 |
| 2 | 找到网络上的开源镜像站地址。 |
| 3 | 配置yum源文件，注意地址和密钥（gpgcheck=1，gpgkey=密钥）。 |

配置网络镜像站和配置本地仓库只有一点点区别，就是gpgcheck不为0，且需要写上gpgkey行，值为密钥的地址。

##### yum工具常见错误

yum源无法解析

由于yum工具没有足够的缓存（记忆）挨个测试能用的yum仓库，所以它会在遇到不可用的仓库时选择失败。有几种方法可以解决问题。

|  |  |
| --- | --- |
| 1 | 练习软件仓库的上游，即软件仓库的更上层管理者，让他们解决这个问题。如果是自己练习的时候遇到了这个错误，就使用其他的解决方案。 |
| 2 | 重新配置软件仓库（yum源文件）的baseurl，使其指向正常工作的仓库。 |
| 3 | 使用临时禁用软件仓库的命令使其暂时不再碍事。  yum --disable-repo=［需要禁用的yum源文件］ |
| 4 | 永久禁用方案！选其一。  yum-config-manager --disable ［yum源文件］  subscription-manager repos --disable=［yum源文件］ |
| 5 | 使用命令，使其遇到不能用的就跳过。但一般而言会导致yum运行缓慢，是个可用的折中选项。  yum-config-manager --save --setopt=iso.skip\_if\_unavailable=true |

pid已被锁定

因为之前的yum任务没有完成，所以导致无法使用后来的yum指令。

|  |  |
| --- | --- |
| 1 | rm -f /run/yum.pid #删除/run/yum.pid。 |
| 2 | yum clean all #清除缓存。 |

##### sudo yum 带验证的yum

在root用户的操作中，sudo yum和yum没有任何不同，但是在普通用户的操作中，sudo yum有一个需要输入密码的执行过程，以确保程序有足够的权限执行命令。

常规yum指令前面加上sudo，就是sudo yum的用法。

安装扩展源

|  |  |
| --- | --- |
| 1 | sudo yum -y install epel-release #安装扩展源 |
| 2 | wget -O /etc/yum.repos.d/epel-7.repo http://mirrors.aliyun.com/repo/epel-7.repo |

##### 源码包安装

从别处获得现有的源码包并将其拖入系统，我忘了记这个了。

在不参考任何资料的情况下，猜测它是用来安装一些不在任何yum源内的软件的。

#### 文件操作——————

##### chmod 修改文件权限

linux的文件系统共有三种（基础的）权限，其中r可以查看文件的内容，w可以修改或删除文件，x可以进入目录或者执行文件。

又有四种身份，其中有root超级用户，所有者，组内人，其他人。超级用户默认具有完全权限，不受权限体系约束；所有者默认有读写权限。如果一个用户属于一个文件的所有者和组内人，只能使用所有者的权限。

|  |  |  |  |
| --- | --- | --- | --- |
| u | 所有者 | r | 读4 |
| g | 所属组 | w | 写2 |
| o | 其他人 | x | 执行1 |

假设全都为无权限，那么……

|  |  |  |  |
| --- | --- | --- | --- |
| 字符方法 | 数字方法 | 字符方法 | 数字方法 |
| u+r | 400 | u+x,g+w | 120 |
| u+rw | 600 | g+x,o+r | 014 |
| u+rwx | 700 | u+rw,g+wx,o+r | 634 |

|  |  |
| --- | --- |
| 1 | chmod ［数字1数字2数字3］ ［文件名］ #修改文件权限。 |
| 2 | chmod ［字符］［+、-或=］［权限］ ［文件名］ #修改文件权限，等号在这里也是赋值的意思。 |

用字符方法修改文件权限也可以一次性修改多个对象的权限，这需要写完一组之后用逗号隔开再写下一组，假如多个目标的权限有共性，那么可以一起设定，例如ug=wx就是把拥有者和所属组权限都改成可写和可执行。

花里胡哨的用法也是有效的，例如：chmod ugo=r,u=rw,o=x 11，虽然确实没有什么用（11是文件名）。

字符方法的等号右边填“-”、空格或者不写参数直接写下一项时，表示无权限，相当于数字方法的0。

|  |  |
| --- | --- |
| 1 | chmod ［字符］=- ［文件名］ #为特定文件设定权限，并剥除特定目标的所有权限。 |
| 2 | chmod ［字符］= ［文件名］ #同上。 |
| 3 | chmod ［字符］= ［文件名］ #同上。和上一个指令相比，等号后少一个空格。 |

根据以上语法加入特殊权限……

|  |  |  |  |
| --- | --- | --- | --- |
| suid | u+s | 4\*\*\* | 任何用户在运行拥有这个权限的命令时，都以该命令的拥有者的身份执行，当拥有者的身份特殊时，例如在拥有者为超级用户的情况下，可以越过一般权限的限制。 |
| sgid | g+s | 2\*\*\* | 任何用户在拥有这个权限的目录下创建的文件和目录都自动继承父目录的所属组。一般用来省去重复为新文件修改所属组的麻烦。 |
| sbit | o+t | 1\*\*\* | 只有该文件的创建者可以删除拥有该权限的目录下的文件，避免文件随便被别人删。 |

指令见下：

|  |  |
| --- | --- |
| 1 | chmod u+s ［文件名］ #为文件设置suid权限，在ll返回的信息里，表示所有者执行权限的位置由s替代。 |
| 2 | chmod g+s ［目录名］ #为目录设置sgid权限，在ll返回的信息里，表示所属组执行权限的位置由s替代。 |
| 3 | chmod o+t ［目录名］ #为目录设置sbit权限，在ll返回的信息里，表示其他人执行权限的位置由t替代。 |

##### chown/chgrp 修改文件所有者和所属组

chown和chgrp的目的都是一样的，前者是后者的高级版，只允许超级管理员（root）使用，后者一般由普通用户使用。

1 chown ［用户名］ ［文件名］ #修改文件的拥有者为［用户名］。

2 chown ［:组名］ ［文件名］ #修改文件的所属组为［组名］。

3 chown ［用户名］［:组名］ ［文件名］ #同时修改文件的拥有者和所属组。

5

6 chgrp ［组名］ ［文件名］ #修改文件的所属组为［组名］。

7

8 -R #递归修改。假如指定修改一个目录，将一次性将目录内所有文件都改变所有者/所属组。

##### umask 权限反掩码

用户的umask值决定了所创建的文件的初始权限，其中超级用户root的umask值是0022，普通用户的umask值是0002。**umask值是八进制数。**

|  |  |  |
| --- | --- | --- |
| 用户类型 | 文件初始权限 | 目录初始权限 |
| root用户0022 | 775 | 755 |
| 其他用户0002 | 664 | 644 |

关于umask的指令：

|  |  |
| --- | --- |
| 1 | umask #查看当前用户的umask值。 |
| 2 | umask ［数字］ #修改当前用户的umask值。 |
| 3 | umask -S #查看当前umask值对目录的权限。 |
| 4 | ehco ‘umask ［umask值］’ >> ~/.bashrc #给当前用户设置永久uamsk值。 |
| 5 | ehco ‘umask ［umask值］’ >> /etc/bashrc #给所有用户设置永久uamsk值。 |

umask值决定初始权限的原理是目录/文件最大权限值的二进制（用数字方法表示的）根据umask值后三位的二进制取反，如下图所示：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ur | uw | ux | gr | gw | gx | or | ow | ox |
| 目录权限最大值 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| umask值0022 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 目录初始权限 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |

文件初始权限：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ur | uw | ux | gr | gw | gx | or | ow | ox |
| 文件权限最大值 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| umask值0022 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 文件初始权限 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

因此，定制需要的umask值只要根据上面的表格，先修改结果，再根据权限最大值反推umask值。可能得凹一段时间。

##### setfacl acl命令管理

文件成功设置acl权限之后会在ls命令给出的信息里显示“+”，就在一开始显示权限的区域的后方。与此同时，修改设置acl权限会导致mask值跟着变化并占用［所属组］的显示区域，之后就不适合用ls看文件的所属组权限。

可以在指令的末尾通过“>”传输acl权限。

|  |  |
| --- | --- |
| 1 | setfacl ［选项］ ［对象］:［对象名］:［权限］ ［文件名］ #对一个文件进行acl权限管理。 |
| 2 | -m #设置acl权限。 |
| 3 | -R #递归设置。 |
| 4 | setfacl -m o:［权限］ ［文件名］ #修改其他人（o）的acl权限，因为没什么用所以很少用。 |
| 5 | setfacl -x ［对象］:［对象名］ ［文件名］ #删除文件的一个acl权限。 |
| 7 | setfacl -b ［文件名］ #删除文件的所有acl权限。 |

由于修改文件的acl权限会一并修改文件权限的mask值，所以只删除设置的部分仍然会在ll返回的信息里留下“+”，必须一并删去mask值才能使加号消失。

对象类型如下：

|  |  |
| --- | --- |
| 1 | u #用户。 |
| 2 | g #所属组。 |
| 3 | m #mask值。 |
| 4 | o #其他人。 |

##### getfacl 查看acl状态

1 getfacl ［文件］ #查看该文件的acl状态。

2

3 -R #递归查询，查看目录中所有子文件和目录的权限。

##### acl的其他用法

有时设置好的acl权限会因为各种原因消失，造成效率上的损失。

|  |  |
| --- | --- |
| 1 | getfacl ［文件名］ > ［文件名］ #将文件的acl权限写入到另一个文件内部，如果后期不更改，这个设置会一直保留。 |
| 2 | setfacl --restore=［文件名］ #使文件中储存的acl权限，只能恢复同一路径的文件。 |
| 3 | setfacl --set-file=［文件名］ ［文件名］ #同上，但是不限路径。 |

通过以上方法备份并恢复权限。

##### lsattr 查看隐藏权限

|  |  |
| --- | --- |
| 1 | lsattr ［文件名］ #查看文件的隐藏权限。 |

##### chattr 设置扩展属性

|  |  |
| --- | --- |
| 1 | chattr ［+，-或=］［权限］ ［文件名］ #给文件设置隐藏权限。 |

大多数隐藏权限只能由root用户创建和移除，它可以限制root用户的操作。

|  |  |
| --- | --- |
| 1 | i #既不能删除它，也不能给它重新命名，不能对该文件创建链接，而且也不能对该文件写入任何数据。 |
| 2 | a #适用于日志类文件，只能在其中追加内容而不能删除或修改内容。 |
| 3 | d #适用于不需要备份的文件，有这个属性的文件不能被备份。 |
| 4 | u #可以修改，但不能被删除。 |

##### 压缩和解压缩

后缀名为“.gz”、“.bz2”、“.xz”的文件会在ls的回复中显示红色，它不一定是压缩文件。同理，是压缩文件但是后缀不是以上的文件不会在ls的回复中显示红色。file可以检查一个文件是否压缩文件。

gzip 压缩

|  |  |
| --- | --- |
| 1 | gzip ［文件名］ #压缩一个文件并为其加上后缀“.gz”。 |
| 2 | gzip -d ［压缩文件夹］ #解压缩已经压缩的文件。 |
| 3 | gzip -c ［文件名］ > ［压缩文件名］ #压缩并保留源文件，同时指定压缩文件名1。 |
| 4 | cat ［文件名］ | gzip > ［压缩文件名］ #压缩并保留源文件，同时指定压缩文件名2。 |

使用file查看被压缩的文件可以看到gzip compressed data字样，说明它是个压缩文件。

bzip2 压缩

|  |  |
| --- | --- |
| 1 | bzip2 ［文件名］ #压缩一个文件并为其加上后缀“.bz2”。 |
| 2 | -d #解压缩。 |
| 3 | -k #保留源文件。 |

使用file查看被压缩的文件可以看到bzip2 compressed data字样，说明它是个压缩文件。

其他解压缩方法：

|  |  |
| --- | --- |
| 1 | bunzip2 ［压缩文件名］ #解压缩。 |

xz 压缩

|  |  |
| --- | --- |
| 1 | xz ［文件名］ #压缩一个文件并为其加上后缀“.xz”。 |
| 2 | -d #解压缩。 |

使用file查看被压缩的文件可以看到XZ compressed data字样，说明它是个压缩文件。

##### tar 打包

需要注意语法，可能造成麻烦。

|  |  |
| --- | --- |
| 1 | tar ［选项］ ［新建包名］ ［被打包文件］ #tar语法。 |
| 2 | -c #打包并保留源文件。打包时必加。 |
| 3 | -v #显示过程。 |
| 4 | -f #指定文件名。必加。 |
| 5 | -t #查看打包文件内容。 |
| 6 | -r #为打包文件追加内容。想要为一个包追加内容又懒得解包时替代-c使用。 |

［新建包名］中，如果包名中包含目录名，如“目录名/包名”或“/目录名/包名”，则表示将打包后的文件储存到对应目录中。

［被打包文件］可表示多个被打包文件，以空格分开。除了文件，［被打包文件］可以写目录名，即为打包目录。

|  |  |
| --- | --- |
| 1 | tar ［选项］ ［包名］ #解包语法。 |
| 2 | -x #解包。解包时必加。 |
| 3 | tar -xf ［包名］ -C ［目录名］ #将包内容解到指定目录下。 |

打包并压缩，以及对于“压缩包”的处理：

|  |  |
| --- | --- |
| 1 | -z #打包并用gzip压缩，或解通过gzip压缩的压缩包为文件。对gzip压缩包使用tar操作时必加。 |
| 2 | -j #打包并用bzip2压缩，或解通过bzip2压缩的压缩包为文件。对bzip2压缩包使用tar操作时必加。 |
| 3 | -J #打包并用xz压缩，或解通过xz压缩的压缩包为文件。对xz压缩包使用tar操作时必加。 |

这只是压缩包文件的一种方法——打包的同时压缩，以及解压缩的同时解包。也可以先打包再用压缩工具压缩。

##### ln 链接文件

就是快捷方式。

|  |  |
| --- | --- |
| 1 | ln -s ［文件名/目录名］ ［链接文件名/链接目录名］ #创建文件或目录的软链接文件。 |
| 2 | ln ［文件名］ ［链接文件名］ #创建文件的硬链接文件。 |
| 3 | unlink ［文件名/链接文件名］ #解除链接。对于链接关系的源文件和链接文件来说，它的作用约等于rm -f。 |

软链接文件在ls的界面显示青色，用file检查会有symbolic link to `xxx'字样。如果将它所链接的文件删除，它就会在ls界面显示成黑底红字，并且失去作用，用file检查会有broken symbolic link to `xxx'字样。如果在其中放入数据，它就会重新生成一个曾经链接的文件并恢复可用（但不会恢复曾被删除的源文件的数据）。

硬链接文件相当于和源文件共享数据的副本，删除源文件或是硬链接文件的其中一个不会丢失数据。硬链接文件和源文件在du界面只显示其中一个。

软链接文件的权限总是777，硬链接文件的权限总是与源文件相同。

##### find 查找文件

一些查找逻辑复杂的find指令可能需要打括号，需要注意在指令中大括号要用转义符号辅助。

|  |  |
| --- | --- |
| 1 | find ［路径］［选项］ ［值］ #查找特定路径中，系统中的文件。和大多数指令一样，可以同时选择多个选项，而且很常用。 |
| 2 | -name ［文件名］ #通过文件名查找。 |
| 3 | -user ［用户名］ #通过所有者查找。 |
| 4 | -group ［组名］ #通过所属组查找。 |
| 5 | -type ［文件类型］ #根据文件类型查找。 |
| 6 | -links ［硬链接数］ #根据硬链接数查找。 |
| 7 | -size ［文件大小］ #精确查找特定文件大小的文件。一次性写两个或更多-size选项表示取区间。 |
| 8 | -size +［文件大小］ #查找大于某个大小的文件。 |
| 9 | -size -［文件大小］ #查找小于某个大小的文件。 |
| 10 | -mtime ［天数］ #查找正好在多少天前修改过的文件。 |
| 11 | -mtime +［天数］ #查找在多少天之前或更久以前修改过的文件。 |
| 12 | -mtime -［天数］ #查找在多少天以内修改过的文件。 |
| 13 | -perm ［权限数字］ #完全匹配，只有与查找权限完全相同的文件能够满足条件。 |
| 14 | -perm /［权限数字］ #部分匹配，只要被搜索文件能够大于或等于ugo任一部分的权限就能满足条件。 |
| 15 | -perm -［权限数字］ #子集匹配，查找条件的标0部分可以匹配任何权限，非0部分视为部分匹配。 |

其他：

|  |  |
| --- | --- |
| 1 | -exec ［命令］ \; #在查找之后执行命令。-exec后面需要写一个完整的命令，命令中表示被查找的文件的选项用{}替代，末尾用/;表示-exec选项的结束。 |
| 2 | -ok #同上，但是会在执行前询问。 |
| 3 | -ls #在查找到文件后使用ls -l查看文件详细信息。 |

-exec的举例：find /etc/ -type f -exec mv {} /etc \;

注意如果exec后接cp的话，最好加-a，不然会更改文件的所有者和所属组，不易查查找结果。

逻辑：

|  |  |
| --- | --- |
| 1 | -a #写在两个选项之间，与，只有两个条件同时生效的文件才会被匹配。 |
| 2 | -o #写在两个选项之间，或，只要两个条件的七种一个生效的文件就会被匹配。 |
| 3 | -not #表示跟置于其后方的选项反着来。 |

文件类型：

|  |  |
| --- | --- |
| 1 | b #块设备文件。 |
| 2 | d #目录。 |
| 3 | c #字符设备文件。 |
| 4 | p #管道文件。 |
| 5 | l #符号链接文件。 |
| 6 | f #普通文件。 |

##### rsync 远程同步文件

从ssh的笔记中截取到的一个远程同步文件的指令。这个指令的先进之处在于可以只改动目标目录内核源目录文件不同的文件。

|  |  |
| --- | --- |
| 1 | rsync ［选项］ ［源目录］ ［目标目录］ #基本语法。将源目录本身同步到目标目录内。如果只需要同步源目录下的文件，源目录部分写［源目录］/\*，表示源目录下的文件。和cp、mv的用法相似。 |
| 2 | -a #保留被同步文件的所有属性。 |
| 3 | -v #详细记录过程。 |
| 4 | --delete #删除目标目录中源目录没有的文件。 |

#### 网络——————

##### 常见网络命令

|  |  |
| --- | --- |
| 1 | ifconfig #查看所有网卡信息，包括网卡名和ip地址等。 |
| 2 | ifconfig ［网卡设备名］ #查看特定网卡的信息。 |
| 3 | ip addr show #一般缩写成ip a s，查看网络信息。 |
| 4 | ip a s ［网卡设备名］ #查看指定网卡的信息。 |
| 5 | ip r #查看网关。 |
| 6 | | grep -w inet #管道用法，只显示有ip地址的那一行。 |
| 7 | tcpdump -i ［网络设备名］ #捕获经过特定接口的所有网络流量，使你能够分析网络通信、排查问题或进行其他网络相关的任务。简称抓包。 |
| 8 | traceroute ［ip地址］ #显示数据包从源到目标网址的路径及每一跳的延迟。 |
| 9 | arping -I ［网卡设备名］ ［ip地址］ #查看ip是否冲突。 |
| 10 | arp -n #查看本地arp缓存。 |

ping 检查网络连通性

|  |  |
| --- | --- |
| 1 | ping ［ip地址］ #检查特定ip地址是否与本机连通。 |
| 2 | -c ［数字］ #ping有限次，ping完即止。 |
| 3 | -I ［网卡设备名］ #用特定网卡ping。 |

网卡别名

|  |  |
| --- | --- |
| 1 | ifconfig ［网卡设备名］:［数字］ ［ip地址/24］ #临时设置网卡别名，为同一个网卡配置多个ip地址。数字为0-255的任何数。 |

临时设置的网卡别名会在重启或网卡设备重启时消失。

开机自动配置别名：

|  |  |
| --- | --- |
| 1 | vim /etc/rc.d/rc.local #打开/etc/rc.d/rc.local文件。 |
| 2 | ifconfig ［网卡设备名］:［数字］ ［ip地址/24］ #设置网卡别名。 |
| 3 | chmod +x /etc/rc.d/rc.local #给/etc/rc.d/rc.local文件设置可执行权限。 |

系统在开机进入系统界面前会优先运行/etc/rc.d/rc.local文件里的指令，将设置别名的指令放进/etc/rc.d/rc.local文件可以保证开机自动配置。

网卡别名很可能不能设置网关。

##### nmtui 修改网络配置

除了用图形系统配置网络外，可以用nmtui命令配置网络。

|  |  |
| --- | --- |
| 1 | nmtui-connect #进入交互式界面，可以激活或停用网卡的某个连接，选项带星号表示已激活。 |
| 2 | nmtui-edit #进入交互式界面，修改网卡信息，修改后需要重启网卡刷新。 |
| 3 | nmtui-hostname #进入交互式界面，修改主机名。 |

##### 修改host文件

目的是为一些域名复杂的网络目标起一个简单好记的别名——域名映射。host文件一般在/etc/hosts。以下是host文件内域名映射的格式。

|  |  |
| --- | --- |
| 1 | ［ip地址］ ［域名映射名1］ ［域名映射名2］ …… |

设置好后可以在需要输出特定ip的时候用更好记、更好打的域名映射代替。不设置的话对系统运行也没有影响（除非在脚本中用了没设置的域名映射）。

##### 修改网卡配置文件（ifcfg-\*）

linux的网卡配置文件一般保存在/etc/sysconfig/network-scripts目录下，是一个格式为ifcfg-［网卡连接名］。

编辑网卡配置文件时可以参考帮助手册：

/usr/share/doc/initscripts-9.49.47/sysconfig.txt

|  |  |
| --- | --- |
| 1 | NAME="ens33" #连接名称。 |
| 2 | DEVICE="ens33" #设备名，网卡的名称，这里写什么表示该配置文件指向的网卡连接服务于哪个网卡。 |
| **3** | ONBOOT=yes #在开机时自动启用网卡，=no时则为不自启。 |
| 4 | NETBOOT=yes #开启网关。 |
| 5 | UUID="4dcebd9d-42ec-4f10-87c1-ac5708adee2d" #文件自动生成的网卡的设备码。 |
| 6 | IPV6INIT=yes |
| **7** | BOOTPROTO=none #不设置网卡的ip、网关和DNS（在下方手动输入） ，选择static时，表示静态指定，选择dhcp时，表示自动获取。 |
| 8 | TYPE=Ethernet #网络类型选项，此为以太网类型。 |
| **9** | IPADDR=192.168.12.222 #该网卡连接的ip地址。 |
| **10** | NETMASK=255.255.255.0 #该网卡连接的ip地址的子网掩码，另有两种编写方法，在三种中选择一种：  MASK=255.255.255.0  PREFIX=24 |
| **11** | GATEWAY=192.168.12.1 #该网卡连接的网关。 |
| **12** | DNS1=8.8.8.8 #网卡连接的dns地址，如果要通过域名连接其他终端，需要设置dns地址。重启网络服务后，该地址会同步到/etc/resolv.conf文件中。一个连接最多可以设置三个dns地址。  DNS2=  DNS3= |

##### 重启网络服务

重启网络服务之后，如果BOOTPROTO设置为dhcp，网卡链接会重新获取ip地址、网关和dns。

重启网络服务之后，配置文件的改变开始生效。

（Centos7）

|  |  |
| --- | --- |
| 1 | systemctl restart network #重启网络服务。 |

（Centos8）

方法1：

|  |  |
| --- | --- |
| 1 | nmcli connection reload ens33 |
| 2 | nmcli connection up ens33 |

方法2：

|  |  |
| --- | --- |
| 1 | nmcli connection reload ens33 |
| 2 | systemctl restart NetworkManager |

##### route 路由相关命令

主要目的是查看网关的配置是否成功。

|  |  |
| --- | --- |
| 1 | route -n #查询网关。 |
| 2 | route add default gw ［网关ip］ #添加默认网关。 |
| 3 | route add 0.0.0.0/0 gw ［网关ip］ #同上。 |
| 4 | route add -net ［目标网段］ netmask ［子网掩码］ ［下一跳地址/出接口］ #添加静态路由，将去往特定网段的数据包送到特定地址或接口。 |
| 5 | route add -net ［目标网段］ netmask ［子网掩码］ gw ［网关］ #添加静态路由，将去往特定网段的数据包送到特定网关。比较稳定。 |

route命令创建的网关和路由都是临时的，如果有永久保持的需求，可以将命令放在/etc/rc.d/rc.local。

##### 查询DNS地址

dns地址一般在重启网络服务后同步在/etc/resolv.conf，用cat指令打开它查看。

|  |  |
| --- | --- |
| 1 | cat /etc/resolv.conf #查询dns地址。vim进去可以修改它。 |

##### nmcli 连接管理命令

[device]系列命令

|  |  |
| --- | --- |
| 1 | nmcli dev status #列出所有设备。 |
| 2 | nmcli dev dis ［网卡设备名］ #临时关闭一个网卡设备（关闭网络连接信息）。 |
| 3 | nmcli device connect ［网卡设备名］ # 临时启用网卡连接。 |
| 4 | nmcli dev show ［网卡设备名］ #显示网络接口属性。 |
| 5 | nmcli net off #关闭所有管理接口。 |

[connection]系列命令

|  |  |
| --- | --- |
| 1 | nmcli con show #显示所有的，包括不活动的连接。 |
| 2 | nmcli con up ［连接名］ #激活链接。 |
| 3 | nmcli con down ［连接名］ #取消激活链接。 |
| 4 | nmcli con add #添加一个新的链接。 |
| 5 | nmcli con mod #修改一个链接。 |
| 6 | nmcli con del ［连接名］ #删除一个链接。 |
| 7 | nmcli con show --active #显示所有活动连接。 |

使用nmcli con add 添加新连接时，需要考虑一系列的 选项，可以参考下方：

|  |  |
| --- | --- |
| 1 | nmcli con add ［选项］［参数］［选项］［参数］…… |
| 2 | con-name #连接名选项，参数为自定的连接名。 |
| 3 | type #网络类型连接名，参数为自定的网络类型，可以在参数位置按两下tab获得提示。 |
| 4 | ifname #设备名选项，参数为本连接想要服务的设备的名。 |
| 5 | autoconnect #自启动选项，参数为yes表示开机自启动，如果为no表示开机不启动。 |

nmcli con mod命令与网卡配置文件的对照：

|  |  |  |
| --- | --- | --- |
| 1 | ipv4.method manual | BOOTPROTO=none |
| 2 | ipv4.method auto | BOOTPROTO=dhcp |
| 3 | ipv4.addresss “［ip地址］ ［网关］” | IPADDR0=［ip地址］  PREFIX0=［子网掩码］  GATEWAY0=［网关］ |
| 4 | ipv4.dns ［dns地址］ | DNS0=［dns地址］ |
| 5 | connection.autoconnect yes | ONBOOT=yes |
| 6 | connection.id ［连接名］ | NAME=［连接名］ |
| 7 | connection.interface-name ［设备名］ | DEVICE=［设备名］ |

##### 常用网络设置

|  |  |
| --- | --- |
| 1 | hostname #查看主机名。 |
| 2 | hostname ［主机名］ #修改主机名，临时的，重启后失效。 |
| 3 | hostnamectl set-hostname ［主机名］ #永久修改主机名，无需重启，自动载入/etc/hostname文件。 |

##### 路由转发

|  |  |
| --- | --- |
| 1 | echo 1 > /proc/sys/net/ipv4/ip\_forward #临时开启路由转发功能。 |
| 2 | echo 0 > /proc/sys/net/ipv4/ip\_forward #临时关闭路由转发功能。 |
| 3 | cat /proc/sys/net/ipv4/ip\_forward #查看当前路由转发功能是否开启，1是开启，0是关闭。 |

永久保存路由转发参数：

|  |  |
| --- | --- |
| 1 | vim /usr/lib/sysctl.d/00-system.conf #打开配置文件。 |
| 2 | net.ipv4.ip\_forward = 1 #在文件中输入这行并保存。 |
| 3 | sysctl -p /usr/lib/sysctl.d/00-system.conf #使配置文件中的设置生效。 |

##### DNS服务器

DNS就是域名系统，DNS服务器就是带域名系统的服务器，它可以通过解析器将域名转换成ip地址，供客户端方便地通过域名上网。

这种事在本地的host文件也可以做到，但dns可以用在更大范围的网络上，它看上去要比host文件方便得多。dns使用tcp协议和udp协议的53号端口，其中tcp协议在主从dns同步数据时使用，udp在dns查询时使用。

dns服务器的背后是一个分布式数据库，数据的结构如同一颗倒置的树，树的结构就被叫做域名空间。

|  |  |  |  |
| --- | --- | --- | --- |
| 域名空间 | | | |
| root根域 | | | |
| 顶级域 | | | 顶级域 |
| 二级域 | | 二级域 | 二级域 |
| 三级域 | 三级域 | 三级域 | 三级域 |

域的最大深度为127层，其中每个节点可以存储63个字符，这就意味着一个极限深度的域名可以写得非常长。但完全合格的域名最多只能包含256个字符。

dns服务器有三种类型，首先是主dns服务器，Master dns，就是普通dns服务器。其次是slave dns服务器，从服务器将主服务器的设置复制过来，用以分担主服务器的压力。最后是cache-only dns，缓存服务器可以存储大量dns解析记录，并且只做转发。

常见域名

域名一般分为两类，一列是组织域，以某单词的缩写命名，一类是地理域，以地区的缩写命名。

|  |  |
| --- | --- |
| 1 | .com #组织域名，商业、企业域名，表示商业、企业、公司。这是当前最广泛流行的域名格式，全球超过1亿个用户。 |
| 2 | .net #组织域名，网络，一般用于从事网络服务的机构或公司，诞生于上世纪80年代。 |
| 3 | .org #组织域名，全球第三大顶级域名，可以追溯到1985年，表示组织organizations。 |
| 4 | .gov #美国政府专用域名。 |
| 5 | .edu #教育机构专用域名。 |
| 6 | .mil #美国军队网站专用域名。由于美国网络起的早，所以占的多。 |
| 7 | .int #国际组织专用域名。 |
| 8 | .cn #地理域，中国顶级域名。 |

查询流程

DNS查询流程包括一次递归并在其中进行多次迭代（在不能直接完成任务的时候提供完成任务的更优方案），详细点说包括以下几个步骤：

|  |  |
| --- | --- |
| 1 | 用户在浏览器中输入域名。 |
| 2 | 用户的计算机首先检查本地缓存中是否有该域名的IP地址。如果找到，则直接使用该IP地址进行连接。 |
| 3 | 如果没有找到，则查询本地DNS解析器（通常是用户的网络服务提供商提供的）。 |
| 4 | 本地DNS解析器首先检查其缓存中是否有该域名的IP地址。如果找到，则返回给用户。 |
| 5 | 如果没有找到，则本地DNS解析器会向根域名服务器发送查询请求。根域名服务器会返回一个顶级域名（TLD）服务器的地址，例如.com、.org或.net等。 |
| 6 | 本地DNS解析器接着向相应的TLD服务器发送查询请求。TLD服务器会返回一个权威域名服务器的地址，该服务器负责管理特定域名的记录。 |
| 7 | 本地DNS解析器最后向权威域名服务器发送查询请求。权威域名服务器会返回该域名的IP地址。 |
| 8 | 本地DNS解析器将IP地址返回给用户的计算机，用户的计算机使用该IP地址与目标服务器建立连接。 |
| 9 | 用户的计算机将查询结果缓存一段时间，以加快未来对该域名的访问速度。 |

以上整个过程通常是自动完成的，用户不需要进行任何额外的操作。

配置文件

dns服务器的工作软件是bind，下载bind后，一些配置文件将会起作用。

|  |  |
| --- | --- |
| 1 | /etc/named.conf #dns服务器的主配置文件，决定了此服务器可以解析哪个域，并且指定区域配置文件的名称。 |
| 2 | /etc/named.rfc1912.zones #区域文件，文件名字是可变的，取决于配置文件里的相关设置。主要记录该dus服务器可解析的域。 |
| 3 | /var/named #bind程序的工作目录，存放正反解文件。正解文件支持通过主机名查询ip地址，反解文件支持通过ip地址查主机名。 |

主配置文件named.conf原始状态内容如下：

|  |  |
| --- | --- |
| 1 | options { #这个大的集合里面设置了dns的主要功能，其中内容每行都需要用分号结尾。 |
| 2 | listen-on port 53 { 127.0.0.1; }; #允许哪些主机访问53号端口。这里可以在括号里写any;表示允许所有主机。 |
| 3 | listen-on-v6 port 53 { ::1; }; |
| 4 | directory "/var/named"; #设定dns服务的zone文件位置。 |
| 5 | dump-file "/var/named/data/cache\_dump.db"; #设定域名缓存文件的保存位置。 |
| 6 | statistics-file "/var/named/data/named\_stats.txt"; #设定记录dns工作状态的文件。 |
| 7 | memstatistics-file "/var/named/data/named\_mem\_stats.txt"; #设定记录内存状态的文件。 |
| 8 | recursing-file "/var/named/data/named.recursing"; |
| 9 | secroots-file "/var/named/data/named.secroots"; |
| 10 | allow-query { localhost; }; #设定哪些主机可以访问dns服务，可以用分号相隔多个地址，也可以写any;表示允许全部。 |
| 11 | recursion yes; |
| 12 | dnssec-enable yes; |
| 13 | dnssec-validation yes; |
| 14 | bindkeys-file "/etc/named.root.key"; |
| 15 | managed-keys-directory "/var/named/dynamic"; |
| 16 | pid-file "/run/named/named.pid"; |
| 17 | session-keyfile "/run/named/session.key"; |
| 18 | }; |
| 19 | logging { #日志相关设定。 |
| 20 | channel default\_debug { |
| 21 | file "data/named.run"; #保存debug级别日志的文件名。 |
| 22 | severity dynamic; |
| 23 | }; |
| 24 | }; |
| 25 | zone "." IN { #定义13台根域服务器的地址。 |
| 26 | type hint; |
| 27 | file "named.ca"; #记录根域服务器地址的文件。 |
| 28 | }; |
| 29 | include "/etc/named.rfc1912.zones"; #设定控制dns服务器可解析区域的管理文件。 |
| 30 | include "/etc/named.root.key"; |

上文中提到了主配置文件默认将可解析域的管理文件设为/etc/named.rfc1912.zones，以下就是该文件内部：

|  |  |
| --- | --- |
| 1 | zone "［域名］" IN { #正解配置举例。 |
| 2 | type master; #该服务器类型，master是主服务器，slave是辅助服务器，hint是根域服务器，forward转发服务器，delegation-only委派服务器，stub只从主服务器学习NS记录。 |
| 3 | file "［相对路径正解配置文件名］"; #设定正解配置文件，这个文件必须存放在/var/named下，且所有者为root，所属组为named，权限为640。 |
| 4 | allow-update { none; }; #设定是否允许接收其他主机更新，none是不限制。 |
| 5 | }; |
| 6 | zone "［目标网段反写］.in-addr.arpa" IN { #反解配置举例。 |
| 7 | type master; |
| 8 | file "［相对路径反解配置文件名  ］"; |
| 9 | allow-update { none; }; |
| 10 | }; |

正解、反解配置文件本身是不存在的，也不会随着设置而自动生成，所以要么需要自己建，要么需要cp -p复制已有的同类型配置文件。

zone文件内容的格式是这样的：

|  |  |
| --- | --- |
| 1 | $TTL 1D #基本缓存时间。 |
| 2 | ［域名］ ［记录类型］ SOA ［主dns服务器的域名］ ［管理员邮箱］ ( |
| 3 | 0 ; serial #序列号，默认为零，dns区域被修改时自动增加。 |
| 4 | 1D ; refresh #刷新时间，不小于重试时间的两倍。 |
| 5 | 1H ; retry #重试时间 |
| 6 | 1W ; expire #过期时间，大于刷新时间和重试时间的和，且不小于重试时间的10倍，一般固定在7天，也就是1W。 |
| 7 | 3H ) ; minimum #最小缓存时间。 |
| 8 | NS ［名称服务器域名］ #设定授权名称服务器的域名 |
| 9 | A ［映射地址］ |
| 10 | AAAA ［映射ipv6地址］ |
| 11 | ［域名］ IN ［资源记录类型］ ［地址］ |

关于基于资源记录的记录类型：

|  |  |
| --- | --- |
| 1 | SOA #起始授权记录，对整个域进行管理工作，一个zone文件只能有一个。 |
| 2 | NS #指明自己在本区中的授权服务器。每个zone文件至少包含一个NS记录。 |
| 3 | A #主机记录，将域名映射成地址，形成域名和ip的对应关系。 |
| 4 | PTR #反解记录，与主机记录相对。 |
| 5 | CNAME #别名记录。 |
| 6 | MX #邮件交换记录。 |

#### 硬盘——————

##### fdisk 分区工具

|  |  |
| --- | --- |
| 1 | fdisk -l #查询所有磁盘信息，显示每个分区的起始点和终止点。 |
| 2 | fdisk ［磁盘名］ #进入针对该粗盘的分区状态。这是个交互式分区指令。 |

以下是交互式分区状态的指令：

|  |  |
| --- | --- |
| 1 | m #获取帮助。 |
| 2 | n #开始分区。之后会提示选p还是选e、l。分别表示不同的分区类型。 |
| 3 | p #设置接下来做的分区是主分区（primary）。一个硬盘最多有四个分区，一般划分三个，剩余一个分给逻辑分区。 |
| 4 | e #设置接下来做的分区是扩展分区（extended）。 |
| 5 | l #设置接下来做的分区是逻辑分区（logical）。扩展分区后可以分逻辑分区，逻辑分区一般可以分15个。 |
| 6 | w #保存退出。 |
| 7 | q #不保存退出。 |

一般情况下，输入p、e或者l之后会逐渐提示以下信息：

|  |  |
| --- | --- |
| 1 | 分区号 (1-4，默认 1)：  #什么也不输入直接按回车将视为选择默认选项。一般可以选择默认选项，除非有别的安排。 |
| 2 | 起始 扇区 (2048-10485759，默认为 2048)：  #什么也不输入直接按回车将视为选择默认选项，其中硬盘的前2048扇区是引导分区，所以一般而言第一个分区从2048s开始。除非有其他安排，否则这里可以直接取默认的选项。 |
| 3 | Last 扇区, +扇区 or +size{K,M,G} (2048-10485759，默认为 10485759)：  #这里的选项一般要自己决定，它决定的分区的大小。可以输入扇区大小和容量大小，格式为：+［扇区数字］和+［容量数字］［K,M,G］ |

到这里一个分区就设置好了，最后以w退出。

##### parted 分区工具

它相比fdisk的优势是支持非交互式分区，方便写脚本。

|  |  |
| --- | --- |
| 1 | parted ［选项］ ［设备名］ ［指令］ |
| 2 | parted ［设备名］ mklabel msdos #将硬盘的分区模式设置为mds。 |
| 3 | parted ［设备名］ mkpart ［分区类型］ ［起始点］ ［终止点］ #非交互式地为硬盘设置分区（msdos模式）。 |

示例：parted /dev/sdb mkpart p 2048s 1000M

|  |  |
| --- | --- |
| 1 | parted ［设备名］ mklabel GPT #将硬盘的分区模式设置为GPT。 |
| 2 | parted ［设备名］ mkpart ［分区名］ ［起始点］ ［终止点］ #非交互式地为硬盘设置分区（gpt模式）。它无需指定分区类型，但需要为分区命名。 |

其他：

|  |  |
| --- | --- |
| 1 | parted ［设备名］ rm ［分区编号］ #删除特定编号的分区。 |

##### mkfs 格式化硬盘分区

格式化的目的是重设文件系统，从而为之后的“挂载”铺路。常用文件系统和操作系统的关系如下，为分区设定文件系统时也要考虑这些因素。另外，mkfs的回复有误导性，使用echo $?检查是否成功。

|  |  |  |  |
| --- | --- | --- | --- |
| 操作系统类型 | 文件系统类型 | | |
| linux | exts | xfs | exfat |
| windows | fat32 | ntfs | exfat |
| 网络文件系统 | cifs | nfs |  |
| MacOS | apfs | exfat |  |

以下是代码部分：

|  |  |
| --- | --- |
| 1 | mkfs -t ［文件系统］ ［分区号］ #将分区设定成特定文件系统。 |
| 2 | mkfs.［文件系统］ ［分区号］ #上一条命令的另一种方法。 |

部分文件系统对分区大小有要求，如果提示没有那个文件或目录，可能不符合大小条件。

##### lsblk/blkid 查询分区及文件系统

分区一般存在/dev目录下。

|  |  |
| --- | --- |
| 1 | lsblk #简单清晰地查询分区状态。 |
| 2 | lsblk --fs #查询各分区状态，相比上一条，也显示分区的设备码和文件系统。 |
| 3 | lsblk --fs ［分区名/磁盘名］ #只查询特定分区或磁盘的状态。 |
| 4 | blkid #用标准输出列出各个分区的全部状态。数据多了会显得乱，但它会显示分区的机器码。 |
| 5 | blkid ［分区名］ #只查询特定分区的状态。 |

都可以同时查询多个分区。

以下是回复信息里每一列的含义：

|  |  |
| --- | --- |
| 1 | NAME #块设备名。 |
| 2 | MAJ:MIN #主要和次要编号。 |
| 3 | RM #是否可移动设备。1可移动，0不可移动。 |
| 4 | SIZE #设备的容量大小。 |
| 5 | RO #是否只读。1是，0不是。 |
| 6 | TYPE #存储设备类型。disk硬盘，part主分区，lvm逻辑分区，rom只读存储设备。 |
| 7 | MOUNTPOINT #挂载点。 |

##### mount 挂载分区

就在“刚才”，yum的那一节提到了挂载本地仓库，mount也能挂载已格式化的分区到目录。挂载的意义是让系统能够访问设备。挂载的流程是先创建一个空目录，再将文件系统挂载到那个目录上。

|  |  |
| --- | --- |
| 1 | mount #查看挂载权限。 |
| 2 | mount ［分区名］ ［目录名］ #将分区临时挂载到目录。 |
| 3 | mount -o ［权限］ ［分区名］ ［目录名］ #将分区临时挂载到目录并设置权限。多个权限之间用逗号隔开。 |
| 4 | mount -o remount,［权限］ ［分区名］ ［目录名］ #重新挂载分区并设置权限。 |
| 5 | mount -a #挂载fstab中存在但还挂载的文件系统。 |

关于权限，见下表：

|  |  |  |  |
| --- | --- | --- | --- |
| rw | 读写 | ro | 只读 |
| atime | 更新访问时间 | noatime | 不更新访问时间 |
| sync | 同步 | async | 异步 |
| dev | 启用设备文件 | nodev | 不启用设备文件 |
| exec | 执行文件可被执行 | noexec | 执行文件不可被执行 |
| suid | 启用特殊权限功能 | nosuid | 不启用特殊权限功能 |
| user | 允许普通用户执行mount | nouser | 不允许普通用户执行mount |
| defaults | rw+suid+dev+exec+auto+nouser+async | | |

绑定

|  |  |
| --- | --- |
| 1 | mount -o bind ［原始目录］［目标目录］ #将两个挂载点绑定到一起，目的是避免多个目标挂载到同一目录上导致相互覆盖（实际上覆盖了也是挂载成功）。 |

永久挂载

用vim将mount指令写进开机优先执行加载的/etc/rc.d/rc.local的末尾并保存退出，将/etc/rc.d/rc.local设置为可执行权限。重启后就可以实现自动挂载。

除此之外，/etc/fstab是系统的自动挂载文件，可以用vim修改它的最后一行，它的书写标准是这样的：、

|  |  |
| --- | --- |
| 1 | 第一列，文件系统。写分区的地址。也可以用分区的机器码（UUID），使用机器码的时候以“UUID=”开头，用双引号把机器码括起来。 |
| 2 | 第二列，挂载点。写挂载点的地址。 |
| 3 | 第三列，文件系统。写分区的文件系统。例子：xfs。 |
| 4 | 第四列，权限或参数。例子为defaults。 |
| 5 | 第五列，dump备份，数字形式。0不备份，1,2备份。 |
| 7 | 第六列，fsck检查，数字形式。0不检测，1,2检测。主要检测磁盘完整性，xfs文件系统不支持，写0。 |

写完后，可以用mount保证内容生效，fstab文件必须保证格式、内容正确，否则可能重启后无法正常开机。

当fstab文件配置错误

重启后将发现系统进入了紧急模式，这时先输入root密码进入shell提示符界面，给根目录添加读写权限，vim修改/etc/fstab的错行。再重启进入。

##### umount 卸载分区

|  |  |
| --- | --- |
| 1 | umount ［设备］ #将指定设备从挂载点卸载。 |
| 2 | umount ［挂载点］ #将制定挂载点的设备卸载。 |
| 3 | -l #强制卸载，允许在进入设备挂载点的同时卸载设备。 |

##### df -Th查看挂载状态

|  |  |
| --- | --- |
| 1 | df -Th #查看挂载状态。 |

##### partprobe 刷新分区表

|  |  |
| --- | --- |
| 1 | partprobe #刷新分区表，通知内核。 |

##### autofs 按需挂载服务

autofs是一种服务，在配置完毕以后，它应当设为开机启动。autofs启动时，如果系统用到挂载时自动将文件系统挂载，不使用时会自动卸载，比较省事。

通常的使用流程为：安装服务应用，进行配置，开启并设置为开机自启。

安装

|  |  |
| --- | --- |
| 1 | yum install autofs #安装自动挂载服务。 |
| 2 | systemctl enable --now autofs #启动自动挂载服务。 |

配置

|  |  |
| --- | --- |
| 1 | mkdir ［映射挂载点］ #首先需要准备autofs的运行环境，这可能需要提前准备一个映射挂载点。映射挂载点必须与映射文件中书写的一致。 |
| 2 | ~~vim /etc/auto.master #配置主控制文件。~~ |
| 3 | echo ［基础挂载点］ ［映射文件］ >> /etc/auto.master #在主控制文件内追加新行，其中基础挂载点是指映射挂载点的上一级目录。 |
| 4 | echo ［映射挂载点］ -fstype=［文件系统类型］,［权限］ ：［文件系统］ >> /etc/auto.［文件系统］ #在映射文件内追加新行，其中映射挂载点是指基础挂载点下用作挂载文件系统的目录，映射文件的名字的一部分最好含有文件系统的名字。 |

第3步举例：echo /mnt/auto /etc/auto.sdc1 >> /etc/auto.master

第4步举例：echo auto1 -fstype=xfs,rw :/dev/sdc1 >> /etc/auto.sdc1

如果有题目提到虚拟路径，它指的就是映射挂载点。

重启生效和验证

|  |  |
| --- | --- |
| 1 | systemctl restart autofs #重启autofs服务。 |
| 2 | cd /［基础挂载点］/［映射挂载点］ #进入映射挂载点。 |
| 3 | df -Th #检查是否自动挂载生效。 |
| 4 | cd #返回家目录。 |
| 5 | 过一段时间。 |
| 6 | df -Th #检查是否自动卸载。超时时间初始为300秒。 |

自定义卸载时间：

|  |  |
| --- | --- |
| 1 | vim /etc/autofs.conf #进入编辑autofs的主配置文件。 |
| 2 | timeout = 5 #找到timeout的行并将其改为5，表示在不使用的5秒后自动卸载。 |
| 3 | watch -n 1 mount #每秒运行一次mount，检查挂载的情况。 |

注意！！

autofs除了占用映射挂载点以外，还会占用基础挂载点，如果它占用了基础挂载点，将会导致基础挂载点原有的用于挂载文件系统的目录不能用ls看到,也不能被其他操作使用和更改！

第二种挂载方法

将目标ftp服务器的共享文件挂载到本地的虚拟目录。

首先安装autofs软件包并启动服务。

其次创建/etc/auto.master.d/［名字.autofs］，并在其中加入以下行：

|  |  |
| --- | --- |
| 1 | ［基础挂载点］ ［映射文件名］ #映射文件在/etc目录下并且以auto.开头。 |

复制主映射文件auto.misc到相同目录并将其重命名为［名字.autofs］，在其中再追加一行：

|  |  |
| --- | --- |
| 1 | cp /etc/auto.misc /etc/［名字.auto］ #复制并改名。 |
| 2 | echo ［虚拟目录］ ［权限］ ［ftp地址:共享目录名］ >> /etc/［名字］.autofs #配置映射文件。 |

最后重启autofs服务。

直接映射方法

|  |  |
| --- | --- |
| 1 | echo /- /etc/［auto.名字］ >> /etc/auto.master.d/［名字.autofs］ #创建和修改控制文件。 |
| 2 | echo ［映射目录名（绝对路径）］ ［权限］ ［ftp地址:共享目录名］ >> /etc/auto.nfs #修改固定名称映射文件。 |
| 3 | systemctl restart autofs #重启服务。 |

由于直接映射方法会在特殊的时候覆盖掉包括根目录在内的整个文件系统，所以不建议在个人计算机上弄。

间接映射方法

|  |  |
| --- | --- |
| 1 | echo ［基础挂载点］ /etc/auto.nfs >> /etc/auto.master.d/［名字.autofs］ #创建和修改控制文件。 |
| 2 | echo ［映射目录名］ ［权限］ ［ftp地址:共享目录名］ >> /etc/auto.nfs #修改固定名称映射文件。 |
| 3 | systemctl restart autofs #重启服务。 |

如果共享文件有多个子目录可以用&代替子目录名，写作［ftp地址:&］。

##### mdadm 磁盘阵列

磁盘阵列主要为了解决三个问题：数据可靠性无法保证、容量有限、吞吐量有限。

|  |  |
| --- | --- |
| raid级别 | 描述 |
| 0，条带化 | 通过条带化方式记录数据，至少2块硬盘，大小尽量一致，优点是读写速度快，磁盘利用率最高。缺点是其中一块硬盘损坏时，另一块数据也会受到影响。 |
| 1，镜像 | 至少2块硬盘，数据分别记录到两块硬盘中，优点是一块当另一块的备份，缺点是利用率只有50%。 |
| 5，奇偶条带化 | 至少3块硬盘，可以从硬盘中存储的奇偶校验吗恢复数据，提升可靠性。性能在raid1和raid0之间。 |
| 10 | 至少4块硬盘，等于先组成两组镜像再将其条带化配置。 |

mdadm指令的选项有多种形态，如下表所示：

|  |  |  |
| --- | --- | --- |
| 第一形态 | 第二形态 | 效果 |
| --create | -C | 创建 |
| --level=［数字］ | -l ［数字］ | 指定raid等级，数字部分是等级，有0、1、5、10等多种选项 |
| --raid-devices=［数字］ | -n ［数字］ | 指定使用设备的数量，不同的raid等级对设备数量有最低需求。 |

以下是指令部分的举例：

|  |  |
| --- | --- |
| 1 | mdadm -C ［磁盘阵列名］ -l ［数字］ -n［数字］ ［设备名］ #创建磁盘阵列。 |
| 2 | -C #创建。 |
| 3 | -l ［数字］ #指定raid等级，有0，1，5，10等多种选择。 |
| 4 | -n ［数字］ #指定使用设备数量，不同等级的磁盘阵列对于设备的最小数量要求也不同。 |
| 5 | -x ［数字］ #指定添加多少个备份设备。备份设备可以替代损坏的设备。 |
| 6 | mdadm -D ［磁盘阵列名］ #查询磁盘阵列的状态。 |
| 7 | mdadm -S ［磁盘阵列名］ # 停止指定的磁盘阵列，停止后对应的磁盘阵列将完全不可用。 |
| 8 | mdadm -A ［磁盘阵列名］ ［成员设备名］ #激活指定的磁盘阵列。 |
| 9 | mdadm -f ［磁盘阵列］ ［设备名］ #报损磁盘阵列的一个设备。 |
| 10 | mdadm -r ［磁盘阵列］ ［设备名］ #移除磁盘阵列的一个设备。 |
| 11 | mdadm -a ［磁盘阵列］ ［设备名］ #为指定的磁盘阵列添加一个备份设备。 |

其他查询方法：

|  |  |
| --- | --- |
| 1 | cat /proc/mdstat #看缓存文件里的md设备状态。和-D效果类似。 |

使用

使用磁盘阵列也需要进行分区和挂载。分区和挂载的过程和正常分区差不多，故略。

删除

如果要删除raid设备，需要完成两步：

|  |  |
| --- | --- |
| 1 | rm -rf ［磁盘阵列名］ #删除raid设备。 |
| 2 | mdadm --zero-superblock ［成员设备名］ #清除设备超级块信息，使其不属于某个md设备。 |

永久生效

以上方法都会在重启之后生效，如果没想起来重新做就有损失数据的风险，即使想得起来也很费时间，所以永久生效是有必要的。

|  |  |
| --- | --- |
| 1 | vim /etc/mdadm.conf #进入mdadm配置文件，查看文件首部有没有DEVICE开头的数据。或者DEVICE开头的数据是否包含已经使用过raid的全部设备。 |
| 2 | DEVICE ［设备名］ ［设备名］ ［设备名］ #如果没有或者不全，就照着这个格式将DEVICE的行补全。 |
| 3 | mdadm -Ds ［磁盘阵列名］ >> /etc/mdadm.conf #将mdadm -Ds生成的设备信息追加到mdadm配置文件中。 |
| 4 | cat /etc/mdadm.conf #检查是否追加成功。 |

使用或停用的流程

|  |  |
| --- | --- |
| 1 | 创建磁盘阵列。 |
| 2 | 格式化和挂载磁盘阵列。 |
| 3 | 修改mdadm配置文件，使其永久生效。 |
| 4 | 修改/etc/fstab文件，使其开机自动挂载。 |
| 5 | 重启验证。 |

停用时：

|  |  |
| --- | --- |
| 1 | umount从挂载点卸载磁盘阵列。 |
| 2 | rm删除磁盘阵列文件。 |
| 3 | 清除mdadm配置文件和fstab文件里的相关内容。 |
| 4 | 清除相关超级块。 |

##### LVM逻辑卷

设置过程中均有返回信息，建议随时echo $?检验指令成功与否。

PV 物理卷

|  |  |
| --- | --- |
| 1 | pvcreate ［分区名］ ［分区名］ #将分区制作成物理卷。 |
| 2 | pvs #显示物理卷信息。 |
| 3 | pvscan #显示简略信息。 |
| 4 | pvremove ［物理卷名］ #删除物理卷。 |
| 5 | pvdisplay #用表格显示详细信息。 |

pvs界面的有以下重要信息：

|  |  |
| --- | --- |
| 1 | PV #物理卷名，其中/dev/sda2是根目录物理卷。 |
| 2 | VG #所属卷组。 |
| 3 | PFree #如果不为零表示空闲可用。 |

VG 卷组

|  |  |
| --- | --- |
| 1 | vgcreate -s 16M ［卷组名］ ［物理卷名］ ［物理卷名］ #将指定的物理卷划入特定卷组，且将PE块的大小设置为16M。 |
| 2 | -s ［PE大小］ #设置卷组的PE大小，常见取值范围是4M到16M。 |
| 3 | vgdisplay #显示vg信息。 |
| 4 | vgremove ［卷组名］ #删除卷组。创建错误时可以使用。 |
| 5 | vgchange -a n ［卷组名］ #使卷组不再具有Active标志，之后可删除。 |

vgdisplay回复的数据中：

|  |  |
| --- | --- |
| 1 | VG Name #vg名称。 |
| 2 | VG Size #VG总大小。 |
| 3 | PE Size #每个PE大小。 |
| 4 | Total PE #PE数量总和。 |
| 5 | Alloc PE / Size #可使用的pe数量。 |
| 6 | Free PE / Size #剩余的pe数量。 |

LV逻辑卷

|  |  |
| --- | --- |
| 1 | lvcreate -l 60 -n ［逻辑卷名］ ［卷组名］ #在卷组中划分一个逻辑卷，并为其划分60个PE块。 |
| 2 | -l ［数字］ #指定pe块的数量。 |
| 3 | -L ［pe块总大小］ #指定pe块的总大小，必须可以被pe块大小整除。不和-l一同使用。 |
| 4 | -n ［逻辑卷名］ #指定分出来的逻辑卷的名字。如果不选，新的逻辑卷会从lvol0开始命名。 |
| 5 | lvs #查看系统中所有逻辑卷的详细信息，几乎是所有的信息。 |
| 6 | lvscan #查看系统中所有逻辑卷的简略信息。 |
| 7 | lvdisplay ［逻辑卷名］ #用列表查看逻辑卷的详细信息。 |
| 8 | lvremove ［卷组名］ #挨个交互式删除逻辑卷。 |

lvdisplay出来的表格有如下信息：

|  |  |
| --- | --- |
| 1 | LV Path #实际物理路径。 |
| 2 | LV Name #lv名字。 |
| 3 | VG Name #该lv属于哪个卷组。 |
| 4 | LV Size #lv大小。 |
| 5 | Current LE #LE数量。 |

此后就是格式化和挂载，和普通分区/文件系统的处理方式没有什么不同。

卸载流程

|  |  |
| --- | --- |
| 1 | 卸载lvm文件系统，如果在fstab文件里写了相关的自动挂载文档，也需要将相关内容注释或清空。 |
| 2 | 使用lvremove挨个删除逻辑卷。 |
| 3 | 使用vgchange -a n ［卷组名］让卷组不再具有Active标志。 |
| 4 | 使用vgremove删除卷组。 |
| 5 | 使用pvremove删除物理卷。 |

扩容

|  |  |
| --- | --- |
| 1 | lvextend ［选项］ ［参数］ ［逻辑卷名］ #为逻辑卷扩容。 |
| 2 | -L ［+数字］M/G #在原有基础上增加一定大小。 |
| 3 | -L ［数字］M/G #将逻辑卷容量增加到特定值。 |
| 4 | -l ［+数字］ #在原有基础上增加一定数量的PE块。 |
| 5 | -l ［数字］ #将逻辑卷PE数量增加到特定值。 |
| 6 | resize2fs ［逻辑卷名/挂载点］ #（ext）同步文件系统和逻辑卷的大小。 |
| 7 | xfs\_growfs ［逻辑卷名/挂载点］ #（xfs）同步文件系统和逻辑卷的大小。 |

在线扩容和离线扩容的流程几乎是一样的，仅有同步的时候有点显而易见的区别。

缩小（仅ext4支持）

|  |  |
| --- | --- |
| 1 | 首先用umount卸载文件系统。 |
| 2 | resize2fs ［逻辑卷名］ ［缩小后大小］ #将逻辑卷的实际大小缩减为指定数值。 |
| 3 | 再用mount重新挂载看看大小。 |
| 4 | lvreduce -L ［+数字］/［数字］ ［逻辑卷名］ #缩小逻辑卷容量。 |
| 5 | 完成。 |

其他方法：

|  |  |
| --- | --- |
| 1 | lvresize --size ［数字］M ［逻辑卷名］ #缩小逻辑卷容量，并自动调整逻辑卷的容量大小，不关心缩小后具体大小时使用。 |

快照

|  |  |
| --- | --- |
| 1 | lvcreate -L 500M -s -n ［快照名］ ［逻辑卷名］ #拍摄一个500M大小的快照，快照放在和逻辑卷相同的vg目录内。 |
| 2 | mount ［快照全名］ ［挂载点］ #恢复快照。 |

##### swap分区

交换分区，其作用是充当虚拟内存。当系统中的内存用光时，就会将一部分文件送到swap分区内，避免系统崩溃或缓慢。除此之外，它也用来支持系统的休眠功能，当系统休眠时，内存中的数据就会保存到swap分区。

swap不一定在各种情况下都够用，所以有时需要进行扩容或更改：

|  |  |
| --- | --- |
| 1 | 用fdisk或parted准备一个容量更大的分区。 |
| 2 | mkswap ［分区名］ #将准备好的分区格式化为swap文件系统。 |
| 3 | swapon ［swap文件系统名］ #临时加载新的swap文件系统。是添加而非更换。 |
| 4 | swapoff ［swap文件系统名］ #卸载swap文件系统。 |

有两种方法可以使swap的设置永久生效，其一：

|  |  |
| --- | --- |
| 1 | vim /etc/rc.local #进入编辑rc.local文件。如果该文件可执行，那么该文件内的指令开机自动运行。 |
| 2 | 在其中追加行：swapon ［swap文件系统］ |

其二：

|  |  |
| --- | --- |
| 1 | vim /etc/fstab #进入编辑fstab文件。 |
| 2 | ［swap文件系统名或UUID］ swap swap ［权限］ 0 0  #swap在fstab文件中的特殊格式。 |
| 3 | swapon -a #加载fstab文件内存在但未挂载的swap文件系统，和mount -a基本相同。 |

其他：

|  |  |
| --- | --- |
| 1 | free #查看系统当前内存的使用情况。 |
| 2 | -m #以M（兆字节）为单位查看。 |
| 3 | -h #按最大单位查看且显示单位。 |
| 4 | swapon -s #查看所有激活的swap文件系统。 |

通过修改分区编号将分区类型声明用于swap：

|  |  |
| --- | --- |
| 1 | fdisk ［磁盘名］ #进入针对特定磁盘的交互式操作界面。 |
| 2 | t #输入t表示改变文件声明用途。 |
| 3 | ［分区号］ #输入分区号表示对哪个分区生效。 |
| 4 | 82 #输入代表声明swap分区的hex代码。 |

做完这些后，分区并没有真的被格式化为swap分区，只是声明了一下，仍然需要用mkswap命令进行格式化。

基于扩容swap分区时展露出的特性，缩小swap容量只需要卸载并删除相应的分区即可。

在本地硬盘上创建swap空间：

|  |  |
| --- | --- |
| 1 | df -h #先检查系统的可用空间。/dev/mapper/centos-root就是系统可用空间的名。 |
| 2 | dd if=/dev/zero of=［swap空间名（绝对路径）］ bs=［块大小］ count=［块数量］ #将一定系统容量转化为swap空间容量。 |
| 3 | mkswap ［swap空间名］ #格式化swap空间。 |
| 4 | chmod 600 ［swap空间名］ #设置swap空间权限只允许所有者读写。 |
| 5 | swapon ［swap空间名］ #加载swap空间。 |

#### 进程与服务——————

每个进程在进入内存时都会携带有一定、由程序生成的信息，其中包括PID，执行者权限及属性以及程序相关代码，这些也是进程的属性。

|  |  |
| --- | --- |
| 1 | echo $$ #查看当前shell编译器的进程号。 |
| 2 | bash #进入当前shell创造的子shell 编译器。 |

##### ps 查看进程（静态）

|  |  |
| --- | --- |
| 1 | ps #查看进程 |
| 2 | -l #显示更详细的进程信息。 |
| 3 | -ef #显示更多的进程信息。 |
| 4 | aux #查看所有系统运行程序。 |
| 5 | a #显示现行终端机下的所有程序，包括其他用户的程序。 |
| 6 | u #以用户为主的格式来显示程序状况。 |
| 7 | x #显示所有程序，不以终端机来区分。 |

以下是ps -l 列出的各信息的含义：

|  |  |  |  |
| --- | --- | --- | --- |
| F | 状态标志 | S | 进程的状态 |
| UID | 所有者uid | PID | 进程号 |
| PPID | 父进程号 | C | CPU利用率 |
| PRI | 优先级 | NI | nice值 |
| ADDR | 内存地址 | SZ | 内存大小 |
| WCHAN | 等待 | TTY | 终端机位置 |
| TIME | 花费的时间 | CMD | 命令行 |

举例表示其中的具体含义：

|  |  |  |
| --- | --- | --- |
| F | 0 | 进程正常运行。 |
| 1 | 进程在内存中，且因为正在执行内核中的关键代码，不能被中断。表示子程序仅进行复制fork而没有实际执行exec。 |
| 2 | 进程在内存中，但不能被中断，可能在执行关键操作。 |
| 4 | 进程在内存中，且可以被中断。表示程序权限为root。 |
| 8 | 进程已退出。 |
| S | R | 正在运行。 |
| S | 可中断（可被唤醒）的睡眠状态。 |
| D | 不可中断的睡眠状态。通常可能在等待I/O。 |
| T | 可能是jobs后台程序或traced跟踪状态。 |
| Z | 退出状态，进程成为僵尸进程，虽然终止，但是在内存中无法移除。 |

以下是ps -ef显示的数据的标题的含义：

|  |  |  |  |
| --- | --- | --- | --- |
| UID | 所有者用户名 | PID | 进程标志 |
| PPID | 父进程标志 | C | cpu占用率 |
| STIME | 进程开始时间 | TTY | 进程所在的终端机 |
| TIME | 使用时间 | CMD | 命令行 |

以下是TTY的具体说明：

|  |  |
| --- | --- |
| 1 | ? #与终端机无关。 |
| 2 | tty1-tty6 #是本机上登录者进程 |
| 3 | pts/0 #由网络连接到主机的进程 |

##### pstree 查看系统进程树

pstree是查看系统进程树的指令，它需要首先安装才能使用，并没有被CentOS7自带。

|  |  |
| --- | --- |
| 1 | yum instll -y psmisc-22.20-16.e17.x86\_64 #安装psmisc软件。 |
| 2 | pstree 查看进程树。 |
| 3 | -p #显示进程号。 |

这个命令会展示出系统中的所有进程和子进程及其关系。由于是标准输出，也可以后缀| grep ［进程名］来查找特定的进程。

##### top 进入监控状态

|  |  |
| --- | --- |
| 1 | top #进入监控状态。 |
| 2 | -d ［秒数］ #设置屏幕刷新时间。 |

以下是监控状态的命令：

|  |  |
| --- | --- |
| 1 | h #显示快捷键帮助。 |
| 2 | k #然后输入进程号，表示终止一个进程。 |
| 3 | r #重新安排一个进程的优先等级。 |
| 4 | s #更改刷新间隔时间，单位为秒。 |
| 5 | l #切换为显示进程和cpu状态信息。 |
| 6 | c #切换为显示命令名称和完整命令行。 |
| 7 | M #根据内存使用大小排序。 |
| 8 | P #根据cpu使用时间排序（默认的）。 |
| 9 | T #根据时间/渐进时间进行排序。 |
| 10 | 1 #展示多核cpu显示 |
| 11 | q #退出top程序。 |

##### uptime 显示各种时间

|  |  |
| --- | --- |
| 1 | uptime #显示当前时间，系统运行了多长时间，当前登录的用户有多少，前1、5、15分钟系统的平均负载。 |

##### pgrep 查看进程信息

|  |  |
| --- | --- |
| 1 | pgrep ［进程关键词］ #查看含有关键词的进程的进程号。这是个简单的竖排列表，只有1列数据。 |
| 2 | -l #同时查看进程的进程名和进程号。 |

##### kill 杀死进程

|  |  |
| --- | --- |
| 1 | kill ［-信号编号］ ［进程号］ #中止进程。 |
| 2 | 1 #SIGHUP，对控制进程或终端进行挂起检测。 |
| 3 | 2 #SIGINT，在终端直接按下ctrl+c时发送该信号。中断当前运行的命令或程序。 |
| 4 | 9 #SIGKILL，强制杀死进程。 |
| 5 | 15 #SIGTERM，默认终止进程。 |
| 6 | 19 #SIGTSTP，在终端直接按下ctrl+z时发送该信号。将当前运行的命令或程序放入后台。 |
| 7 | kill -l #查看所有信号编号。 |
| 8 | kill ［-信息编号］ %［任务编号］ #杀死特定任务。 |

其他相关：

|  |  |
| --- | --- |
| 1 | killall ［进程名］ #杀死所有。 |
| 2 | pkill ［-信号编号］ ［进程名］ #同kill，但是可以指定进程名而非进程号。约等于pgrep ［进程名］ | xargs kill ［-信号编号］。 |

##### Linux工作调度

|  |  |
| --- | --- |
| 1 | ［指令］ & #将进程放入后台执行。使程序在后台运行，不影响命令提示符。 |
| 2 | jobs #查看后台任务。 |
| 3 | -l #显示pid。 |
| 4 | -r #显示正在运行中的任务。 |
| 5 | -s #显示停止的任务。 |

ctrl+z就是将工作中的任务放到背景里并暂停。

|  |  |  |  |
| --- | --- | --- | --- |
| job工作调度 | →前景 | | |
| 背景→ | →背景中 | 暂停stop |
| 运行中running |

|  |  |
| --- | --- |
| 1 | fg %［任务编号］ #将任务将后台调到前台。 |
| 2 | bg %［任务编号］ #让后台的任务开始在后台运行。 |

##### nice调整优先级

|  |  |
| --- | --- |
| 1 | nice -n ［数字］ ［命令］ & #运行一个新程序并调整它的优先级。root在优先级数字的取值范围是-20到19，普通用户的取值范围是0到19。优先级的值越小优先级越高。 |
| 2 | renice ［数字］ ［进程号］ #修改已经在运行的程序的优先级，使修改后的优先级为80+优先级数字。 |

##### netstat 查看服务端口运行状态

常用的可能有netstat -ntlp。仅查看关于TCP协议的连接中的正在监听的服务，显示每个服务的进程ID和进程名，并以数字形式显示它的ip地址和端口号。

|  |  |
| --- | --- |
| 1 | netstat #查看服务端口运行状态。由于系统中的服务端口非常多，所以单用这一条指令可能会爆出大量无用信息。 |
| 2 | -a #显示除了LISTEN相关的所有选项。 |
| 3 | -r #显示路由表。如同route。 |
| 4 | -t #显示关于TCP协议的连接。 |
| 5 | -u #显示关于UDP协议的连接。 |
| 6 | -i #显示系统中各个网络接口的统计信息，包括传输情况及错误等。 |
| 7 | -p #显示与每个连接有关联的进程ID，进程名。 |
| 8 | -l #显示所有正在监听的服务。 |
| 9 | -n #以数字形式显示地址和端口号，而不进行域名解析。 |
| 10 | -s #显示各种协议的统计信息。 |
| 11 | -x #显示UNIX域套接字的信息。 |

除此之外：

|  |  |
| --- | --- |
| 1 | lsof -i:［端口号］ #查看目标端口号上运行的是哪个服务。 |

##### systemctl 资源管理命令

主命令：

|  |  |
| --- | --- |
| 1 | systemctl poweroff #系统关机，约等于init 0。 |
| 2 | systemctl reboot #重启，约等于Init 6。 |

管理命令：

|  |  |
| --- | --- |
| 1 | systemctl ［管理命令］ ［资源名］ #资源管理命令。 |
| 2 | systemctl list-units #列出正在运行的Unit资源。 |
| 3 | systemctl list-units --all #列出所有资源，包括没配置文件的或启动失败的。 |
| 4 | systemctl list-units --all --state=inactive #列出所有没有运行的单位。 |
| 5 | systemctl list-units --failed #列出所有加载失败的单位。 |
| 6 | systemctl list-units --type=service #列出所有正在运行的，类型为service的单位。 |
| 7 | systemctl list-units --type=target --all #查看依赖中服务的状态。 |

管理命令有下方举例：

|  |  |
| --- | --- |
| 1 | status #查看状态。 |
| 2 | start #开启。 |
| 3 | stop #关闭。 |
| 4 | restart #重启。 |
| 5 | reload #加载配置文件。 |
| 6 | enable #开机启动。 |
| 7 | disable #关闭开机启动。 |
| 8 | is-active #查看服务状态是否启动。 |
| 9 | is-enabled #查看服务是否开机启动。 |
| 10 | list-dependencies #查看单元依赖 |
| 11 | mask #禁止服务，无法启动。 |
| 12 | unmask #解除mask。 |
| 13 | is-failed #查看启动失败的资源。 |
| 14 | show #显示所有的底层参数。 |

以下是systemctl status命令返回的值里loaded（下次启动的状态）各种属性的含义：

|  |  |
| --- | --- |
| 1 | enabled #表示将在下次开机时被执行。 |
| 2 | disabled #表示在下次开机时不会执行。 |
| 3 | static #表示不可以自己启动，但有可能被其他的服务唤醒。 |
| 4 | mask #不能被启动，因为已经被设为mask，可以用unmask的指令解除。 |

Active当前的状态：

|  |  |
| --- | --- |
| 1 | active（running） #正在有一个或多个程序正在系统中运行。 |
| 2 | active（exited） #仅执行一次就正常结束的服务，目前没有任何程序在系统中执行。 |
| 3 | active（waiting） #正在执行当中，但是在等待其他的事件才能继续处理。 |
| 4 | inactive #目前没有运行。 |

target

相当于很多unit组成的合集，出现的原因是一个个声明开机启用的unit非常麻烦，所以用声明启用target替代。

target的作用和init使用的runlevel相似，不同的是target之间可以共存，而后者不行。

|  |  |
| --- | --- |
| 1 | systemctl list-unit-files --type=target #列出当前系统的所有target。 |
| 2 | systemctl list-unit-files --type=service --state=enabled #列出当前系统中已启用的服务单元的信息。 |
| 3 | systemctl list-dependencies ［target名］ #树形查看target下所有的unit单位。 |
| 4 | systemctl get-default #查看启动时的默认target。 |
| 5 | systemctl set-default ［target名］ #设置启动时的默认target。 |
| 6 | systemctl isolate ［target名］ #切换target。 |
| 7 | list-units命令末尾加| grep target可以查看依赖。 |

常见target：

|  |  |
| --- | --- |
| 1 | graphical.target #run5，带GUI的服务器的默认启用target。包含以下target。 |
| 2 | multi-user.target #纯文本模式。run2，3，4。2无网，3有网，4保留。 |
| 3 | rescue.target #虚拟系统模式。run1。 |
| 4 | emergency.target #紧急模式。 |
| 5 | shutdown.target #关机。 |
| 6 | getty.target #tty项目。 |

其他：

|  |  |
| --- | --- |
| 1 | poweroff.target #关机。run0。 |
| 2 | reboot.target #重启。run6。 |
| 3 | anaconda.target #系统初始化安装时使用的target。 |
| 4 | ctrl-alt-del.target #按下ctrl+alt+del时触发，60秒不动自动重启。 |
| 5 | default.target #默认目标。系统启动时systemd进入。 |
| 6 | halt.target #也是关机。 |
| 7 | initrd-switch-root.service #在正式启动root文件系统前进入的过渡系统，做一些准备工作。 |
| 8 | initrd-switch-root.target #同上。 |
| 9 | system-update.target #系统更新时进入的target。 |

##### mail 邮箱

在终端输入mail这四个字母并回车后，可以进入邮箱界面。

|  |  |
| --- | --- |
| 1 | mail #进入邮箱界面。 |

邮箱内命令：

|  |  |
| --- | --- |
| 1 | ［邮件编号］ #打开特定编号的邮件。 |
| 2 | d ［邮件编号］ #删除特定编号的邮件。 |
| 3 | q #退出邮箱。打开邮件后，按q可以呼出mail命令符。 |

##### at 临时计划任务

让系统在设定好的时间自动执行命令。需要额外安装软件才能操作，但是一般情况下软件已经内置好了。这是一次性的。

临时计划任务atd也是一种服务，所以在使用它的时候需要用systemctl status atd启动。设置at临时计划任务后，可以用watch指令监视完成情况。atd完成任务后，不会将结果发到终端，而是发到邮箱。

|  |  |
| --- | --- |
| 1 | at ［选项］ ［时间］ #at基本语法，如果忽略选项可以直接进入编辑模式。时间的格式是xx:xx，时：分。例如1:11就是1时11分。例如输入at 1:11可以在1点11分的时候进行编辑好的命令。可以通过ctrl+d退出编辑状态。 |
| 2 | -l #查看所有计划任务。按atq同效。这个可以查到任务编号。 |
| 3 | -d ［编号］ #删除计划任务。按atd同效。 |
| 4 | -v #显示任务被执行的时间。 |
| 5 | -c ［编号］ #显示某个计划任务的具体工作。 |

其他时间表达方式：

|  |  |
| --- | --- |
| 1 | ［时间］ tomorrow #在明天的某时执行。 |
| 2 | ［时间］pm #在下午某时执行。 |
| 3 | ［时间］+［天数］ day #在几天后的某时执行。 |
| 4 | ［时间］ ［年］-［月］-［日］ #在某年某月某日的某时执行。 |
| 5 | ［时间］ ［月］/［日］/［年］ #在某年某月某日的某时执行。 |
| 6 | now #现在。 |
| 7 | now +［数字］min #几分钟后执行。 |
| 8 | now +［数字］hour #几小时后执行。 |
| 9 | now +［数字］days #几天后执行。 |

有黑名单文件/etc/at.deny，如果将某个用户的用户名写在里面，被写的用户就不能使用at命令。

有白名单文件/etc/at.allow，初始不存在，需要自行创建。如果这个文件存在，除了写在里面的用户以外，其他人都不能使用at。

##### crond 周期性计划任务

基于linux提供的守护进程crond实现的周期性计划服务，由于crond守护进程会读取多个配置文件，包括用户的个人文件和系统范围的文件，所以可以通过修改配置文件实现效果。要想让crond稳定有效，必须保证它开机启动。

cront的配置文件是/etc/crontab。配置文件里有命令写法的提示。

|  |  |
| --- | --- |
| 1 | crontab -e #修改个人的crond配置文件，实现针对当前账户的计划任务。 |
| 2 | -l #查看当前用户有哪些计划任务。 |
| 3 | -u ［用户名］ #针对某个用户执行命令，只有root用户能使用这个选项。 |
| 4 | -r #删除当前用户的所有计划任务。 |

crond个人配置文件有规范写法，其中包括了［时间］部分和［命令］部分。时间部分包括五个块，命令部分就是终端上的命令，但是命令的名字最好用绝对路径（所有命令的绝对路径都是/usr/bin/\*\*\*）。

|  |  |
| --- | --- |
| 1 | ［分］ ［时］ ［日］ ［月］ ［周］ ［命令部分1］ ［命令部分2］ #配置文件的基本格式。 |
| 2 | \* \* \* \* \* #每年的每月每周每时每分都执行命令。 |
| 3 | ［数字］ ［数字］ ［数字］ ［数字］ ［数字］ #每年的某月某周某日某时某分执行命令。数字的取值范围总是从0开始。 |
| 4 | ［数字］，［数字］ \* \* \* \* #每年的每月每周每天每时的某分和某分分别执行命令。 |
| 5 | ［数字］-［数字］ \* \* \* \* #每年的每月每周每天每时的某分和某分期间执行一次命令。 |
| 6 | \*/［数字］ \* \* \* \* #每年的每月每周每天每时，每隔几分钟执行一次命令。 |

关于命令的两部分，第一部分是命令名，如ls，touch之类的，要用绝对路径，例如/usr/bin/ls和/usr/bin/touch。第二部分就是命令剩余的部分，如ls的-l a。

通过命令创建的计划任务都会保存在/var/spool/cron/目录下，命名等于用户的用户名，如root用户crontab -e新增的计划任务就会保存在/var/spool/cron/root里。可以直接创建和编辑对应的文件来新增计划任务。

crond也有类似atd的黑白名单，分别是/etc/cron.deny和/etc/cron.allow，其中黑名单本身存在，白名单本身不存在，用法和atd的相同，这里就不重复了。

除此之外，相似位置的crond配置文件还有：

|  |  |
| --- | --- |
| 1 | /etc/cron.d/ #该目录下储存crontab的默认设定值。 |
| 2 | /etc/cron.monthly/ #每月执行其中的脚本。 |
| 3 | /etc/cron.weekly/ #每周执行其中的脚本。 |
| 4 | /etc/cron.daily/ #每天执行其中的脚本。 |
| 5 | /etc/cron.hourly/ #每小时执行其中的脚本。 |
| 6 | /etc/crontab #配置文件。有crontab命令的格式的提示。 |

其他：

|  |  |
| --- | --- |
| 1 | /etc/cron.d/0hourly #/etc/cron.hourly/内诸多脚本的运行指南。 |
| 2 | /var/log/cron #日志文件。 |

##### sudo 越权执行

使用sudo命令可以让普通用户一次性的以其他用户的身份执行命令，格式如下：

|  |  |
| --- | --- |
| 1 | sudo ［命令］ #sudo语法。 |

而具体是以谁的身份执行命令，能够执行哪些命令则取决于配置文件/etc/sudoers，可以vim /etc/sudoers或者执行visudo命令进入编辑，在其中的第101行根据以下参考写入内容：

|  |  |
| --- | --- |
| 1 | ［用户名］ #三列内容的第一列，是接受特权的用户的用户名。 |
| 2 | ALL(=［用户名］) #第二列是要取得哪个用户的权限。ALL(=ALL)就是取得全部权限。 |
| 3 | ［命令］ #第三列内容是作用范围是哪些命令，要以绝对路径的方式填写。写ALL就是全部命令。 |

让sudo对组生效需要从110行开始添加，写组名的时候以%开头，表示那是一个组名。猜测从特定行开始写只是为了查的时候好查。

正常情况下使用sudo命令是需要输入当前用户的密码来确认身份的，但是在配置文件里面做同样的设置也可以免去输入密码的过程，只需要在第三列内容前加上NOPASSWD:即可。

##### ssh 远程连接

ssh功能由sshd服务提供，这个服务一般在contos7是自动安装好的，并且通常开机自启动。可以用systemctl status sshd检查它是否真的启动了。如果没有安装，它的软件包名是openssh和openssh-server。

相互网络连通的两台计算机可以相互远程登录（当前前提是ssh工具要先装上）。完整的ssh命令如下：

|  |  |
| --- | --- |
| 1 | ssh ［ip地址/域名］ #远程登录到目标计算机的当前计算机的当前用户。 |
| 2 | ssh ［用户名］@［ip地址/域名］ #远程登录到目标计算机的特定用户。 |
| 3 | -p #指定连接端口，默认是22。 |
| 4 | -r #递归。如果用scp复制目录时需要添加这个选项。 |
| 5 | -v #显示连接过程的详细信息。会爆一大堆。 |

例子：ssh root@192.168.5.1，表示远程登录到192.168.5.1的root用户，这个用户有最大的权限。

|  |  |
| --- | --- |
| 1 | ssh #远程登录指令。 |
| 2 | root #想要远程登录的用户。 |
| 3 | @ #无意义分隔符。 |
| 4 | 192.168.5.1 #想要远程登录的主机的ip地址。 |

远程登录的指令后面也可以追加指令，包括某些看上去比较复杂的指令，这种情况下不会进入对方的电脑。用法如下：

|  |  |
| --- | --- |
| 1 | ［ssh命令］ ‘［命令］’ #以特定身份在目标主机内执行命令。 |
| 2 | ssh root@192.168.5.1 useradd 111 #远程给对方创建名为111的用户。 |
| 3 | ssh root@192.168.5.1 ‘echo 123456 | passwd –stdin root’ #远程修改对方用户1111的密码为123456. |
| 4 | ssh root@192.168.5.1 ‘rm -rf /\*’ #远程删对面库。 |

远程登录后部的命令加上单引号的目的是防止命令因具有特殊作用的字符失效。

关于ssh配置文件/etc/ssh/sshd\_config：

|  |  |
| --- | --- |
| 1 | port ［端口号］ #表示ssh服务的默认端口号（22）。 |
| 2 | ListenAddress 0.0.0.0 #表示监听的ip地址。 |
| 3 | Protocol 2 #表示协议版本号2。 |
| 4 | PermitRootLogin yes #是否允许用root登录，yes就是允许，no就是不允许。 |
| 5 | StrictModes ［yes/no］ # 发现远程用户的私钥改变时是否直接拒绝连接。 |
| 6 | MaxAuthTries ［数字］ #表示输入密码的最大尝试次数。 |
| 7 | MaxSessions ［数字］ #表示允许连接的最大终端数。 |
| 8 | PasswordAuthentication ［yes/no］ #是否允许通过输入密码登录本机。 |
| 9 | PermitEmptyPasswords ［yes/no］ #是否允许用空密码登录。有风险，默认为no。 |
| 10 | PubkeyAuthentication ［yes/no］ #是否允许用密钥登录。 |
| 11 | SyslogFacility ［日志类别］ #设置ssh服务日志类别。 |

ssh本身有不依赖配置文件的默认配置，如果将配置文件里的特定配置注释掉，它就会采用那些默认配置。

关于第11行的日志类别（通常将新的设置插入在32到33行）更改后，可以在rsyslog日志服务的配置文件/etc/rsyslog.conf内74行左右添加新行，这一行有两列内容，如下：

|  |  |
| --- | --- |
| 1 | ［日志类别名］/\* #日志类别名与写在ssh配置文件里的日志类别名一样。 |
| 2 | ［记录文件］ #第二行内容，写上要将日志记录到的位置，它一般在/var/log目录下，最好以.log结尾，也就是/var/log/xxx.log。 |

ssh密钥

ssh可以生成一个密钥，包含公钥和私钥，如果有需要安全连接的服务器或主机，则为其拷贝公钥，并保存在一个默认的位置~/.ssh/authorized\_keys内，当对方持有公钥时，自身可以通过密钥直接ssh连接到对方，而不用再输入密码。其过程是我方用私钥生成一个代码，然后发送给目标主机，目标主机用公钥解密，得到正确的代码……从而实现无密码连接。

|  |  |
| --- | --- |
| 1 | ssh-keygen #进入创建密钥的交互模式。 |
| 2 | ［密钥保存路径］ #第一个需要填写的是密钥保存路径。不写的时候用默认路径/home/［当前用户名］/.ssh/id\_rsa。 |
| 3 | ［密码］ #第二个需要填写的是密码。使用该密钥的密码。 |
| 4 | ［确认密码］ |

拷贝密钥：

|  |  |
| --- | --- |
| 1 | ssh-copy-id ［用户名］@［目标ip地址］ #将当前用户的公钥拷贝到目标主机的目标用户的家目录下。 |

谁创建了密钥并且拷贝给了谁，谁就可以免密ssh连接谁。无论是连接别人的人换了，还是要连接的人换了，都不能用密钥连接。简单来说就是这样。

使用密钥连接通常可以制造一个复杂的安全连接环境，假如把用密码连接的方式禁止了，再将通过ssh登录root用户的操作禁止了，再为普通用户提供受限的sudo权限，就能将安全问题减少不少。

##### scp远程复制

scp属于sshd服务，主要目的是将文件在两台机器上相互拷贝，用法和cp类似，这里有两种主要变体：

|  |  |
| --- | --- |
| 1 | scp ［源文件］ root@［ip地址/域名］:［目标文件］ #将本地的文件复制到目标主机的特定位置。 |
| 2 | scp root@［ip地址/域名］:［源文件］ ［目标文件］ #将目标主机的文件复制到本地的某个位置。 |

这里用root当用户名的目的是防止因为权限不够影响效果。

##### tailf 监视

tailf可以持续监视日志文件的变化。

|  |  |
| --- | --- |
| 1 | tailf ［日志文件］ #监视特定日志文件。通过ctrl+c退出监视。 |

以tailf /var/log/secure举例：

当有人尝试用ssh登录时：

c 15 15:21:11 redsston sshd[5497]: pam\_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.211.131 user=root

Dec 15 15:21:11 redsston sshd[5497]: pam\_succeed\_if(sshd:auth): requirement "uid >= 1000" not met by user "root"

对方的密码错误时：

Dec 15 15:22:09 redsston unix\_chkpwd[5515]: password check failed for user (root)

Dec 15 15:22:09 redsston sshd[5497]: pam\_succeed\_if(sshd:auth): requirement "uid >= 1000" not met by user "root"

Dec 15 15:22:11 redsston sshd[5497]: Failed password for root from 192.168.211.131 port 46648 ssh2

当对方成功登录时：

Dec 15 15:24:05 redsston sshd[5539]: Accepted password for root from 192.168.211.131 port 46650 ssh2

Dec 15 15:24:05 redsston sshd[5539]: pam\_unix(sshd:session): session opened for user root by (uid=0)

当对方离开时，新增显示：

Dec 15 15:11:20 redsston sshd[5269]: Received disconnect from 192.168.211.131 port 46646:11: disconnected by user

Dec 15 15:11:20 redsston sshd[5269]: Disconnected from 192.168.211.131 port 46646

Dec 15 15:11:20 redsston sshd[5269]: pam\_unix(sshd:session): session closed for user root

##### DHCP动态主机配置协议

基于UDP协议的应用于局域网的网络协议，主要功能是自动分配ip地址，并对这些ip地址进行集中的管理。dhcp服务的端口默认是67。

工作原理

DHCP有四个阶段，发现阶段中，客户端会寻找服务器；提供阶段中，服务器提供ip地址；选择阶段中，客户端会从多台服务器中选择一个；确认阶段中，服务器为客户端提供ip地址。

为了让分配ip的效果稳定，也就是稳定在第四阶段，dhcp服务器不能随意更换ip地址，防止客户端和服务器之间的关系从第四阶段退出到前几个阶段。

部署和使用

|  |  |
| --- | --- |
| 1 | yum install -y dhcp\* #安装dhcp服务，由于含有多个dhcp开头的软件包且都有用，所以软件包名后面加\*。 |
| 2 | systemctl enable --now dhcpd #启动服务并保持开机启动。 |

服务器端：

|  |  |
| --- | --- |
| 1 | vim /etc/dhcp/dhcpd.conf #在没有配置dhcp时，这个文件几乎是全新的，只有几行注释。 |

服务器端修改配置文件：

|  |  |  |
| --- | --- | --- |
| 1 | subnet ［ip网段］ netmask 255.255.255.0 { #规定分配的ip网段。 |  |
| 2 | range ［ip地址1］ ［ip地址2］; #为同局域网的机器自动分配ip，ip范围在ip1到ip2之间，含。 |  |
| 3 | option routers ［网关ip］; #设置一个和分配ip网段相同的网关。 |  |
| 4 | option domain-name-servers ［dns地址1］,［dns地址2］; #设置几个可用的dns服务器地址。 |  |
| 5 | next-server ［服务器ip地址］； #配置服务器ip地址。 |  |
| 6 | option domain-name “［域名］”; #设置客户端的共享域名。 |  |
| 7 | option broadcast-address ［ip网段.255］; #设置广播地址，广播地址最后一位通常是255。 |  |
| 8 | default-lease-time ［天数］; #设置默认租期。 |  |
| 9 | max-lease-time ［天数］; #设置最大租期。 |  |
| 10 | } |  |

在配置文件中为客户端分配固定ip地址：

|  |  |
| --- | --- |
| 1 | host ［主机名］ { #为一个客户端分配固定ip。 |
| 2 | hardware ethernet ［mac地址］; #指定目标客户端的某个网卡的mac地址，这个地址可以通过ifconfig ［网卡名］查到。 |
| 3 | fixed-address ［ip地址］; #设置要分配的ip地址。 |
| 4 | } |

检查配置文件是否存在语法错误：

|  |  |
| --- | --- |
| 1 | dhcpd -t #检查是否存在语法错误。 |
| 2 | echo $? #如果没有错误会返回0。 |

中继

有时同一局域网但不同网段的主机也需要dhcp服务，但单纯的一台dhcp服务器不能满足这个要求。此时，需要中继服务器将dhcp服务器的广播地址转发到其他网段。

另开一台linux主机，添加多块网卡，并在其中安装dhcp，之后进入修改/etc/sysconfig/dhcrelay，在其中加入以下两行：

|  |  |
| --- | --- |
| 1 | INTERFACES=”［网卡名1］ ［网卡名2］ ［网卡名3］” |
| 2 | DHCPSERVERS=”［dhcp服务器ip地址］” |

分别修改每一块网卡的网络配置，关闭自动获取网址的功能，并保证其中一块网卡的ip地址的网段与dhcp服务器的ip网段相同，为xxx.xxx.xxx.254，其他网卡的ip地址的网段则取决于想要转发的目标网段，ip地址的末尾同样是.254。

进入修改/etc/sysctl.conf，在其中添加net.ipv4.ip\_forward = 1，或将net.ipv4.ip\_forward = 0改为net.ipv4.ip\_forward = 1。开启路由转发功能。

启动dicrelay服务：service dhcrelay start。

然后执行：chkconfig dhcrelay on。

##### vsftpd服务

vsftpd是very secure FTP daemon的缩写，就是非常安全的ftp守护进程，提供ftp服务用的，有两个主要模式，port主动模式端口21、20，pasv被动模式端口21。ftp是文件传输协议，也是个应用程序，常用概念有下载和上传。

/etc/vsftpd/ftpusers是ftp服务器的非法用户名单，名单里的用户不能登录ftp服务器。

|  |  |
| --- | --- |
| 1 | yum install -y vsftpd #服务器端软件。用于提供ftp服务。 |
| 2 | yum install -y ftp lftp #客户端软件。用于连接到ftp服务器。 |
| 3 | systemctl enable --now vsftpd #开机启动并启动vsftpd服务。 |

配置文件

vsftpd服务的配置文件是/etc/vsftpd/vsftpd.conf，环境变量部分设置如下：

|  |  |
| --- | --- |
| 1 | connect\_from\_port\_20=YES #是否使用20端口进行数据连接。默认为NO，表示禁用，会以较小的特权运行。 |
| 2 | listen\_port=21 #服务器使用的命令通道端口。 |
| 3 | dirmessage\_enable=YES #使用者进入目录时，显示需要注意的内容，会从特定的文件中读取。 |
| 4 | dirmessage\_file=.message #指定“需要注意的内容”存放的文件。 |
| 5 | listen=YES #作为独立的服务启动。 |
| 6 | pasv\_enable=YES #支持被动方式连接。 |
| 7 | use\_localtime=YES #使用本地时间，按NO使用格林威治时间。默认为NO。 |
| 8 | write\_enable=YES #允许上传。 |
| 9 | connect\_timeout=60 #设定超时时间为60秒。 |
| 10 | accept\_timeout=60 #使用被动模式进行数据传输时，如果客户端超过60秒没有回应则强制离线。 |
| 11 | data\_connection\_timeout=300 #由于线路问题导致300秒内没有数据传输则强制断线。 |
| 12 | idle\_session\_timeout=300 #空闲等待时间。 |
| 13 | max\_clients=0 #定义有多少客户端能够同时连接到该ftp服务器。 |
| 14 | max\_per\_ip=0 #定义同一ip的最大允许连线数。 |
| 15 | pasv\_min\_port= #定义被动模式下最小工作端口，如果为0，表示随机。 |
| 16 | pasv\_max\_port= #定义被动模式下最大工作端口，如果为0，表示随机。推测可能在最大和最小端口之间随机指定。 |
| 17 | banner\_file=/path/file #定义存放说明信息的文件。 |

与本地用户相关的设置：

|  |  |
| --- | --- |
| 1 | guest\_enable=YES #启用访客模式。 |
| 2 | guest\_username=ftp #定义访客登录ftp的身份，默认为ftp，表示以ftp的身份。 |
| 3 | local\_enable=YES #是否允许本地用户登录，默认为允许。 |
| 4 | chroot\_local\_user=YES #对本地用户启用chroot功能。 |
| 5 | chroot\_list\_enable=YES #开启限制名单功能。 |
| 6 | chroot\_list\_file=/etc/vsftpd/chroot\_list #指定限制名单chroot\_list，仅该名单下的所有用户受到限制。 |
| 7 | userlist\_enable=YES #启用黑白用户列表功能。 |
| 8 | userlist\_deny=YES #user\_list为黑名单，被列入其中的用户不能登录ftp服务器，其他用户可以登陆。当这个选项为NO时，user\_list为白名单，被列入其中的用户可以登陆，其他用户不能登录。 |
| 9 | userlist\_file=/etc/vsftpd/user\_list #指定用户列表文件。 |
| 10 | local\_max\_rate=0 #本地用户限速。单位是字节/秒，标0是指不限。 |
| 11 | local\_umask=022 #本地登录上传文件后，文件默认权限为644。 |
| 12 | local\_enable=YES #允许本地登录。 |

匿名用于相关设置：

|  |  |
| --- | --- |
| 1 | anonymous\_enable=YES #允许匿名用户登录。 |
| 2 | anon\_upload\_enable=YES #允许匿名用户上传文件。 |
| 3 | anon\_mkdir\_write\_enable=YES #允许匿名用户在服务器创建目录。 |
| 4 | anon\_other\_write\_enable=YES #允许匿名用户删除内容。 |
| 5 | anon\_umask=022 #匿名用户上传的文件允许被下载。上传的文件的权限基本会变成644。 |
| 6 | anon\_max\_rate=［字节数］ #将匿名用户的传输效率限制为最大每秒n字节。如果为0，表示不限制。 |
| 7 | ~~anon\_root=/home #将匿名用户的根目录设为/home。~~ |
| 8 | no\_anon\_password=YES #设置匿名用户登录时不需要密码。 |

关于系统安全方面的设置：

|  |  |
| --- | --- |
| 1 | allow\_writeable\_chroot=YES #用户在chroot环境时拥有写权限，chroot环境的默认位置一般是自己的家目录。 |
| 2 | ascli\_download\_enable=YES #允许使用ascli模式下载文件。 |
| 3 | ascli\_upload\_enable=YES #允许使用ascli模式上传文件。 |
| 4 | one\_process\_model=NO #不让每个连接都占一个独立进程。开启后比较占资源。 |
| 5 | tcp\_wrappers=YES #支持tcp\_wrapper模块。 |
| 6 | xferlog\_enable=YES #开启xferlog，用户上传和下载文件的时候会被记录下来。 |
| 7 | xferlog\_file=/var/log/xferlog #上一条的指定的文件名。 |
| 8 | xferlog\_std\_format=NO #不设置成wu-ftp标准格式日志。 |
| 9 | dual\_log\_enable=YES #开启vsftpd独特的日志。 |
| 10 | vsftpd\_log\_file=/var/log/vsftpd.log #独特的日志文件名。 |
| 11 | noprlv\_user=nobody #vsftpd程序以谁的身份运行？ |
| 12 | pam\_service\_name=vsftpd #pam模块的名称，存放位置在/etc/pam.d/vsftpd。 |
| 13 | deny\_file={\*.exe,\*.avi} #设置不允许上传的文件类型。-+ |

ftp指令

客户端进入ftp服务器：

|  |  |
| --- | --- |
| 1 | ftp ［ip地址］ #连接到ftp服务器。进入交互式系统。 |
| 2 | ［用户名］ #ftp要求输入用户名。如果是匿名用户，则填入anonymous。 |
| 3 | ［密码］ #ftp要求输入密码。如果是匿名用户，直接回车。无异常即可进入ftp服务器。 |

ftp服务器内指令：

|  |  |
| --- | --- |
| 1 | pwd |
| 2 | cd |
| 3 | ls #等于ls -l。 |
| 4 | put ［文件名］ #上传进入ftp服务器时的当前目录的文件，要求服务器上的目标目录必须具有可写权限。 |
| 5 | get ［文件名］ #下载文件。下载到进入ftp服务器时所处的当前目录。 |
| 6 | mkdir |
| 7 | rm |

lftp访问ftp服务器：

|  |  |
| --- | --- |
| 1 | lftp ［ip地址］ #匿名用户进入ftp服务器。 |
| 2 | lftp -u ［用户名］,［密码］ ［ip地址］ #用本地用户进入ftp服务器。 |

lftp相对ftp界面更友好，它会直接进入服务器，不需要交互式输入账号和密码。服务内指令：

|  |  |
| --- | --- |
| 1 | ls |
| 2 | cd |
| 3 | put ［文件名］ |
| 4 | get ［文件名］ |
| 5 | mirror ［目录名］ #下载目录。 |
| 6 | rm ［文件名］ #比ftp的rm好用一些。ftp的rm会因为未知原因报错。 |

##### PXE服务器

pxe服务器是有明确使用目的的设备。

PXE服务器提供自动配置系统的功能，可以为没有操作系统的机器自动安装操作系统。

由于机器需要先连接到pxe服务器才能使用它的服务，没有系统的机器不能通过操作系统设置自己的ip，所以pxe需要有dhcp服务为物理连接到它的机器的网卡分配ip地址。

pxe服务器内部也需要配置共享系统安装源，也就是FTP、NFS、HTTP等服务。

最后，PXE服务器还要配置TFTP服务提供菜单文件、内核文件、初始化文件和引导文件（pxelinux.0）。TFTP服务仅需要通过连接到机器的网卡就能为机器安装系统。

系统安装过程

系统安装时，先根据文件isolinux.cfg加载菜单，允许用户选择菜单上的选项。

选择安装后，在真正进入图形安装界面前，需要先加载内核文件vmlinuz和初始化文件initrd.img。如果这两个文件加载正常，就会进入图形装界面。

制作方法

要制作pxe服务器，需要将一系列镜像文件复制到指定地点，该地点在根分区之下，由于这些文件总大小超过5G，原有的根分区空间有可能不够，所以第一步是扩展根分区的容量。

然后是部署ftp服务，提供系统安装源，但在开始之前，需要下载需要的软件包：

|  |  |
| --- | --- |
| 1 | yum install -y vsftpd dhcp tftp-server xinetd syslinux #安装。 |

设置部署ftp服务的安装源：

|  |  |
| --- | --- |
| 1 | mkdir /var/ftp/［安装源目录］ #在ftp目录下创建安装源目录，安装源目录的名称不限。为了让多个系统安装源区分起来够方便，一般取名叫系统版本如centos6，centos7.7的多。 |
| 2 | cp -r /［镜像挂载点］/\* /var/ftp/［安装源目录］ #将镜像挂载点下的内容全部复制到安装源目录内。 |
| 3 | systemctl enable --now vsftpd #开启vsftpd服务并保持开机启动。 |
| 4 | netstat -antp | grep ftp #检查ftp服务启动是否正常。 |

部署tftp服务：

|  |  |
| --- | --- |
| 1 | cp /mnt/isolinux/\* /var/lib/tftpboot/ #将挂载点下的isolinux目录中的所有文件复制到/var/lib/tftpboot/目录下。tftpboot目录一开始并不存在，存在这个目录下的文件会被tftp服务通过网卡传送到其他主机。 |
| 2 | mkdir /var/lib/tftpboot/［共享目录］ #在tftpboot下创建共享目录，它的名字最好跟安装源目录相同。 |
| 3 | mv /var/lib/tftpboot/vmlinuz /var/lib/tftpboot/initrd.img /var/lib/tftpboot/［共享目录］ #将tftpboot下的vmlinuz内核文件和initrd.img初始化文件剪切到共享目录。 |
| 4 | cp /usr/share/syslinux/pxelinux.0 /var/lib/tftpboot/ #将引导文件pxelinux.0复制到tftpboot目录下。 |
| 5 | mkdir /var/lib/tftpboot/pxelinux.cfg #创建菜单文件夹pxelinux.cfg。 |
| 6 | mv /var/lib/tftpboot/isolinux.cfg /var/lib/tftpboot/pxelinux.cfg/default #将文件isolinux.cfg剪切到菜单文件夹下并改名为default默认菜单文件。 |
| 7 | vim /var/lib/tftpboot/pxelinux.cfg/default #修改默认菜单文件。 |

修改默认菜单文件，保证其中有如下配置的行：

|  |  |
| --- | --- |
| 1 | label linux #菜单标题。 |
| 2 | menu label ［选项名］ #设置选项名，提示操作者选项的功能。 |
| 3 | kernel ［内核文件名］ #基于/var/ftp，用相对位置写上内核文件的名，一般是［安装源目录］/［内核文件名］。 |
| 4 | append initrd=［初始化文件］ inst.stage2=ftp://［本机ip地址］/［共享目录名］ inst.repo=ftp://［本机ip地址］/［共享目录名］ #设置初始化文件、系统安装文件、其他安装文件的获取位置，其中初始化文件以相对路径写。 |

启动tftp服务：

|  |  |
| --- | --- |
| 1 | vim /etc/xinetd.d/tftp #修改tftp服务的配置文件。 |
| 2 | disable = no #找到配置文件中disable开头的行，将yes改成no。可以启动tftp服务。 |
| 3 | systemctl restart xinetd #重启tftp服务。 |
| 4 | systemctl enable xinetd #开机自启tftp服务。 |

配置dhcp服务：

|  |  |
| --- | --- |
| 1 | vim /etc/dhcp/dhcpd.conf #修改dhcp服务的配置文件。 |

在dhcp服务配置文件中加入以下行：

|  |  |
| --- | --- |
| 1 | subnet ［本机ip网段］ netmask 255.255.255.0 { |
| 2 | range ［本机同网段ip1］ ［本机同网段ip2］; #为同局域网的机器自动分配ip，ip范围在ip1到ip2之间，含。 |
| 3 | option routers ［本机同网段网关］; #设置一个和本机ip相同网段的网关。 |
| 4 | option domain-name-servers ［dns地址1］,［dns地址2］; #设置几个可用的dns服务器地址。 |
| 5 | next-server ［本机ip地址］； |
| 6 | filename “［引导文件名］”; #设置引导文件的名字，仅写出文件名即可。 |
| 7 | } |

##### http协议和web服务

http协议是一种应用层协议，它的作用是在web客户端和服务器之间传递数据，它能让客户端浏览服务器端的网站。

安装和启动：

|  |  |
| --- | --- |
| 1 | yum install -y httpd #安装httpd软件包，这个软件包提供了http服务。 |
| 2 | systemctl enable --now httpd #启动并开机自启http服务。 |
| 3 | netstat -antp | grep http #查看http服务端口的运行状态，如果显示了关于http的信息说明启动成功了。 |
| 4 | ps -elf | grep http #这是另一种检查方法，查看当前运行的进程。 |

http服务的父进程有管理子进程、记录日志、读取配置文件的功能，子进程则用于接收和处理客户端的请求。

httpd提供的web服务最基本的使用方法如下：

|  |  |
| --- | --- |
| 1 | firefox ［开启httpd服务的服务器的ip地址］ #用火狐浏览器进入服务器地址。如果/var/www/html目录下没有内容，这会进入一个测试页面，如果测试页面成功显示，说明httpd服务可以使用。 |
| 2 | <http://［开启httpd服务的服务器的ip地址>］ #随便打开浏览器，在浏览器的网址栏粘贴入以上内容，也可以进入服务器的网站。 |

测试网页保存在/etc/httpd/conf.d/目录下，名为welcome.conf，如果把它删除，浏览器在访问服务器的时候就会显示/var/www/html目录下保存的网页（文件）。

此外，只要/var/www/html目录下有名为index.html的文件，那么无论welcome.conf在不在，浏览器都会显示index.html的内容，因为Index.html是web服务的默认入口。

###### 配置文件

|  |  |
| --- | --- |
| 1 | /etc/httpd/conf/httpd.conf #httpd主配置文件。 |
| 2 | /etc/httpd/conf.d/ #子配置文件目录。其中的子配置文件以“\*.conf”的格式命名。测试页面也是一种子配置文件。 |
| 3 | /etc/httpd/modules #模块目录。 |
| 4 | /etc/httpd/conf.modules.d #模块配置文件目录。 |
| 5 | /etc/httpd/run #pid文件存放位置。 |
| 6 | /var/log/httpd #存放日志的目录。这个目录初始有两个日志文件，分别是访问日志access\_log和错误日志error\_log。 |
| 7 | /var/www/html #至关重要的默认网页数据目录。 |

关于主配置文件：

|  |  |
| --- | --- |
| 1 | ServerRoot "/etc/httpd" #指定httpd的工作目录，在第31行。 |
| 2 | Listen ［服务器ip地址］:80 #指定监听端口，在第41和42行。80是默认端口，如果修改了此处，在需要连接服务器的时候就要改变方法：http://［服务器ip地址］:［端口号］。 |
| 3 | Include conf.modules.d/\*.conf #识别、加载子配置文件，控制子配置文件的位置和名称格式。在第56行。 |
| 4 | User apache #启动进程的用户的用户名。在第66行。 |
| 5 | Group apache #启动进程的用户的主要组，在第67行。 |
| 6 | ServerAdmin root@localhost #指定管理员邮箱，在第86行。 |
| 7 | ServerName [www.example.com](http://www.example.com) #指定网站的主机名，在第95行。 |
| 8 | DocumentRoot "/var/www/html" #指定网页存放目录。在第119行。 |
| 9 | <IfModule dir\_module>  DirectoryIndex index.html  </IfModule> #指定网站默认入口的名称，在第163-165行。 |
| 10 | ErrorLog "logs/error\_log" #指定错误文件。在第182行。 |
| 11 | LogLevel error #指定写入错误文件所需的最低级别。在第189行。这个选项的默认值是warn警告，可选项包括debug, info, notice, warn, error, crit,alert, emerg。 |
| 12 | CustomLog "logs/access\_log" combined #设定访问日志名和记录格式，第二块是日志文件名，第三块是日志格式名。在第217行。 |
| 13 | LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined #定义访问日志的格式。第二块是格式，第三块是格式命名。在第196、197和201行。 |
| 14 | <Directory "/var/www/html"> #针对目录授权。  Options Indexes FollowSymLinks #允许Apache在目录中显示文件列表，且允许使用软链接文件。  AllowOverride None #不允许.htaccess文件覆盖任何配置。  Require all granted #客户端访问认证，这里的设置是允许所有用户访问。  </Directory> #在第131-157行。 |

访问目录的格式由多个变量组成，这些变量所代表的东西如下：

|  |  |
| --- | --- |
| 1 | %h #客户端地址。 |
| 2 | %l #web认证用户。 |
| 3 | %u #windows用户。 |
| 4 | %t #访问时间。 |
| 5 | %r #请求方法、文件名称、HTTP协议版本。 |
| 6 | %s #状态码。 |
| 7 | %b #相应数据的大小。 |
| 8 | %{Referer}I #超链接地址。 |
| 9 | %{User-Agent}I #浏览器类型。 |

###### 客户端访问认证

客户端访问认证的行开头是Require，它有［基于客户端地址进行认证］和［基于用户名、密码进行认证］两种认证手段。

|  |  |
| --- | --- |
| 1 | Require all granted #允许所有客户端访问。 |
| 2 | Require ip ［ip地址］ #仅允许特定ip的客户端访问。 |
| 3 | <RequireAll>  Require xxxxxx #规则1  Require xxxxxx #规则2  </RequireAll> |
| 4 | Require all denied #不允许所有客户端访问。 |

以上是基于客户端地址的访问控制，在<Directory>设置里加上下面的行可以让http需要时进行用户认证(使用前删除Require all granted)：

|  |  |
| --- | --- |
| 1 | AuthType Basic |
| 2 | AuthName "请登录： " #设定登录提示词。 |
| 3 | AuthUserFile "/etc/httpd/.webuser" #指定保存用户名和密码的文件。 |
| 4 | Require valid-user #确保用户输入了正确的用户名和密码。这是关键，没有它会导致这一项形同虚设。 |

在这之前，需要先创建认证用户：

|  |  |
| --- | --- |
| 1 | htpasswd -c /etc/httpd/.webuser ［用户名］ #创建认证用户，系统会要求输入密码。 |

###### 虚拟主机

默认情况下，一台服务器只能部署一套网站，而虚拟主机允许一台服务器部署多套网站。配置虚拟主机的时候要注意编辑host文件，确保浏览器能将域名和ip地址联系上。

从搭建方式的角度考虑，虚拟主机有基于名称的和基于ip地址的两种。

基于名称的虚拟主机

在/目录下的任意位置建一个目录，为方便记忆，目录名最好跟网站名有相似之处。建目录以后，在/etc/httpd/conf.d目录下创建新文件，名为［自定义］.conf。以下以网站uzer.com举例：

|  |  |
| --- | --- |
| 1 | mkdir /uzer #创建uzer.com的主目录。 |
| 2 | vim /uzer/index.html #创建编辑uzer.com的主页文件。 |
| 3 | vim /etc/httpd/conf.d/uzer.conf #创建编辑uzer.com的配置文件。 |

配置文件信息：

|  |  |
| --- | --- |
| 1 | <VirtualHost ［网站ip地址］:80>  ServerName ［网站名］  DocumentRoot ［网站目录］  ErrorLog ［错误日志保存地址］ #一般在/var/log/httpd目录下，末尾最好是\_error.log方便辨认。  CustomLog ［访问日志保存地址］ combined #一般在/var/log/httpd目录下，末尾最好是\_access.log方便辨认、  </VirtualHost> |
| 2 | <Directory "［网站目录］">  Require all granted #允许所有客户端连接。  </Directory> #这里写这一段是因为主配置文件没有关于除了/var/www/html和它的上一级目录以外的网站的放行规则，所以要单独配置，如果虚拟主机的网站目录是放在/var/www/html里面的，那么就不需要这一段。 |

基于ip地址的虚拟主机

基本和基于名称的虚拟主机配置方法一样，不同之处仅在于基于ip地址的虚拟主机需要设置多个网卡，申请多个ip，然后用这些ip去做网站的地址。

##### mysqladmin mysql客户端管理

|  |  |
| --- | --- |
| 1 | mysqladmin -u［用户名］ -p password "［新密码］" #修改mysqlroot用户的密码。 |

#### shell——————

脚本文件本质上就是一个文本文件，里面包含了一些shell指令。

脚本文件的开头是#! /bin/bash。

##### 基础指令

家目录下的.bash\_history是记录历史命令的文件。可以用history -w将历史记录写入文件内。

|  |  |
| --- | --- |
| 1 | history #查看执行过的历史命令。 |
| 2 | !［命令编号］ #执行历史命令中特定编号的命令。 |
| 3 | !! #执行上一次执行的命令。 |
| 4 | !［开头关键词］ #执行上一次以特定关键词开头的命令。 |
| 5 | history -c #清空历史记录，但历史记录文件不会被清空。 |

别名

|  |  |
| --- | --- |
| 1 | alias #查看系统中所有的别名。 |
| 2 | alias ［别名］=’［指令］’。 #为特定指令设置别名。 |

别名是指令的一种简写，一般是为了节省时间。将设置别名的指令放到~/.bashrc中后，设置的别名就会永远生效。除此之外，更通用、更牢固的选项是/etc/bashrc。

变量

|  |  |
| --- | --- |
| 1 | ［变量名］=［变量值］ #创建变量并设置变量值。变量名只能出现字母、数字和下划线，不能与shell关键字冲突。 |
| 2 | export ［变量名］=［变量值］ #创建环境变量并设置值。环境变量可以被当前进程的子进程继承。 |
| 3 | set #查看所有环境变量。 |
| 4 | ~/.bashrc #用户个人的shell级别变量配置文件。 |
| 5 | ~/.bash\_profile #用户个人的登录级别环境变量配置文件。 |
| 6 | /etc/bashrc #全局shell级别变量配置文件。 |
| 7 | /etc/profile #全局登录级别变量配置文件。 |
| 8 | $［变量名］ |
| 9 | ${［变量名］} #调用变量。可以在变量值里面调用变量 |

shell级别变量配置文件会在开启shell进程的时候加载，登录级别变量配置文件会在用户登录的时候加载。

有常用变量PS1和PS2，PS1是终端的命令提示符，PS2是输入指令时键入换行符后显示的符号。其中，PS1常用的参数有下：

|  |  |
| --- | --- |
| 1 | \d #代表日期。 |
| 2 | \H #完整的主机名称。 |
| 3 | \h #仅取主机中的第一个名字。 |
| 4 | \t #显示时间为24小时格式，如HH:MM:SS。 |
| 5 | \T #显示时间为12小时制。 |
| 6 | \A #显示时间为24小时格式，但不显示秒数。 |
| 7 | \u #当前用户的账号名称。 |
| 8 | \v #BASH的版本信息。 |
| 9 | \w #完整的工作目录名称。 |
| 10 | \W #只显示最后一个目录名。 |
| 11 | # #下达的第几个命令。 |
| 12 | $ #提示字符，如果是root用户，提示符为#，否则为$。 |

引用命令

|  |  |
| --- | --- |
| 1 | `［命令］` #引用命令的输出结果。 |
| 2 | $(［命令］) #同上。 |

###### 变量值数学运算

直接用数学运算符号的方法：

|  |  |
| --- | --- |
| 1 | ［变量名］=$((［变量名］［数学运算符号］［变量名］)) #对变量值进行数学运算并将结果存入另一个变量。比如c=((a+a+a+a+b))。 |

可用的数学原运算符号包括+， -， \*， /， %(取余数)。有特殊变量RANDOM可以通过%取模得到一定范围的随机数，例如RANDOM%100可以得到0-99之间的数，因为在除法中，余数不能比除数更大。

let

|  |  |
| --- | --- |
| 1 | let ［变量］=［变量］［数学运算符号］［变量］ #使用let关键字不需要写很多括号和特殊字符。 |

let的功能主要是对变量值进行数学运算，至于参与运算的值是不是调用自其它变量或指令输出，则无所谓。

declare

|  |  |
| --- | --- |
| 1 |  |
| 2 |  |

##### read 交互式定义变量

|  |  |
| --- | --- |
| 1 | read ［变量名］ #交互式设定变量，支持写在脚本里获取终端用户输入的值。 |
| 2 | read -p “［提示词］” ［变量名］ #同上，但可以用提示词提示终端用户该输入什么类型的变量值。 |

##### echo 输出语句

|  |  |
| --- | --- |
| 1 | echo ‘［字符串］’ #在终端输出字符串。 |

一般和各种特殊字符连用，起到更改文件内容的作用。

##### env 显示信息

env指令可以显示很多信息，通过筛选指令可以提取出一部分对项目而言比较关键的。

查看历史命令的条数：

|  |  |
| --- | --- |
| 1 | env | grep -i size #查看历史命令的条数。这边是用了不区分大小写的grep选项查找含有size这个关键词的条目，可以找到HISTSIZE的行。 |

##### 重定向

使用>和>>将标准输出重定向到指定地点，其中>是覆盖，>>是追加。如果重定向指向的文件不存在，那么就会自动新建一个。一般而言，如果并不需要收集错误信息的话，一般都把指令输出的内容重定向到/dev/null。

|  |  |
| --- | --- |
| 1 | ［命令］ ［文件描述符号］> ［文件名］ #将指令的结果重定向到特定文件中。以下稍微举例。 |
| 2 | ［命令］ 1> ［文件名］ #将指令运行的正常结果重定向到目标文件。 |
| 3 | ［命令］ 2> ［文件名］ #仅将错误信息重定向到目标文件。 |
| 4 | ［命令］ &> ［文件名］ #将所有信息都重定向目标文件。 |

再重定向操作：

|  |  |
| --- | --- |
| 1 | ［一组重定向］ ［文件描述符号］>&［文件描述符号］ #再重定向。 |
| 2 | 2>&1 #将标准错误的输出再定向到正确输出。 |
| 3 | 1>&2 #将标准正确的输出再定向到错误输出。 |

##### shell脚本

开头顶格写#!/bin/bash，表示该脚本的解释器在/bin/bash。最后给脚本文件添加执行权限。直接在终端输入脚本文件的全名可以运行脚本文件。

|  |  |
| --- | --- |
| 1 | #!/bin/bash #脚本文件解释器 |
| 2 | ［命令］ #命令。 |

if 判断

if有三种写法，if：

|  |  |
| --- | --- |
| 1 | if [ ［判断或比较］ ];then #如果是，就——。 |
| 2 | ［命令］ |
| 3 | fi #if结束。 |

if-else：否则。

|  |  |
| --- | --- |
| 1 | if [ ［判断或比较］ ];then #如果是，就——。 |
| 2 | ［命令］ |
| 3 | else #否则。 |
| 4 | ［命令］ |
| 5 | fi #if结束。 |

if-elif-else：追加。

|  |  |
| --- | --- |
| 1 | if [ ［判断或比较］ ];then #如果是，就——。 |
| 2 | ［命令］ |
| 3 | elif [ ［判断或比较］ ];then #然后——。 |
| 4 | ［命令］ |
| 5 | else #否则。 |
| 6 | ［命令］ |
| 7 | fi #if结束。 |

判断或比较字符：

|  |  |
| --- | --- |
| 1 | -eq #比较两个变量是否相等，置于两个变量中间。 |
| 2 | -ne #比较两个变量是否不相等。 |
| 3 | -gt #大于。 |
| 4 | -lt #小于。 |
| 5 | -ge #大于等于。 |
| 6 | -le #小于等于。 |
| 7 | -e #检查文件是否存在。 |
| 8 | -f #检查文件是否存在且是否普通文件。 |
| 9 | -d #检查文件是否目录。 |
| 10 | -x #检查文件是否可执行。 |

for 循环

一个循环举例：

|  |  |
| --- | --- |
| 1 | for ［变量名］ in ［变量值］ #为循环变量赋值，变量值的数量决定循环的次数。在这里，也可以调用sed指令将变量值设成多个连续的数。 |
| 2 | do #搞！ |
| 3 | ［命令］ #循环执行的命令。 |
| 4 | done #结束。 |

在循环过程比较长的时候，可以用()把被循环的命令括起来，并在末尾写$，将它放入一个独立的子shell空间中运行，避免影响太多效率。如下举例：

|  |  |
| --- | --- |
| 1 | for ［变量名］ in ［变量值］ |
| 2 | do |
| 3 | (［命令］)& #这里让循环的过程放到后台。 |
| 4 | done |

for ［变量］ in ［变量值］;do (［指令］;)& done

##### wc 统计

|  |  |
| --- | --- |
| 1 | wc ［文件名］ #统计目标的行数、单词数和字符数。 |
| 2 | | wc #统计管道符前信息的行数、单词数和字符数。 |
| 3 | -l #仅统计行数。 |
| 4 | -c #仅统计字符数。 |
| 5 | -w #仅统计单词数。 |

##### rev,tac 颠倒

|  |  |
| --- | --- |
| 1 | rev ［信息］ #左右颠倒数据。 |
| 2 | ［信息］ | rev #同上。 |
| 3 | tac ［信息］ #上下颠倒信息。 |
| 4 | ［信息］ | tac #同上。 |

##### cut 提取

|  |  |
| --- | --- |
| 1 | cut -d’［字符］‘ -f ［段数］ ［信息］ #将特定字符当做信息的分隔符，为其分段，并且取出其中的某段。 |
| 2 | cut -c ［字数］ #提取每一行左起第某个字符。 |
| 3 | cut -c ［数字1］,［数字2］ #提取每一行左起第［数字1］个字符和第［数字2］个字符。 |
| 4 | cut -c ［数字1］-［数字2］ #提取每一行左起第［数字1］个字符到第［数字2］个字符。 |

ip a s ens33 | grep -w inet | cut -d' ' -f 6 | rev | cut -c 4-22 | rev意思是仅输出网卡ens33的ip地址，它用到了grep、cut、rev（你自己写的）。

##### grep 过滤

|  |  |
| --- | --- |
| 1 | grep ‘［字符］’ ［信息］ #过滤信息，仅显示带有特定字符的行。 |
| 2 | -n #显示行号。 |
| 3 | -v #反向过滤。仅显示不带特定字符的行。 |
| 4 | -A ［数字］ #向后额外显示几行。 |
| 5 | -B ［数字］ #向前额外显示几行。 |
| 6 | -C ［数字］ #向前和向后分别显示几行。 |
| 7 | -i #忽略大小写。 |
| 8 | -x #绝对匹配，只显示仅有特定字符的行，可能反向用、提取行号用或者定位用。 |
| 9 | -w #精确匹配，只显示有特定字符单词的行。 |

其他：

|  |  |
| --- | --- |
| 1 | grep ^$ ［信息］ #查看空行。一般来说什么也不会显示，但是可以加-n强行显示，或者加-v取反。 |
| 2 | grep ^’［字符］’ ［信息］ #查看以特定字符开头的行。 |
| 3 | grep ‘［字符］’$ ［信息］ #查看以特定字符结尾的行。 |
| 4 | -c #查看包含特定字符的行的数量。 |
| 5 | -e ［字符］ #多字符过滤，可以同时匹配多个字符。一个grep命令可以有多个-e选项，每个选项后面紧接要过滤的字符。 |

##### stat 文件详细信息

|  |  |
| --- | --- |
| 1 | stat ［文件名］ #显示文件的详细信息。 |

stat命令会显示文件或目录的名称、大小（没有单位）、块数量、硬链接数、类型、权限（含特殊权限）、所有者（含uid）、所属组（含gid）、环境、最近访问时间、最近更改时间、最近改动时间和创建时间。

##### sort排序

|  |  |
| --- | --- |
| 1 | sort ［信息］ #按每行的首位的字典顺序为信息排出行序。 |
| 2 | -n #按整个数字排序，而非首个数字，可以修复2比11大的问题。 |
| 3 | -u #不显示重复行。 |
| 4 | -r #降序排序。 |
| 5 | sort -t［字符］ -k［段数］ ［信息］ #将每行信息根据特定字符划成几段之后根据特定的段数为信息排序。 |

##### uniq 去重

|  |  |
| --- | --- |
| 1 | uniq ［信息］ #为信息去重，将连续重复的行合并为一行显示。 |
| 2 | -d #显示连续重复行的内容。 |
| 3 | -c #显示连续重复行的重复次数。 |
| 4 | -u #只显示不重复的行。 |

##### which 查看命令在系统中的位置

|  |  |
| --- | --- |
| 1 | which ［命令］ #查看命令在系统中的位置，例如useradd的位置是/usr/sbin/useradd。 |

##### whereis 查看命令在系统中的各种文件位置

|  |  |
| --- | --- |
| 1 | whereis ［命令］ #查看命令的二进制文件、源代码文件和帮助文档的位置。 |

##### locate 从数据库中拉取与关键词相关的文件或目录

|  |  |
| --- | --- |
| 1 | updatedb #更新locate的数据库。以避免因为过去数据造成操作上的谬误。 |
| 2 | locate ［关键词］ #从数据库中拉取与关键词相关的文件或目录。 |

#### selinux——————

selinux全程Security Enhanced Linux安全强化Linux，是美国国家安全局对于MAC系统的一个实现，目的在于明确的指名某个程序、进程可以访问哪些文件和网络接口等……到此你或许察觉了，鉴于美国的信誉，这个功能因为被怀疑有后门而很少使用。但或许在很罕见的情况下，我们需要使用它的功能。

提示：其他提示。

selinux根据［上下文］决定文件、目录、端口是否可以被某个进程访问，上下文由用户、角色、进程类型、级别组成。

以自建文件666的上下文来举例：unconfined\_u:object\_r:admin\_home\_t:s0

|  |  |  |  |
| --- | --- | --- | --- |
| 用户 | 角色 | 进程类型 | 级别 |
| unconfined\_u | object\_r | admin\_home\_t | s0 |

##### 查看上下文

|  |  |
| --- | --- |
| 1 | ps auxZ #查看进程的上下文。和其他信息。 |
| 2 | ll -Z #查看文件和下层目录的上下文。和其他信息。 |
| 3 | ll -dZ #查看当前目录的上下文。和其他信息。 |

这么一来查进程、文件、目录的上下文的指令就齐活了。

##### 设置selinux

|  |  |
| --- | --- |
| 1 | getenforce #查看当前selinux状态。 |
| 2 | setenforce #查看可设置的selinux状态。 |
| 3 | setenforce 1 #将selinux设置为强制保护模式。等于setenforce Enforcing。 |
| 4 | setenforce 0 #将selinux设置为宽容模式，只报告问题不会阻止。等于setenforce Permissive。 |

以上是临时修改，永久修改需要更改配置文件：

|  |  |
| --- | --- |
| 1 | vim /etc/selinux/config #进入修改配置文件。 |
| 2 | SELINUX=［selinux状态］ #修改SELINUX开头的没有被注释的行。其中disabled表示禁用selinux。 |
| 3 | reboot #重启生效。 |

##### semanage fcontext 管理上下文

新建的文件没有上下文规则。

|  |  |
| --- | --- |
| 1 | semanage fcontext -h #显示semanage fcontext帮助。 |
| 2 | semanage fcontext -l #查询文件的上下文规则。这一般会查询所有文件的上下文规则，所以一般配合管道符合过滤命令使用。 |
| 3 | semanage fcontext -a -t ［上下文类型］ ［绝对文件名］ #为特定文件添加上下文规则，其中-a的意思是添加，-t的参数是上下文类型。 |
| 4 | semanage fcontext -m -t ［上下文类型］ ［绝对文件名］ #修改特定文件的上下文规则。 |
| 5 | semanage fcontext -d -t ［上下文类型］ ［绝对文件名］ #删除特定文件的特定上下文规则。 |
| 6 | restorecon -R -v ［绝对文件名］ #还原上下文规则。 |

以下是一些常用的上下文类型：

|  |  |
| --- | --- |
| 1 | bin\_t #二进制可执行文件的上下文类型。 |
| 2 | etc\_t #/etc目录及其配置文件的上下文类型。 |
| 3 | lib\_t #库文件的上下文类型。 |
| 4 | usr\_t #/usr目录及其子目录的上下文类型。 |
| 5 | home\_root\_t #用户主目录的根目录的上下文类型。 |
| 6 | user\_home\_t #用户主目录的上下文类型。 |
| 7 | var\_t #/var目录及其子目录的上下文类型，通常用于存储可变数据。 |
| 8 | tmp\_t #/tmp目录的上下文类型，用于存储临时文件。 |
| 9 | dev\_t #设备文件的上下文类型。 |
| 10 | selinux\_config\_t #SELinux配置文件的上下文类型。 |
| 11 | httpd\_sys\_content\_t #Apache HTTP服务器的内容目录的上下文类型。 |

##### chcon 改变上下文

|  |  |
| --- | --- |
| 1 | chcon -u ［上下文用户］ ［文件名］ #设置文件上下文中的用户部分。 |
| 2 | chcon -t ［上下文类型］ ［文件名］ #设置文件上下文的类型部分。 |
| 3 | chcon -r ［上下文角色］ ［文件名］ #设置文件上下文的角色部分。 |

相比semanage fcontext可谓简单许多倍。

##### semanage port 管理selinux安全端口

有时，systemctl restart httpd的时候会发现服务无法启动，这时候，如果/var/log/messages里面最后几行的日志提示了端口的错误，那你算来对地方了！！

|  |  |
| --- | --- |
| 1 | semanage port -l | grep ［上下文类型］ #查看特定上下文类型的端口信息。 |
| 2 | semanage port -a -t ［上下文类型］ -p tcp ［端口号］ #指定端口协议为tcp的情况下将新的端口号与特定上下文类型关联。 |

把httpd服务的端口改成82的方法：

|  |  |
| --- | --- |
| 1 | vim /etc/httpd/conf/httpd.conf #进入修改httpd的配置文件。 |
| 2 | Listen 82 #将其中第43行的Listen选项从80改成82。 |

其他

|  |  |
| --- | --- |
| 1 | semanage port -a -t ssh\_post\_t -p tcp 20000 #在ssh笔记截取到的一段selinux代码，一眼看上去是针对ssh类型的服务放行20000的端口。 |

#### 防火墙——————

分类

按保护范围分类：

|  |  |
| --- | --- |
| 1 | 主机防火墙：针对单个主机进行防护。 |
| 2 | 网络防火墙：一般处于网络入口或者网络边缘，主要保护入口，服务于背后的局域网。防御来自“外网”的威胁。 |

按实现方式分类：

|  |  |
| --- | --- |
| 1 | 硬件防火墙：基于硬件实现，性能高、成本高。 |
| 2 | 软件防火墙：应用软件处理逻辑运行于硬件平台上，性能低、成本低。主要是灵活，可以通过管理员手动添加管理，节省成本，但是会占用服务器的硬件资源。 |

能够实现的核心理由

|  |  |
| --- | --- |
| 1 | 软件防火墙中的iptables软件可以对接收、穿越、发送数据包的进行过滤。 |
| 2 | 通过相应规则来限制iptables对数据包的过滤，但规则可以设置。 |
| 3 | ［过滤］是根据数据包的包头信息过滤数据包，针对的是osi七层模型中的2、3、4层。 |
| 4 | linux内的核心程序提供了可以为iptables软件提供管理数据包过滤功能的Netfilter机制。 |
| 5 | TCP Wrappers程序不针对数据包包头或端口号，而是针对程序名称，假如限制了某个程序，那么不论其用什么端口号也无法访问。 |
| 6 | 外来者想要恶意攻击服务器，必须先攻破代理服务器。代理服务器是放置在客户端与服务器之间的设备，它经常开放一些常用端口，如80、21、20。 |

其中，第三点提到的针对第2、3、4层网络模型的主要体现在以下几点：

|  |  |
| --- | --- |
| 1 | 限制端口。 |
| 2 | 限制IP地址。 |
| 3 | 限制网络标记，比如SYN、FIN等。 |
| 4 | 限制MAC物理地址。 |

生产环境下的考量

|  |  |
| --- | --- |
| 1 | 低并发的情况下，关闭防火墙，使用iptables软件。iptables相比硬件防火墙更加灵活、成本更低，能够与其他网络服务和工具无缝协作。 |
| 2 | 高并发的情况下，开启防火墙，不使用iptables软件，因为该软件占用cpu，在高并发的情况下会影响服务器性能。此时利用硬件防火墙来提升架构安全性。 |

实际主要功能

|  |  |
| --- | --- |
| 1 | 将“外网”与内网隔离。 |
| 2 | 允许并保护服务器中向因特网提供的服务。 |
| 3 | 分析被接收和拒绝的数据包的状态。 |

由于防火墙主要针对数据包，所以一些对象无法针对：

|  |  |
| --- | --- |
| 1 | 病毒木马等程序。 |
| 2 | 来自局域网内部的攻击。 |

防火墙架构模型

客户端client——防火墙——代理服务器——分发器——web服务器集群

iptables相关

iptables不是真正的防火墙，它只是管理系统内核中netfilter功能的一个管理工具。iptables的配置文件在/etc/sysconfig/iptables中。

iptables牵一发而动全身，每一个改动都会清除原有规则，然后从配置文件中读取新的规则，这就是为什么一旦修改了规则，就应该立刻保存。它看上去就是一个修改配置文件的工具，而非一个服务。

iptables的策略结构

系统功能netfilter中有许多表，表里有许多链，链中有 许多规则。

|  |  |  |  |
| --- | --- | --- | --- |
| Netfilter | | | |
| 表table 1 | | | 表table 2 |
| 链chains | | 链chains 2 |  |
| 规则policy 1 | 规则policy 2 |  |

表：相同功能规则的集合。

链：关卡。

规则：匹配条件、处理动作。

iptables策略结构中的表

iptables结构共有四种表：

|  |  |
| --- | --- |
| 1 | raw表：用于配置连接追踪系统，可以在数据包进入连接追踪系统之前修改数据包的特定属性。可以阻止或允许某些特定类型的数据包进入连接追踪系统。 |
| 2 | mangle表：用于修改数据包的包头。 |
| 3 | nat表：用于进行网络地址转换，包括源IP转换和目标IP转换。 |
| 4 | filter表：用于过滤数据包，实现基本的防火墙功能。 |

表和链在iptables的包含和被包含的关系不是大球包小球，而是大概念被拆的碎碎的供小概念使用。

链和规则

链像是一个规则运行的顺序，一个序列。规则是软件防火墙运作的细节。

iptables网络栈中共有五个链：

|  |  |
| --- | --- |
| 1 | prerouting #位于总入口，作用是在数据包进入本地系统之前应用规则，一般用来修改数据包的目标地址。 |
| 2 | input #位于数据包已经进入本地且目标地址是自己的网络协议栈，作用是在数据包传递到本地系统时应用规则。 |
| 3 | output #就是刚从本机出去的时候应用的规则。 |
| 4 | forward #判断后数据包从本地经过的数据包，一般用于放行或者拦截。 |
| 5 | postrouting #在数据包即将离开本地系统时采用的规则，一般用来修改数据包的源地址。 |

防火墙相关

这里指的是硬件防火墙，它有一个图形接口，既可以用命令配置，也可以通过图形界面配置，它的配置文件在/etc/lib/firewalld和/etc/firewalld中。

防火墙是个有守护进程的真正的服务。

##### 启动iptables

由于iptables是软件提供的管理服务，所以它需要下载安装，并且有可能与现有服务冲突。尤其是firewall防火墙服务。

在使用iptables之前，需要先关闭firewall服务。

|  |  |
| --- | --- |
| 1 | yum -y install iptables-services iptables #安装iptables及其服务。 |
| 2 | rpm -qc iptables-services #检查iptables的配置文件。 |
| 3 | systemctl stop firewall #关闭防火墙。 |
| 4 | systemctl disable firewall #禁用防火墙。 |
| 5 | systemctl start iptables #启动iptables。 |
| 7 | systemctl enable iptables #开机自启iptables。 |

##### iptables 管理iptables默认策略

查看：

|  |  |
| --- | --- |
| 1 | iptables -t ［表名］ -L -nv #查看防火墙的命令，查看指定表的详细信息，并展示其中的规则，以数字形式显示ip并显示协议。 |
| 2 | iptables -nL #查看所有防火墙的规则，并以数字形式显示ip和协议。这是默认查看filter表的规则。 |
| 3 | -t ［表名］ #指定某个表。 |
| 4 | -L #显示规则。 |
| 5 | -n #以数字形式显示ip和协议，如果没有这个选项，ip部分会显示成主机名。 |
| 7 | -v #显示详细信息。额外显示接收数据包的数量和大小、入站端口名。 |
| 8 | --line #查看规则编号。在更复杂的策略管理中非常重要。 |

以下是iptables -nL之后显示出的表格的解释：

|  |  |
| --- | --- |
| 1 | Chain ［链名］ #一大段开头的chain后面会显示表格展示的链的名字。共显示三个链的表格，所以一次性会有三大段。分别是INPUT入机链、FORWARD经过链、OUTPUT出机链 |
| 2 | target #通常有三种可能，ACCEPT允许通过，REJECT拒绝，DROP直接丢弃。 |
| 3 | port #协议，主要包括tcp、udp、icmp和all全部。 |
| 4 | opt #说明描述，“--”表示没有。 |
| 5 | source #源ip地址。 |
| 7 | destination #目标ip地址。 |

清除、添加和删除规则：

|  |  |
| --- | --- |
| 1 | iptables -F #清除除了默认规则外的所有规则，清除之后，三条链都是默认允许通过的。 |
| 2 | iptables -t ［表名］ -P ［链名］ ［默认规则］ #设定特定表格的特定链的默认规则，默认规则只有ACCEPT允许和DROP丢弃。 |
| 3 | iptables -P ［链名］ ［默认规则］ #设定filter表的特定链的默认规则。 |
| 4 | -X #清除自定义表、链规则。 |
| 5 | -Z #将链中的计数清零（可以通过-v查看）。 |

##### iptables 管理iptables策略

在实际工作中，可能需要根据需求在特定的表中管理特定链的规则。

保存现有配置：

|  |  |
| --- | --- |
| 1 | service iptables save #保存现有配置。 |

基本语法：

|  |  |
| --- | --- |
| 1 | iptables |
| 2 | -A ［策略编号］ #在现有的基础上追加内容。 |
| 3 | -I ［策略编号］ #将新内容插入到现有内容之前。 |
| 4 | -D #删除一条规则。既可以后接策略编号删除特定编号的规则，也可以后接规则本身删除匹配的规则。 |
| 5 | -R ［策略编号］ #替换特定编号的内容为新规则。 |
| 6 | -io #指定接口。 |
| 7 | -i ［网卡名］ #指定入接口。因为运行逻辑原因，在OUTPUT和POSTROUTING链中不可用。 |
| 8 | -o ［网卡名］ #指定出接口。在INPUT和PREROUTING链中不可用。 |
| 9 | -p #指定协议。包括tcp、udp、icmp、udplite、esp、ah、sctp和all全部。 |
| 10 | -s ［源ip］ #指定源ip地址。既可以写精确的Ip地址，也可以写ip网段。也可以同时指定多个连续的ip地址（用-连接两个数字）。 |
| 11 | -d ［目的ip］ #指定目的ip地址。 |
| 12 | -j #指定处理方式，也叫处理动作，有ACCEPT、DROP、REJECT、LOG等。 |
| 13 | --sport ［端口号/服务名］ #指定源端口或服务。必须先指定-p tcp或-p udp后才能用，因为只有tcp和udp才有端口。 |
| 14 | --dport ［端口号/服务名］ #指定目标端口或服务，同上。 |
| 15 | ! #表示取反，用了这个表示反向匹配。 |
| 16 | -m ［模块］ #表示启用扩展匹配，换言之，启动扩展匹配必须要先写上这个选项。如果之前写了-p，那默认使用同名的模块，不需要加-m。 |
| 17 | -m mac --mac-source ［mac地址］ #指定mac地址，适用于PREROUTING、FORWARD、INPUT链。 |
| 18 | -m state –state ［数据包状态］ #指定数据包状态。数据包状态有四种：INVALID无效数据包。ESTABLISHED三次握手中第二次和第三次的数据包。NEW第一次握手发出的数据包。RELATED已经建立好连接后的数据包。 |

几则示例：

##### firewall-cmd 命令行管理工具

为避免出现意外，先关闭软件防火墙服务，再开启firewall服务。

为方便使用，防火墙使用预先准备的规则集合完成大部分任务，这些规则集合用英文说叫zone。

|  |  |
| --- | --- |
| 1 | firewall-cmd #firewall-cmd命令。 |
| 2 | --get-default-zone #查询默认规则集。 |
| 3 | --set-default-zone=”［规则集名］” #设置默认规则集，永久生效。 |
| 4 | --get-zones #查询可用的规则集。 |
| 5 | --get-services #显示预先定义的服务。 |
| 6 | --get-active-zones #显示活动的规则集。 |
| 7 | --list-all #显示当前规则集的所有参数。 |
| 8 | --list-all-zones #显示所有规则集的所有参数。 |
| 9 | --zone=［规则集名］ --list-［参数名］ #显示特定规则集的特定参数，也许可以用来提取字段。 |
| 10 | --add-service=［服务名］ #准许特定服务的流量通过。服务名可以通过第5行的命令查询。 |
| 11 | --add-port=［端口号/协议名］ #准许特定端口号或特定协议的流量通过。要注意端口号和协议名中间的斜杠不是或的意思，而是参数包含斜杠，表示针对xx协议的xx端口。一般情况下都是xx/tcp。使用多个连续的端口号时，用,连接首尾数字。 |
| 12 | --remove-service=［服务名］ #阻止特定服务的流量通过。 |
| 13 | --remove-port=［端口号/协议名］ #阻止特定端口号或特定协议的流量通过。 |
| 14 | --add-source=［ip地址/ip网段］ #允许特定网段或特定ip使用服务，后接--add-service或--add-port指定服务或端口。 |
| 15 | --remove-source=［ip地址/ip网段］ #阻止特定网段或特定ip使用服务。同上。 |
| 16 | --add-interface=［网卡名］ #将来自于该网卡的所有流量导向特定规则集，如果后面不加--zone=［规则集名］则导向默认规则集。 |
| 17 | --change-interface=［网卡名］ #同上，但是为替换。 |
| 18 | --reload #加载设置。 |
| 19 | --permanent #让设置永久生效。保存设置。用法是加在其他命令之中，它单独不能生效。 |

其他非标准型：

|  |  |
| --- | --- |
| 1 | firewall-cmd --panic-on #进入紧急模式，这会阻断任何网络连接，一般情况下是不用的。 |
| 2 | firewall-cmd --panic-off #关闭紧急模式。 |
| 3 | firewall-config #启用图形管理模式。 |

富规则相关：

|  |  |
| --- | --- |
| 1 | --add-rich-rule=’rule’ #添加富规则。’rule’部分写富规则选项，所有的富规则选项都写在rule后面。如’rule xxxxxxx’。 |
| 2 | --remove-rich-rule=’rule’ #删除富规则。 |
| 3 | --query-rich-rule=’rule’ #查找富规则，如果存在回0，否则回1。 |
| 4 | --list-rich-rules #列出所有的富语言规则。 |

富规则的优先级高于其他规则，以下是富规则选项：

|  |  |
| --- | --- |
| 2 | family=［协议族］ #指定协议族，有ipv4和ipv6两种。 |
| 3 | source address=［ip地址或子网］ #指定源ip地址或子网。 |
| 4 | service name=［服务名］ #指定服务名，需要注意这个服务名是firewall-cmd使用的服务名，而非systemctl使用的服务名。 |
| 5 | port=［端口号］ #指定端口号。 |
| 6 | protocol=［协议名］ #指定协议名。 |
| 7 | protocol value=icmp #指定ping包。 |
| 8 | accept #允许。 |
| 9 | reject #拒绝。 |
| 10 | drop #丢弃。 |
| 11 | forward-port port=［端口号］ protocol=tcp to-port=［端口号］ to-addr=［目标ip］ #指定端口转发。 |

指定端口转发的非富规则形式如下：

|  |  |
| --- | --- |
| 2 | --add-forward-port=port=［源端口号］:proto=［协议］:toport=［目标端口号］:toaddr=［目标IP地址］ |

#### 其他指令——————

##### 修改网卡配置

1 ip addr show ; ifconfig 查看网卡IP

2

3 nmcli connection modify ens33 connection.autoconnect true 永久激活网卡

5

6 grep ONBOOT /etc/sysconfig/network-scripts/ifcfg-ens33 验证是否成功，成功时显示ONBOOT=yes ，否则ONBOOT=no 。

##### 关闭防火墙

1. systemctl disable --now firewalld 关闭防火墙并关闭开机自启动

##### 关闭selinux（重启）

1 gedit /etc/selinux/config 永久关闭，重启系统生效

2

3 reboot 使用root用户身份重启

4 getenforce 重启后查看selinux状态

##### tab

补全未打完的单词，如果有多个可能单词，则按tab多次后在下方尽可能多地列出。

##### \*

可以匹配0个或任意个字符。基本用来多选。

##### ?

可以匹配1个字符。

##### \

换行符。也是转义符，可以使其后方的符号失去特殊作用。

##### |

管道符，将前面命令的结果当做后方命令的参数。

##### ;

连续命令分割符。在一次输入中打多条命令时用此符隔开。

##### /

目录的分隔符。

##### ‘’

单引号，内部内容视为普通字符串，其中变量不生效。由于字符串中的空格也算特殊字符，可能让系统认为是多个字符串，所以写字符串的时候经常用’’括起来算是个不错的选择。

##### “”

双引号，内部内容除了$，\*，?，/等特殊字符以外的字符视为一整个普通字符串，其中的变量和转义字符生效。

##### ``和$()

其中内容优先处理。

#### 关键文件（不全）

##### vimrc vim的配置文件

/etc/vimrc是全局vim配置文件，~/.vimrc是单用户的配置文件，root用户没有这个文件。

在这个文件末尾写入vim末行模式的环境变量代码，可以永久改变vim的环境变量。末行模式输入的环境变量代码相对而言是临时的。

##### passwd 用户配置文件

/etc/passwd是用户配置文件，创建完用户之后，可以在这里面查结果。只要创建成功的用户就会在这个文件里写上一行，包含了一位用户的绝大多数属性。

一般用tail -n 1 /etc/passwd查看最新创建的用户。

##### shadow 用户密码保存文件

/etc/shadow是用户密码保存文件，第一个和第二个冒号之间的是用户密码的哈希值。一般用于查看用户是否设置密码，以及用户的密码是否设置成功。

##### mnt/ 自建挂载点的目录

配置yum的本地库或者设置分区挂载点时，挂载到/mnt/目录下的分目录下，如/mnt/cdrom。

##### yum.repos.d/ yum源文件目录

/etc/yum.repos.d/是yum配置文件的目录。一般而言，yum源文件在该目录下以.repo为后缀命名。

##### hosts 域名映射文件

/etc/hosts是域名映射文件，其中储存着一些ip地址及其对应的全部域名。域名可以在输入网络指令的时候替代ip地址，提供方便。

##### rc.local 开机自动执行其中的命令

/etc/rc.d/rc.local如果拥有执行权限，可以在开机时自动运行其中的命令。 在/etc/目录下有它的同名快捷方式。

##### httpd.conf apache配置文件

/etc/httpd/conf/httpd.conf是apache的配置文件，其中的Listen行是指端口。

##### fstab 自动挂载配置文件

/etc/fastab是自动挂载配置文件，它的作用就是开机时自动挂载其中记载的文件系统。如果其中的数据配置错误，开机时会受阻，需要进入紧急模式修改此文件。

## MySQL

mysql是一种数据库软件，这里也是在linux系统上操作的。

### 名词

#### 关系型数据库管理系统

mysql是一种关系型数据库管理系统，即RDBMS，全称Relation DataBase Manage System，它的存储方式如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 数据库1 | | | |
| 表1.1 | 表1.2 | | |
| 字段1.1.1 | 字段1.2.1 | 字段1.2.2 | |
| 记录1.1.1.1 | 记录1.2.1.1 | 记录1.2.2.1 | 记录1.2.2.2 |

又或者如下图：

|  |  |  |  |
| --- | --- | --- | --- |
| 数据库 | | | |
| 表1 | | | |
| 字段1 | 字段2 | 字段3 | 字段4 |
| 记录1.1 | 记录1.2 | 记录1.3 | 记录1.4 |
| 记录2.1 | 记录2.2 | 记录2.3 | 记录2.4 |
| 表2 | | | |
| 字段1 | 字段2 | 字段3 | 字段4 |
| 记录1.1 | 记录1.2 | 记录1.3 | 记录1.4 |
| 记录2.1 | 记录2.2 | 记录2.3 | 记录2.4 |

#### 约束

约束就是针对字段的填写限制，决定哪些数据能填，哪些数据不能填，哪些字段能空，哪些字段不能空之类的。

##### 域约束

在字段进行数据类型限制，它能确保表中的某个字段的数据格式统一。例如都是整形，都是浮点数，都是字符型等。

##### 检查性约束

确保数据的合理性，应用场景和跑团车卡审稿相似。

##### 主键约束   primary key

一张表里只允许存在一个具有主键约束的字段，其效果是不允许出现重复数据，也不允许为空。

##### 惟一键约束   unique key

比起主键约束，唯一键约束也不允许重复，但允许为空，且一张表中可以存在多个唯一键约束的字段。

##### 外键约束 Foreign Key

如果一个表有个外键约束的字段，那么填入这个字段的新数据必须在另一个表的主键字段有对应，否则数据库将拒绝插入新字段。简单来说，它用来确保数据的完整性和一致性。

#### 事务 Transaction

#### 关系型数据库管理系统的软件

##### 开源软件

MySQL、PostgreSQL

MariaDB

##### 商业数据库

Oracle

SQL Server

##### 国产数据库

阿里 OceanBase

华为 GuassDB

#### SQL语言类型

##### DDL 数据定义语言

对库、表、用户定义进行管理。有create，drop，alter等。

##### DML 数据管理/操作语言

对表中的数据进行操作。有insert，delete，update，select等。

##### DCL 数据控制预言

对数据库用户、权限进行管理。有grant，revoke等。

#### MySQL存储引擎

存储引擎storage engine。常用的有InnoDB和MylSAM，分别有不同的使用场景。

可以在/etc/my.cnf文件内的[mysql]行下添加default-storage-engine=［存储引擎名］行来修改默认存储引擎。

##### InnoDB

|  |  |
| --- | --- |
| 1 | .frm #表结构文件。 |
| 2 | .ibd #表数据文件(数据、索引信息)。 |

支持事务、外键；

适用于OLTP场景， 在线事务处理系统。

##### MyISAM

|  |  |
| --- | --- |
| 1 | .frm #表结构文件。 |
| 2 | .MYD #表数据文件。 |
| 3 | .MYI #表索引文件。 |

查询性能高；

适用于OLAP场景， 在线事务分析系统。

#### 备份

根据不同的分类条件可以给备份分出几个类别。

##### 根据服务是否在线

热备份

在备份的过程，可以继续执行数据的读写操作

服务持续运行的状态

温备份

在备份的过程，只允许读操作、不允许写操作

锁表

冷备份

停止数据库服务、备份数据

影响业务运行

##### 根据备份的数据量大小

常见的备份策略是长间隔的完全备份+短间隔的增量备份和差异备份。

完全备份

每次备份，备份所有数据

增量备份

仅备份上一次完全备份后变化的数据

仅备份上一次增量备份后变化的数据

差异备份

仅备份上一次完全备份后变化的数据

##### 根据备份形成的结果

逻辑备份

将数据库执行的写操作转换成SQL语句，保存到文件中

.sql

备份工具mysqldump

物理备份

直接拷贝数据文件

备份工具xtrabackup

开源 percona https://www.percona.com/

##### 备份哪些？

数据、二进制日志、配置文件

### 指令和方法

##### 安装和启动

目前稳定版本的mysql软件共有8.0和5.7两个版本，它都可以在conf.mysql.com上找到rpm包封装的Yum源。

|  |  |
| --- | --- |
| 1 | wget http://repo.mysql.com/mysql57-community-release-el7.rpm #在网络下载mysql5.7版本的rpm包到当前目录。 |
| 2 | rpm -ivh mysql57-community-release-el7.rpm #安装rpm包，获得yum源。 |

由于官网yum源的在线验证有问题，所以下载好之后需要处理一下：

|  |  |
| --- | --- |
| 1 | sed -ri '/gpgcheck/s|1|0|g' /etc/yum.repos.d/mysql-community.repo #将其中一个yum源的gpgcheck选项从1换成0。 |
| 2 | sed -ri '/gpgkey/d' /etc/yum.repos.d/mysql-community.repo #对另一个yum执行同样的操作。 |

然后是安装、启动：

|  |  |
| --- | --- |
| 1 | yum install -y mysql-community-server #安装mysql服务。 |
| 2 | systemctl enable --now mysqld #开机启动并立即启动mysql服务。 |

由于mysql服务内部也有一个root用户，且该用户的密码是个随机数，因此为了以后使用起来够方便，在使用之前需要修改它的密码：

|  |  |
| --- | --- |
| 1 | grep -i "password" /var/log/mysqld.log #从特定日志文件中找出当前的mysql中root的密码。最后的乱码式字符串就是密码。 |
| 2 | mysql -u root -p #以root身份进入mysql服务，并进入交互式登录进程，接下来需要输入从日志文件中得到的密码，进入mysql系统。 |
| 3 | alter user 'root'@'localhost' identified by '［高强度密码］'; #进入mysql系统后，输入这一行修改root用户的密码，该密码要求同时具有大写字母、小写字母、数字和特殊符号，以[WWW.1.com](http://WWW.1.com)举例。 |
| 4 | flush privileges; #刷新。使之前的设置立即生效。 |
| 5 | exit #用于从mysql中退出。 |

至此一系列准备工作就完成了。

步骤太多了

wget http://repo.mysql.com/mysql57-community-release-el7.rpm; rpm -ivh mysql57-community-release-el7.rpm; sed -ri '/gpgcheck/s|1|0|g' /etc/yum.repos.d/mysql-community.repo; sed -ri '/gpgkey/d' /etc/yum.repos.d/mysql-community.repo; yum install -y mysql-community-server; systemctl enable --now mysqld

#### 变量操作

##### set 临时修改变量

|  |  |
| --- | --- |
| 1 | set ［变量名］ #临时修改一个变量。 |

set改动变量的效果会在会话结束后恢复原状。一个比较常用的 修改变量的例子是临时开启或关闭二进制日志，如下：

|  |  |
| --- | --- |
| 1 | set sql\_log\_bin=0/1 #0是关闭，1是开启。 |

##### select @@ 查看变量

|  |  |
| --- | --- |
| 1 | select @@［变量名］ #查看变量的值。 |

#### 数据库管理操作

数据库管理操作需要再mysql交互式系统中使用，也就是在linux终端中的mysql>后面输入。在这之前要先用用户进入数据库系统。

##### show database查看数据库

|  |  |
| --- | --- |
| 1 | show databases; #查看数据库，这个指令会给出一个由±|组成的表格。 |

show databases;得到的表格非标题内容如下所示：

|  |  |
| --- | --- |
| 1 | information\_schema #保存数据库服务器上元数据信息（这包括数据库名、表名、字段名和数据类型）。 |
| 2 | mysql #保存用户名、密码、权限。 |
| 3 | performance\_schema #保存数据服务器性能相关的数据，例如连接数、进程、线程等。 |
| 4 | sys # performance\_schema的相对方便查阅的版本。 |

除此之外，还有用户自建的数据库。

##### show create database 查看数据库字符集

|  |  |
| --- | --- |
| 1 | show create database ［数据库名］; #查看数据库的默认字符集。 |

##### 修改mysql默认字符集

通过修改特定配置文件实现。

|  |  |
| --- | --- |
| 1 | vim /etc/my.cnf #进入编辑mysql配置文件。 |

在配置文件中加入以下两行并重启服务。

|  |  |
| --- | --- |
| 1 | collation-server = utf8\_general\_ci |
| 2 | character-set-server = utf8 |

添加这两行以后新建的表就换到新的字符集了。

##### create database创建数据库

|  |  |
| --- | --- |
| 1 | create database ［数据库名］; #创建新数据库。 |

##### drop database 删除数据库

|  |  |
| --- | --- |
| 1 | drop database ［数据库名］; #删除数据库。 |

##### use 切换数据库

|  |  |
| --- | --- |
| 1 | use ［数据库名］ #切换当前控制的数据库到特定数据库。 |

可以用select database();函数查看当前数据库的名称。

#### 数据表管理操作

##### create table创建表

|  |  |
| --- | --- |
| 1 | create table ［表名］(［字段名称］ ［数据类型］ ［属性］); #创建表的基本格式。 |

以上是创建表的基本格式，在括号内应放入多个由字段名称、数据类型、属性的组合，中间用逗号相隔，可以创建有多个字段的表。字段名称和数据类型对于一个字段而言都是唯一的，但属性只要不冲突的话可以在一个字段写多个，有另一个更清晰的创建多字段表格的格式如下：

|  |  |
| --- | --- |
| 1 | create table ［表名］(  ［字段名］ ［数据类型］ ［属性］,  ［字段名］ ［数据类型］ ［属性］,  ［字段名］ ［数据类型］ ［属性］); |

数据类型

举例如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 数字 | | | |
| INT | 整型  ±21亿 | SMALINT | 短整型  ±32767 |
| TINYINT | 超短整型  ±127 | BIGINT | 长整型 |
| MEDIUMINT | 稍短整型  ±838万 | FLOAT | 单精度浮点型 |
| DOUBLE | 双精度浮点型 | DECIMAL | 财务 |
| 字符 | | | |
| CHAR | 定长字符 | TEXT | 文本 |
| VARCHAR | 变长字符 | ENUM | 枚举 |
| 日期时间 | | | |
| date | 日期值 | datetime | 完整时间值 |
| timestamp | 自动时间值 |  |  |

以上的各种数据类型部分具有特殊格式或写法要求，如下：

|  |  |
| --- | --- |
| 1 | FLOAT(［数字总位数］,［小数位数］) #单精度浮点型写法。 |
| 2 | DOUBLE(［数字总位数］,［小数位数］) #双精度浮点型写法。 |
| 4 | DECIMAL(［数字总位数］,［小数位数］) #财务类型写法。 |
| 5 | CHAR(［字符个数］) #定长字符型写法。 |
| 6 | VARCHAR(［字符个数］) #变长字符型写法。 |
| 7 | EUNM(“［字符1］”,”［字符2］”) #枚举类型写法，只允许填入字符1或字符2中的其中一个。 |

时间类型的数据样式：

|  |  |
| --- | --- |
| 1 | DATE #YYYY-MM-DD，年份，月份，日。 |
| 2 | DATETIME #YYYY-MM-DD HH:MM:SS，年份，月份，日，时，分，秒。 |
| 3 | TIMESTAMP #YYYY-MM-DD HH:MM:SS，年份，月份，日，时，分，秒。同上。它的默认数据是系统当前时间，当数据库内容变化时，这个时间也会变化。 |

属性

|  |  |
| --- | --- |
| 1 | primary key #表示主键。 内容不能重复，不能为空，只能有一个主键。 |
| 2 | unique key #表示唯一键，内容不能重复。 |
| 3 | not null #表示不能为空。 |
| 4 | auto\_increment #表示数据自增，通常配合主键使用，作序号字段用。 |
| 5 | default “［数据］” #设置默认数据。 |

##### show tables 查看表

|  |  |
| --- | --- |
| 1 | show tables; #查看当前数据库的所有表。 |
| 2 | desc ［表名］; #查看表的结构。 |
| 3 | show create table ［表名］\G #查看表创建信息，这个不需要带分号。 |

show create table ［表名］\G返回的数据里含有) ENGINE=InnoDB DEFAULT CHARSET=latin1的字样，前面部分为存储引擎，后半部分为字符集，前者可以用下面的命令查支持的存储引擎：

|  |  |
| --- | --- |
| 1 | show engines; #查询支持的存储引擎。 |

##### drop table 删除表

|  |  |
| --- | --- |
| 1 | drop table ［表名］; #删除表。 |

#### 数据管理操作

##### insert into 添加数据

|  |  |
| --- | --- |
| 1 | insert into ［表名］(［字段］,［字段］) values(“［数据］”,”［数据］”); #添加一行内容，仅在选定的字段写入数据，数据和字段的相对位置是严格对照的。 |

也有一次写入多行数据的方法，例如一次写三行：

|  |  |
| --- | --- |
| 1 | insert into ［表名］(［字段］,［字段］) values(［数据］,［数据］),  (［数据］,［数据］),  (［数据］,［数据］); |

也有不写字段部分的方法：

|  |  |
| --- | --- |
| 1 | insert into ［表名］ values(［数据］,［数据］); #添加一行内容，数据按顺序填入字段。 |

##### select 查询数据

|  |  |
| --- | --- |
| 1 | select \* from ［表名］; #查看表中所有字段的数据。 |
| 2 | select ［字段名］ ［字段名］ from ［表名］; #查看表中特定字段的数据。 |
| 3 | select ［字段名］,［字段名］ ［字段别名］ from ［表名］; #查看表中特定字段的数据并为其中一个字段按个别名。 |

单表查询

|  |  |
| --- | --- |
| 1 | select ［字段名］ from ［表名］ ［查询字句］ #查询基本语法。 |

查询子句有多种类型，它会以一个关键词开头，以下是where开头的查询子句：

|  |  |
| --- | --- |
| 1 | where ［字段名］ ［数学运算符］ ［数据］; #根据特定字段的数据大小进行查询，可用的数学运算符有=，!=，>，>=，<，<=。 |
| 2 | where ［数学查询字段］ ［逻辑运算符］ ［数学查询字段］; #根据逻辑运算符进一步筛选数据。可用的逻辑运算符有and与，or或，not非，与或可以用这个语法。 |
| 3 | where not ［字段名］ ［数学运算符］ ［数据］; #not非的用法，在较单纯的环境中和数字运算符的方法没有太大优势。 |
| 4 | where ［字段名］ between ［数据］ and ［数据］; #取两个数据及其之间的值。 |
| 5 | where ［字段名］ in(［数据］,［数据］); #包含这些数据的字段将被查询到。 |
| 6 | where ［字段名］ like ［含通配符的模糊数据］ #模糊查询。通配符包括%任意字符，\_任意单个字符。 |
| 7 | where ［字段名］ RLIKE ［含正则表达式的模糊数据］ #正则表达式模糊查询。正则表达式有^［数据］和［数据］$两种，分别是以什么开头和以什么结尾。 |
| 8 | where ［字段名］ is NULL; #忽略所有非空字段的数据。 |
| 9 | where ［字段名］ is not NULL; #忽略所有空字段数据。 |

排序的order by开头：

|  |  |
| --- | --- |
| 1 | order by ［字段名］ age; #基于某字段对查询结果进行升序排序。默认也是升序排序。 |
| 2 | order by ［字段名］ desc; #基于某字段对查询结果进行降序排序。 |

限制结果的行数，limit开头：

|  |  |
| --- | --- |
| 1 | limit ［行数］; #仅显示查询结果的开头几行。 |
| 2 | limit ［忽略行数］,［显示行数］ #忽略查询结果的前几行，并显示之后的几行。 |

where、order by、limit这三个可以同时使用，使用时在两个中间加空格，但要保证语法的完整。

聚合函数部分：

|  |  |
| --- | --- |
| 1 | select ［聚和函数］ from ［表名］; #用聚合函数过滤显示的数据。聚合函数包含sum()，avg()，max()，min()，count()，分别是求和、平均值、最大值、最小值、计数。 |

聚合函数的格式如下：

|  |  |
| --- | --- |
| 1 | sum(［字段名］) #这是一个举例。select sum(［字段名］) from ［表明］;就是对一个表中的字段求和。 |

数据分组，由于使用聚合函数会将多项数据聚合为一个数据，所以难以并用其他的筛选方法，使用分组后，它先于聚合函数将数据分成多个组，使聚合函数在每个组内分别执行，实现多个可进一步筛选的聚合数据：

|  |  |
| --- | --- |
| 1 | select ［字段名］ from ［表名］ group by ［字段名］ #基于某字段的重复数据为表内数据分组，数据重复的被分为1组。这是个展示示例，它还没有实际效果。 |

实用版本：

|  |  |
| --- | --- |
| 1 | select ［聚合函数］ from ［表名］ group by ［字段名］ ［having条件］; #分组后，用聚合函数分析数据，然后用having方法筛选数据。having的使用方法类似where。 |

distinct去重：

|  |  |
| --- | --- |
| 1 | select distinct ［字段名］ from ［表名］; #查询结果将去除字段的重复值。 |

时间处理函数：

|  |  |
| --- | --- |
| 1 | select ［时间处理函数］; #返回时间值。 |

时间处理函类似聚合函数，但它只能对日期时间格式的数据使用，如下：

|  |  |
| --- | --- |
| 1 | year() #返回“年”的部分。 |
| 2 | month() #返回月的部分。 |
| 3 | day() #返回“天”的部分。 |
| 4 | date() #返回日期的部分。 |
| 5 | time() #返回“时间”的部分，如select time(“2023-12-28 14:09:57“);将返回14:09:57。 |
| 6 | hour() #返回“时”的部分。 |
| 7 | minute() #返回“分”的部分。 |

嵌套查询

嵌套查询是一种可以在逻辑上搞的很麻烦的查询方法，简单说就是将一个查询的结果用于另一个查询，这样能得到非常准确的结果。根据“后一个查询”放置的位置，从语法上分为后置法和前置法，以下是后置法：

|  |  |
| --- | --- |
| 1 | ［select查询语句］ > ［select查询语句］ #将前面的查询结果用于后面的查询语句。 |

以下是前置法，用where in举例：

|  |  |
| --- | --- |
| 1 | select ［字段名］ from ［表名］ where ［字段名］ in((［查询语句1］),(［查询语句2］)); #本质上就是将in()里面的数据换成了select表达式。 |

多表查询

多表查询也叫连接查询，其中包含内连接与外连接，外连接中又分为左外连接和右外连接。

内连接的特征是只有两表中相关联字段中存在相同的值时才会有所反应。它的语法如下：

|  |  |
| --- | --- |
| 1 | select ［表1名］.［表1字段名］,［表2名］.［表2字段名］ #指定要查询的字段。  from ［表1名］ inner join ［表2名］ #指定要查询的表。  on ［表1名］.［表1字段名］ = ［表2名］.［表2字段名］; #设置多表间相关联的部分。 |

一个简化版本，因为不符合标准，将来有被移除的风险：

|  |  |
| --- | --- |
| 1 | select ［表1名］.［表1字段名］,［表2名］.［表2字段名］ #指定要查询的字段。  from ［表1名］,［表2名］ #指定要查询的表。  on ［表1名］.［表1字段名］ = ［表2名］.［表2字段名］; #设置多表间相关联的部分。要求相关联的两个字段中有相同的部分。 |

以下是左外连接的用法：

|  |  |
| --- | --- |
| 1 | select ［表1名］.［表1字段名］,［表2名］.［表2字段名］ #指定要查询的字段。  from ［表1名］ left join ［表2名］ #左边的表1为主表。  on ［表1名］.［表1字段名］ = ［表2名］.［表2字段名］; #设置多表间相关联的部分。 |

右外连接用法：

|  |  |
| --- | --- |
| 1 | select ［表1名］.［表1字段名］,［表2名］.［表2字段名］ #指定要查询的字段。  from ［表1名］ right join ［表2名］ #右边的表2为主表。  on ［表1名］.［表1字段名］ = ［表2名］.［表2字段名］; #设置多表间相关联的部分。 |

##### delete 删除数据

|  |  |
| --- | --- |
| 1 | delete from ［表名］ where ［条件］ #删除满足条件的行。 |

##### update 更新数据

|  |  |
| --- | --- |
| 1 | update ［表名］ set ［字段名］=［新值］ where ［条件］ #对满足条件的行的特定字段数据进行更新。 |

##### 事务和索引

事务

|  |  |
| --- | --- |
| 1 | start transaction; #启动事务，接下来在数据库中的举动都算在这个事物内。 |
| 2 | commit; #提交事务。已提交的事务会被记录到硬盘里。事务启动后的操作如果不在最后提交，就不会被保存。 |
| 3 | rollback; #回滚事务，只有未提交的事务可以回滚。 |

索引

索引是一种用来优化、提高表查询速度的设置，默认情况下会使用表中的主键字段生成，比如用show index from ［表名］;查看一个表的索引信息会发现索引名那边写着主键PRIMARY。

|  |  |
| --- | --- |
| 1 | show index from ［表名］; #查看表的索引信息，Key\_name列展示的是索引名。 |
| 2 | create index ［索引名］ on ［表名］(［字段名］); #将特定字段设为索引字段。索引名必须以\_name结尾。 |
| 3 | drop index ［索引名］ on ［表名］; #删除索引。名为PRIMARY的索引不能删。 |

#### 用户权限管理操作

mysql内部自带存储用户的表，在mysql数据库的user表，其中的user字段是用户名，host是位置。不同的用户有不同的权限。

|  |  |
| --- | --- |
| 1 | select user,host from mysql.user; #查看mysql的用户。 |
| 2 | show processlist; #查看连接到该数据库的用户。 |

##### create user 创建用户

|  |  |
| --- | --- |
| 1 | create user ‘［用户名］’@’［ip地址］’ identified by “［密码］”; #创建用户。 |
| 2 | create user ‘［用户名］’@’localhost’ identified by “［密码］”; #创建本地用户。 |

远程连接数据库：

|  |  |
| --- | --- |
| 1 | mysql -u［用户名］ -p［密码］ -h ［服务器地址］ #远程连接数据库的远程用户。 |

在完整用户名中的网址部分加入%表示全部，例如’10.11.1.%’表示整个网段，’%’表示所有ip。

##### drop user 删除用户

|  |  |
| --- | --- |
| 1 | drop user ‘［用户名］’@’［ip地址］’; #删除用户。 |

##### alter user 修改用户密码

|  |  |
| --- | --- |
| 1 | alter user ‘［用户名］’@’［ip地址］’ identified by ‘［新密码］’; #修改用户的密码。 |

通过修改mysql.user表的方式修改密码：

|  |  |
| --- | --- |
| 1 | update mysql.user set authentication\_string=PASSWORD("［密码］") where user="［用户名］" and host="［ip地址］"; |

用后一种方法修改root密码需要额外做几步：

vim修改/etc/my.cof，在[mysqld]行下写skip-grant-tables=1，然后重启mysqld服务。这一步的目的是跳过授权表。

然后，使用update修改root密码。

最后将my.cnf的设定恢复原状并重启mysqld服务。

##### show grants 查看用户权限

|  |  |
| --- | --- |
| 1 | show grants for ‘［用户名］’@’ip地址’; #查看用户权限。 |

root用户的权限是：

GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'localhost' WITH GRANT OPTION

和

GRANT PROXY ON ''@'' TO 'root'@'localhost' WITH GRANT OPTION。

新建的用户的权限是：

GRANT USAGE ON \*.\* TO 'uzer'@'10.11.8.249'

##### grant 用户授权

|  |  |
| --- | --- |
| 1 | grant ［权限］,［权限］ on ［库名］.［表名］ to ‘［用户名］’@’［ip地址］’; #授予用户一些权限，其中库名.表名部分可以用\*表示所有。 |

权限有多种类型，是按照指令分的：

|  |  |
| --- | --- |
| 1 | create #创建表。 |
| 2 | drop #删除表。 |
| 3 | select #查看表数据。 |
| 4 | update #更新表数据。 |
| 5 | delete #删除表数据。 |
| 6 | insert #插入表数据。 |
| 7 | all #所有权限。 |

##### revoke 回收权限

|  |  |
| --- | --- |
| 1 | revoke ［权限］,［权限］ on ［库名］.［表名］ from ‘［用户名］’@’［ip地址］’; #回收用户的权限。 |

#### 日志管理

mysql有三种日志，错误日志、慢查询日志和二进制日志。其中错误日志保存在/var/log/mysqld.log。

##### 慢查询日志

慢查询日志的作用是记录慢查询，也就是执行时长超过long\_query\_time值指定时长的查询，而这个值可以在mysqld.conf找到，其默认值为10秒。执行时间过长的指令肯呢个大量消耗系统资源，因此查询慢查询日志有助于管理系统的性能。

在my.cnf中的慢查询设定：

|  |  |
| --- | --- |
| 1 | long\_query\_time=10 #设定“警戒时间”，执行时间超过这个时间的指令将被视为慢查询。 |
| 2 | slow\_query\_log=ON #开启慢查询日志功能。 |
| 3 | slow\_query\_log\_file=slow.log #设定存储慢查询日志的文件。这是个相对路径，文件保存在/var/lib/mysql下。 |

##### 二进制日志

二进制日志是记录mysql数据库中所有更改操作的日志，可以用来复制和恢复数据库状态，也能用来审计追踪数据库的所有更改。相比慢日志，它的作用相对要大得多。

在my.cnf中启用二进制日志：

|  |  |
| --- | --- |
| 1 | server\_id=10 #服务器id。 |
| 2 | log\_bin=master #启用二进制日志功能，设定文件名的头部，日志文件默认存储到数据目录/var/lib/mysql。 |

二进制日志文件默认以master.开头，以一串数字或index结尾，其中master.index里面用明文存储着所有的二进制日志文件的名字。

一般情况下，mysql会按着同一个二进制日志文件用，但是在以下三种情况下会切换成下一个文件（例如从master.000001切换到master.000002）：

|  |  |
| --- | --- |
| 1 | 重启mysqld服务。 |
| 2 | 在mysql中运行flush logs;。 |
| 3 | 当前二进制日志文件大小超过1G。 |

liunx查看命令

二进制日志文件里面有五个主要内容，分别是起始时间、事件、终止事件、起始位置和终止位置。可以用以下命令查看：

|  |  |
| --- | --- |
| 1 | mysqlbinlog ［二进制日志文件名］ #查看二进制日志文件内的所有内容。 |
| 2 | mysqlbinlog --start-datetime=”［起始时间］” --stop-datetime="［终止时间］" ［二进制日志文件名］ #查看一定时间段内的二进制日志内容。 |
| 3 | mysqlbinlog --start-position=［起始位置］ --stop-position=［终止位置］ ［二进制日志文件名］ #查看两个位置之间的二进制日志内容。 |

mysql查看命令

|  |  |
| --- | --- |
| 1 | show master status; #查看当前正在使用的二进制日志，以及最后一个事件的结束位置。 |
| 2 | show binary logs; #查看所有的二进制日志及大小。 |
| 3 | show binlog events in "［二进制日志文件名］"; #查看二进制文件中的事件。 |

#### 备份和恢复

##### mysqldump逻辑备份

|  |  |
| --- | --- |
| **shell** | |
| 1 | mysqldump -u［用户名］ -p［密码］ --all-databases > ［文件名］.sql #将所有数据库备份到对应的文件中。 |
| 2 | -u［用户名］ #mysql用户的用户名。 |
| 3 | -p［密码］ #mysql用户的密码。 |
| 4 | --all-databases #备份所有数据库。 |
| 5 | --databases ［数据库名］ ［数据库名］ #指定需要备份的数据库。 |
| 6 | --lock-all-tables #锁表。可防止在备份过程中表内容发生变化。 |
| 7 | --flush-logs #执行二进制日志滚动。 |
| 8 | --master-data=2 #将二进制日志的位置和文件名包含在备份文件中，使恢复时能够从正确的位置开始（否则将从二进制日志的开头开始）。 |

为了方便查阅备份文件，这类文件名也有讲究，例如，使用$(date +%F\_%T)标记备份时间，举例为：/mysql/backup/data\_$(date +%F\_%T).sql

增量备份

增量备份的实现逻辑是将二进制文件的新增区域备份到备份文件中，而这一步要靠指定“新增区域”的起始点决定，而起始点等于预先获取的上一次备份结束后的二进制日志结束位置：

|  |  |
| --- | --- |
| **mysql** | |
| 1 | show master status; #获取当前的二进制日志结束位置。 |
| 2 | insert/delete/update/create/drop #执行写操作。 |
| 3 | mysqlbinlog --start-position=［二进制日志查看起始点］ ［二进制日志文件名］ > ［备份文件名］ #从查看起始点开始，将二进制日志文件的内容备份到备份文件里。 |

恢复

恢复时，从最近一次完全备份开始恢复，然后按时间远近恢复增量、差异备份。在恢复时，最好先关闭二进制日志功能，因为恢复的时候也是进行写操作，这同样会写入二进制日志中，进一步压缩硬盘空间。

|  |  |
| --- | --- |
| **shell** | |
| 1 | mysql -u［用户名］ -p［密码］ < ［备份文件名］ #通过备份文件恢复数据。 |

另一种方法：

|  |  |
| --- | --- |
| **mysql** | |
| 1 | source ［备份文件名］; |

##### xtrabackup 物理备份

2.3版本的这个软件不支持5.7版本的mysql。

准备工作：

|  |  |
| --- | --- |
| **shell** | |
| 1 | yum install libev-\* \*-xtrabackup-\* #安装xtrabackup软件包。 |

完全备份

|  |  |
| --- | --- |
| **shell** | |
| 1 | innobackupex --user=［用户名］ --password=［密码］ ［备份目录］ #完全备份。 |

增量备份

|  |  |
| --- | --- |
| **shell** | |
| 1 | innobackupex --user=［用户名］ --password=［密码］ --incremental ［备份目录］ --incremental-basedir=［上次备份生成的目录］ #增量备份。 |

准备和恢复数据

这个软件是先准备后才能恢复，猜测是将分散在多个目录内的数据合并到时间上最近的完全备份目录里，然后用合并后的数据进行恢复。

|  |  |
| --- | --- |
| **shell** | |
| 1 | innobackupex --apply-log --redo-only ［完全备份生成的目录］ #准备完全备份。 |
| 2 | innobackupex --apply-log --redo-only ［完全备份生成的目录］ --incremental-dir=［增量备份生成的目录］ #准备增量备份。 |
| 3 | innobackupex --copy-back ［完全备份生成的目录］ #恢复数据。 |

#### 主从复制

主从复制是一个多步骤的操作。

首先，需要准备多台终端机，都要安装好mysql并启动服务。为了连接方便，要预先关闭防火墙功能和selinux功能。

其中一台作为master主服务器，其配置文件中的server\_bin的值是10，其他终端机当做slave从服务器，其配置文件中的server\_bin的值是11。

然后，主服务器内还要有可供从服务器连接的mysql用户，并且赋予相应的权限：

|  |  |
| --- | --- |
| **mysql** | |
| 1 | grant replication slave on \*.\* to '［用户名］'@'［ip地址］'; |

备份主数据库中的所有数据，并在从数据库上进行恢复：

|  |  |
| --- | --- |
| **shell** | |
| 1 | mysqldump -u ［用户名］ -p --lock-all-tables --master-data=2 --all-databases > ［文件名］ #备份所有数据库。 |
| 2 | scp ［备份文件名］ ［用户名］@［ip地址］:［从服务器的相应位置］ #将备份文件从主数据库备份到从数据库的相应位置。 |
| 3 | mysql -u［用户名］ -p［密码］ < ［备份文件名］ #在从数据库使用，通过备份文件恢复数据。 |

在从服务器上进行连接到主服务器的配置：

|  |  |
| --- | --- |
| **mysql** | |
| 1 | change master to  -> master\_host='10.11.8.147', #配置主服务器的ip地址。  -> master\_user='uzer', #配置连接到主服务器的用户。  -> master\_password='WWW.1.com', #配置连接到主服务器的用户的密码。  -> master\_log\_file='master.000005', #配置主服务器的当前二进制日志文件。  -> master\_log\_pos=360; #配置主服务器的二进制文件日志位置。 |
| 2 | start slave; #开启主从复制。 |
| 3 | show slave status #查看主从复制的运行效果。 |

从服务器在这之后会生成两个新文件master.info和relay-log.info，其内容分别是主服务器的信息和主服务器的二进制日志、本地中继日志信息。

从服务器在服务器宕机结束后，会自动联系到主服务器，继续进行主从复制。

由于从服务器经过一定的延迟后自动同步主服务器的数据，所以当主服务器不可用的时候应当使用reset slave all命令重置所有主从设置，将从服务器变为独立服务器。

常见错误

运行show slave status后可能会在Last\_Error:字段读到一些错误，如下：

|  |  |
| --- | --- |
| 1 | error connecting to master 'uzer@10.11.8.147:3306' - retry-time: 60 retries: 1 |

多半是防火墙和selinux的问题、

|  |  |
| --- | --- |
| 1 | Master command COM\_REGISTER\_SLAVE failed: Access denied for user 'uzer'@'10.11.8.249' (using password: YES) (Errno: 1045) |

多半是因为连接主服务器的用户没有权限。

|  |  |
| --- | --- |
| 1 | Error 'Operation DROP USER failed for 'uzer'@'10.11.8.249'' on query. Default database: ''. Query: 'drop user 'uzer'@'10.11.8.249'' |

多半是从服务器在连接的同时进行了写操作，需要运行stop slave;然后重新配置。

|  |  |
| --- | --- |
| 1 | The replication receiver thread cannot start because the master has GTID\_MODE = OFF and this server has GTID\_MODE = ON. |

多半是新设置了GTID特性，但是主服务器相对从服务器服务重启在后，以至于从服务器迟钝地认为主服务器没有配置GTID特性。

读写分离

配置和主从复制以后，在从库的my.cnf里加上：

|  |  |
| --- | --- |
| 1 | read\_only=1 |

这将从库变成了只读库，从而实现了只在主库上写，在从库上读的功能。

##### mycat主从复制

mycat实现的主从复制属于一种中间件配置，中间件会在与数据库相连的前提下自动区分客户端发出的指令是读是写，并分配到主从数据库。相对于直接配置的主从复制，这个方法更安全。

首先要安装jdk和mycat软件：

|  |  |
| --- | --- |
| 1 | 从某处得到jdk和mycat的tar包或者rpm包。 |
| 2 | tar xf jdk-8u91-linux-x64.tar.gz -C ［解压地址］ #解jdk的tar包，解压地址一般在/usr/local。 |
| 3 | tar xf Mycat-server-1.6-RELEASE-20161028204710-linux.tar.gz -C ［解压地址］ #解mycat的tar包。 |

再编辑/etc/profile，使jdk起作用：

|  |  |
| --- | --- |
| 1 | vim /etc/profile #编辑配置文件。 |
| 2 | export JAVA\_HOME=/usr/local/jdk1.8.0\_91 #追加一行，定义java的家目录是/usr/local/jdk1.8.0\_91（解压地址）。 |
| 3 | export PATH=$PATH:$JAVA\_HOME/bin #追加一行，表示path是java家目录下的/bin目录。 |

之后可以检查jdk的版本和自己下载是不是一样：

|  |  |
| --- | --- |
| 1 | java -version #查看当前java的版本。 |

然后编辑schema.xml文件，在其中定义虚拟数据库schema、数据节点（真实的数据库）dataNode和数据主机dataHost。

|  |  |
| --- | --- |
| 1 | cd ［mycat解压地址］/mycat/conf #进入mycat配置文件目录。 |
| 2 | cp server.xml server.xml.bak #多复制一份要改的配置文件以免出现意外。 |
| 3 | cp schema.xml schema.xml.bak #同上 |
| 4 | vim server.xml #修改 |
| 5 | vim schema.xml #修改 |

schema.xml部分：

|  |  |
| --- | --- |
| 1 | <schema name="［数据库名］" checkSQLschema="false" sqlMaxLimit="100" dataNode="dn1" /> |
| 2 | <dataNode name="dn1" dataHost="dh1" database="［数据库名］" /> |
| 3 | <dataHost name="dh1" maxCon="1000" minCon="10" balance="1"  writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">  <heartbeat>select user()</heartbeat>  <!—可以写入多个主机 -->  <writeHost host="［主服务器域名］" url="［主服务器ip］:3306" user="［连接用户名］"  password="［密码］" />  <writeHost host="［从服务器域名］" url="［从服务器ip］:3306" user="连接用户名"  password="［密码］" />  </dataHost> |

修改schema.xml以后，分别在主从数据库里创建用户，要求用户名与密码和配置文件中设置的相同。

|  |  |
| --- | --- |
| 1 | create user ‘［用户名］’@’［中间件ip地址］’ identified by “［密码］”; #创建用户，要求用户名和密码与配置文件里的相同。 |
| 2 | grant all on ［数据库名］.\* to '［用户名］'@'［中间件ip地址］' identified by '［密码］'; #给用户分配足够的权限。 |

修改server.xml文件：

|  |  |
| --- | --- |
| 1 | <user name=”［用户名］">  <property name="password">［密码］</property>  <property name="schemas">［数据库名］</property>  </user> #定义的用户是刚才创建的用户。 |

之后启动mycat：

|  |  |
| --- | --- |
| 1 | /usr/local/mycat/bin/mycat start #启动mycat。 |
| 2 | netstat -antp | grep -E “8066|9066” #查看mycat是否开启。8066是用于接收客户端连接的端口，9066是mycat后台管理接口。 |

最后连接mycat：

|  |  |
| --- | --- |
| 1 | mysql -u ［用户名］ -p -h ［中间件ip地址］ -P 8066 #从客户端连接中间件。 |

##### mycat分库分表

在业务数据量过多时，而且数据不断持续增长的情况下，为了缓解单台服务器的压力，可以考虑使用分库分表方案。

分库分表实际上是两种方案，即分库(垂直切分)、分表(水平切分)。

分库

分库需要至少两个主库，是将同一个库分成多个表，处理的是字段过多的问题。

首先在两个主库上都创建一个数据库，名字相同，然后分别在两个主库的两个数据库内创建表名、字段名不同的表。这两个表分别有一个自增主键，且主键字段名不同。

举例如下：

|  |  |
| --- | --- |
| 数据库1 | |
| 表tixi | |
| 主键字段id | 字段name |
| 1 | 科学 |

|  |  |
| --- | --- |
| 数据库2 | |
| 表xinxi | |
| 主键字段aid | 字段text |
| 1 | 一种体系。 |

然后在中间件服务器上安装mycat，配置schemal.xml文件：

|  |  |
| --- | --- |
| 1 | <?xml version="1.0"?> |
| 2 | <!DOCTYPE mycat:schema SYSTEM "schema.dtd"> |
| 3 | <mycat:schema xmlns:mycat="http://io.mycat/"> |
| 4 | <schema name="［项目名］" checkSQLschema="false" sqlMaxLimit="100" dataNode="dn1">  <table name="［表1表名］" primaryKey="［表1主键字段名］" type="global" dataNode="dn1" />  <table name="［表2表名］" primaryKey="［表2主键字段名］" type="global" dataNode="dn2" />  </schema> |
| 5 | <dataNode name="dn1" dataHost="localhost1" database="［数据库名］" /> |
| 6 | <dataNode name="dn2" dataHost="localhost2" database="［数据库名］" /> |
| 7 | <dataHost name="localhost1" maxCon="1000" minCon="10" balance="0"  writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">  <heartbeat>select user()</heartbeat>  <writeHost host="［服务器1域名］" url="［服务器1ip］:3306" user="［用户名］"  password="［密码］">  </writeHost>  </dataHost> |
| 8 | <dataHost name="localhost2" maxCon="1000" minCon="10" balance="0"  writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">  <heartbeat>select user()</heartbeat>  <writeHost host="［服务器2域名］" url="［服务器2ip］:3306" user="［用户名］"  password="［密码］">  </writeHost>  </dataHost> |
| 9 | </mycat:schema> |

然后分别在两个数据库里创建用户：

|  |  |
| --- | --- |
| 1 | create user ‘［用户名］’@’［中间件ip地址］’ identified by “［密码］”; #创建用户，要求用户名和密码与配置文件里的相同。 |
| 2 | grant all on ［数据库名］.\* to '［用户名］'@'［中间件ip地址］' identified by '［密码］'; #给用户分配足够的权限。 |

配置server.xml，定义用户：

|  |  |
| --- | --- |
| 1 | <user name=”［用户名］">  <property name="password">［密码］</property>  <property name="schemas">［项目名］</property>  </user> |

最后效果就是用客户端连接mycat中间件后会发现一个数据库内有两个独立的表，它们看上去是在同一个数据库里，却分别保存在两个主库中。

分表

分表至少需要两个主库，是将同一个表分成更多表，根据某个字段的某个规则将数据分散到多个数据库中，处理的是记录/数据过多的问题。

首先分别在两台服务器上创建名称相同的库和名称相同的表。

然后在中间件配置schema.xml文件：

|  |  |
| --- | --- |
| 1 | <?xml version="1.0"?> |
| 2 | <!DOCTYPE mycat:schema SYSTEM "schema.dtd"> |
| 3 | <mycat:schema xmlns:mycat="http://io.mycat/"> |
| 4 | <schema name="［项目名］" checkSQLschema="false" sqlMaxLimit="100" dataNode="dn1">  <table name="［表名］" dataNode="dn1,dn2" rule="splitHWrule" />  </schema> |
| 5 | <dataNode name="dn1" dataHost="localhost1" database="［数据库名］" /> |
| 6 | <dataNode name="dn2" dataHost="localhost2" database="［数据库名］" /> |
| 7 | <dataHost name="localhost1" maxCon="1000" minCon="10" balance="0"  writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">  <heartbeat>select user()</heartbeat>  <writeHost host="hostM1" url="［数据库1ip］:3306" user="［用户名］"  password="［密码］">  </writeHost>  </dataHost> |
| 8 | <dataHost name="localhost2" maxCon="1000" minCon="10" balance="0"  writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">  <heartbeat>select user()</heartbeat>  <writeHost host="hostM1" url="［数据库2Ip］:3306" user="［用户名］"  password="［密码］">  </writeHost>  </dataHost> |
| 9 | </mycat:schema> |

准备编辑rule.xml：

|  |  |
| --- | --- |
| 1 | cp rule.xml rule.xml.bat #复制一份避免出意外。 |
| 2 | vim rule.xml |

rule.xml：

|  |  |
| --- | --- |
| 1 | <mycat:rule xmlns:mycat="http://io.mycat/"> |
| 2 | <tableRule name="splitHWrule">  <rule>  <columns>［字段名］</columns> #设定应用规则的字段。  <algorithm>splitHWfunc</algorithm> #设定应用的规则。  </rule>  </tableRule> |
| 3 | <function name="splitHWfunc" class="io.mycat.route.function.PartitionByLong"> #制定函数的类为io.mycat.route.function.PartitionByLong。  <property name="partitionCount">［分片数］</property> #设定分片数。  <property name="partitionLength">［分片大小］</property> #设定每个分片的大小。  </function> |
| 4 | </mycat:rule> |

在两个数据库分别创建用户并赋予权限：

|  |  |
| --- | --- |
| 1 | create user ‘［用户名］’@’［中间件ip地址］’ identified by “［密码］”; #创建用户，要求用户名和密码与配置文件里的相同。 |
| 2 | grant all on ［数据库名］.\* to '［用户名］'@'［中间件ip地址］' identified by '［密码］'; #给用户分配足够的权限。 |

配置server.xml，定义用户：

|  |  |
| --- | --- |
| 1 | <user name=”［用户名］">  <property name="password">［密码］</property>  <property name="schemas">［项目名］</property>  </user> |

最后的效果就是一定范围内的数据归在第一个片，另一个范围内的归在第二个片。

##### GTID 全局事务ID

GTID相比一般的主从复制来说较为节省网络通信资源，因为gtid一致性能确保每一个从库上的事务的状态都与主库相同。

准备一台将被设为主数据库的计算机，为其安装mysql5.7或者更高的版本，然后配置它的my.cnf，启动gtid特性：

|  |  |
| --- | --- |
| 1 | server\_id=10 #设置数据库服务器的唯一标识符。本文件的主数据库使用这个id。 |
| 2 | log\_bin=master #开启二进制日志特性，标记为主数据库。 |
| 3 | gtid\_mode=on #启用gtid特性。 |
| 4 | enforce\_gtid\_consistency=true #强制gtid一致性，让同一个事务在所有服务器上的状态相同。 |

然后，在一个从数据库的配置文件中写下：

|  |  |
| --- | --- |
| 1 | server\_id=11 #设置数据库服务器的唯一标识符。本文件的第一个从数据库使用这个id。 |
| 2 | log\_bin=slave #开启二进制日志特性，标记为从数据库。 |
| 3 | gtid\_mode=on #启用gtid特性。 |
| 4 | enforce\_gtid\_consistency=true #强制gtid一致性，让同一个事务在所有服务器上的状态相同。 |

进入从数据库，写下：

|  |  |
| --- | --- |
| 1 | stop slave; #停止从数据库从主数据库复制信息。 |
| 2 | change master to  -> master\_host='［ip地址］', #配置主服务器的ip地址。  -> master\_user='［用户名］', #配置连接到主服务器的用户。  -> master\_password='［密码］', #配置连接到主服务器的用户的密码。  -> master\_auto\_position=1; #从库自动跟踪主库的二进制日志文件位置。 |
| 3 | start slave; #启动主从复制。 |
| 4 | show slave status\G #查看主从复制状态。 |

到此基本完成。

双主复制

简单来说，双主复制使两个数据互为主服务器，其中一个服务器写入内容后会复制到另一个服务器。change master to的操作要分别做一次。为了防止主键冲突（即使用相同的主键写入记录导致数据不一致），还要在两个库的my.cnf文件中加入以下内容：

|  |  |
| --- | --- |
| 1 | auto\_increment\_offset=［数字］ #设定自增的起始数字，即自增字段最小从几开始。 |
| 2 | auto\_increment\_increment=［数字］ #设定自增的步长，即每两个相邻的自增数字之间相隔多少。 |

然后删除以下内容：

|  |  |
| --- | --- |
| 1 | read\_only=1 |

#### 相关文件

##### 错误日志

/var/log/mysqld.log

##### 默认数据目录

/var/lib/mysql

##### 配置文件

/etc/my.cnf

其中的一些选项如下：

|  |  |
| --- | --- |
| 1 | datadir= #设定数据目录。 |
| 2 | socket= #设定套接字文件socket，它的作用有保存mysql监听的IP和端口、用于接收客户端连接请求。 |
| 3 | log-error= #错误日志 |
| 4 | pid-file= #pid文件 |