**防伪码：宝剑锋从磨砺出，梅花香自苦寒来。**

**1.如何判断mysql主从是否同步？该如何使其同步？**

|  |  |
| --- | --- |
| 1  2  3 | Slave\_IO\_Running  Slave\_SQL\_Running；  略 |

**2.mysql的innodb如何定位锁问题，mysql如何减少主从复制延迟？**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | mysql的innodb如何定位锁问题:  在使用 show engine innodb status检查引擎状态时，发现了死锁问题  在5.5中，information\_schema 库中增加了三个关于锁的表（MEMORY引擎）：innodb\_trx         ## 当前运行的所有事务innodb\_locks       ## 当前出现的锁innodb\_lock\_waits  ## 锁等待的对应关系  mysql如何减少主从复制延迟:  如果延迟比较大，就先确认以下几个因素：  1. 从库硬件比主库差，导致复制延迟  2. 主从复制单线程，如果主库写并发太大，来不及传送到从库，就会导致延迟。更高版本的mysql可以支持多线程复制  3. 慢SQL语句过多  4. 网络延迟5. master负载  主库读写压力大，导致复制延迟，架构的前端要加buffer及缓存层6. slave负载  一般的做法是，使用多台slave来分摊读请求，再从这些slave中取一台专用的服务器，只作为备份用，不进行其他任何操作.  另外， 2个可以减少延迟的参数:  –slave-net-timeout=seconds 单位为秒 默认设置为 3600秒  #参数含义：当slave从主数据库读取log数据失败后，等待多久重新建立连接并获取数据  –master-connect-retry=seconds 单位为秒 默认设置为 60秒  #参数含义：当重新建立主从连接时，如果连接建立失败，间隔多久后重试。  通常配置以上2个参数可以减少网络问题导致的主从数据同步延迟  MySQL数据库主从同步延迟解决方案  最简单的减少slave同步延时的方案就是在架构上做优化，尽量让主库的DDL快速执行。还有就是主库是写，对数据安全性较高，比如sync\_binlog=1，innodb\_flush\_log\_at\_trx\_commit  = 1 之类的设置，而slave则不需要这么高的数据安全，完全可以讲sync\_binlog设置为0或者关闭binlog，innodb\_flushlog也可以设置为0来提高sql的执行效率。另外就是使用比主库更好的硬件设备作为slave。 |

**3.osi七层模型，tcp三次握手过程，tcp连接断开过程，什么情况下tcp进入time\_wait?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | 答：其他问题答案略  什么情况下tcp进入time\_wait?  当关闭一个 socket 连接时，主动关闭一端的 socket 将进入TIME\_WAIT状态，而被动关闭一方则转入CLOSED状态。  具体过程如下：1、 客户端发送FIN报文段，进入FIN\_WAIT\_1状态。2、 服务器端收到FIN报文段，发送ACK表示确认，进入CLOSE\_WAIT状态。3、 客户端收到FIN的确认报文段，进入FIN\_WAIT\_2状态。4、  服务器端发送FIN报文端，进入LAST\_ACK状态。5、 客户端收到FIN报文端，发送FIN的ACK，同时进入TIME\_WAIT状态，启动TIME\_WAIT定时器，超时时间设为2MSL。6、 服务器端收到FIN的ACK，进入CLOSED状态。7、  客户端在2MSL时间内没收到对端的任何响应，TIME\_WAIT超时，进入CLOSED状态。 |

**4.什么是跨站脚本攻击，有何危害，sql注入攻击如何防范？**

|  |  |
| --- | --- |
| 1 | 答：略 |

**5.海量文件如何存储？**

|  |  |
| --- | --- |
| 1 | 答：使用分布式存储，如mfs、hadoop等 |

**6.如何重置mysql root密码？**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | 答：一、 在已知MYSQL数据库的ROOT用户密码的情况下，修改密码的方法：  1、 在SHELL环境下，使用mysqladmin命令设置：mysqladmin –u root –p password “新密码”   回车后要求输入旧密码  2、 在mysql>环境中,使用update命令，直接更新mysql库user表的数据：  Update  mysql.user  set  password=password(‘新密码’)  where  user=’root’;  flush   privileges;  注意：mysql语句要以分号”；”结束  3、 在mysql>环境中，使用grant命令，修改root用户的授权权限。  grant  all  on  \*.\*  to   root@’localhost’  identified  by  ‘新密码’；  二、 如查忘记了mysql数据库的ROOT用户的密码，又如何做呢？方法如下：  1、 关闭当前运行的mysqld服务程序：service  mysqld  stop（要先将mysqld添加为系统服务）  2、 使用mysqld\_safe脚本以安全模式（不加载授权表）启动mysqld 服务  /usr/local/mysql/bin/mysqld\_safe  --skip-grant-table  &  3、 使用空密码的root用户登录数据库，重新设置ROOT用户的密码  ＃mysql  -u   root  Mysql> Update  mysql.user  set  password=password(‘新密码’)  where  user=’root’;  Mysql> flush   privileges; |

**7.把/data目录及其子目录下所有以扩展名.txt结尾的文件中包含oldgirl的字符串全部替换为oldboy**

|  |  |
| --- | --- |
| 1  2 | 答：  find /data/ -type f -name "\*.txt" | xargs sed -i 's/oldgirl/oldboy/g' |

**8.创建目录/data/oldboy,并且在该目录下创建文件oldboy.txt,然后在文件“oldboy.txt”里写入内容“inet  addr : 10.0.0.8   Bcast : 10.0.0.255  Mask :  255.255.255.0 ”(不包含引号)**

|  |  |
| --- | --- |
| 1  2 | 答：# mkdir -p /data/oldboy  # echo "inet addr: 10.0.0.8 Bcast: 10.0.0.255 Mask:255.255.255.0" >> /data/oldboy/oldboy.txt |

**9.脚本计算1+2+3…. +100=？**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | 答：  [root@localhost ~]# cat sum.sh  #!/bin/bash  num=1  sum=0  while [ $num -le 100 ]  do     sum=$(expr $sum + $num)     let num++  done  echo "1+2+3...+100="$sum  [root@localhost ~]# sh sum.sh  1+2+3...+100=5050 |

**1.Apache和MySQL 两种服务能否装在同一台机器上，如何查看Apache和mysql端口和进程？**

|  |  |
| --- | --- |
| 1 | 答案略 |

**2.如何在一台虚拟机上同时部署4个网站，访问域名分别是 www.test1.net   www.test2.net  test1.jjwxc.net   test2.jjwxc.net**

|  |  |
| --- | --- |
| 1 | 答案略 |

**3.统计一下/var/log/nginx/access.log 日志中访问量最多的前十个IP?**

|  |  |
| --- | --- |
| 1  2  3 | cat access\_log | awk ‘{print $1}’ | uniq -c|sort -rn|head -10  或  awk '{print $1}' /var/log/nginx/access.log | sort | uniq -c | sort -nr -k1 | head -n 10 |

**4.怎么查看当前系统中每个IP的连接数，怎么查看当前磁盘的IO,怎么查看当前网络的IO?**

**答：**

**怎么查看当前系统中每个IP的连接数：**

|  |  |
| --- | --- |
| 1  2  3  4 | # netstat -n | awk '/^tcp/ {print $5}' | awk -F: '{print $1}' | sort | uniq -c| sort –rn  sort命令：进行排序，-r 反向排序 -n 使用纯数字进行排序  uniq 将重复的数据仅仅列出一个来显示，uniq -c，进行计数  awk -F: '{print $1}'  以F 为分界符，取出第一个：之前的数据 |

**怎么查看当前磁盘的IO：**

**1） iostat可以提供丰富的IO状态数据。**

**iostat 是 sysstat 工具集的一个工具，需要安装。**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | [root@localhost ~]# iostat -d -k 1 10  Linux 3.10.0-327.el7.x86\_64 (localhost.localdomain) 03/23/2017 \_x86\_64\_(2 CPU)  Device:            tps    kB\_read/s    kB\_wrtn/s    kB\_read    kB\_wrtn  sda              16.60       597.83        29.44     384048      18909  scd0              0.03         0.10         0.00         66          0  dm-0             15.78       551.54        26.20     354311      16831  dm-1              0.22         1.97         0.00       1268          0 |

**参数 -d 表示，显示设备（磁盘）使用状态；-k某些使用block为单位的列强制使用Kilobytes为单位；1 10表示，数据显示每隔1秒刷新一次，共显示10次。**

**tps：该设备每秒的传输次数。**

**kB\_read/s：每秒从设备读取的数据量；kB\_wrtn/s：每秒向设备写入的数据量；kB\_read：读取的总数据量；kB\_wrtn：写入 的总数量数据量；这些单位都为Kilobytes。**

**-x 参数：使用-x参数我们可以获得更多统计信息。**

**iostat -d -x -k 1 10 #查看设备使用率（%util）、响应时间（%await）**

**await：每一个IO请求的处理的平均时间（单位是微秒）。这里可以理解为IO的响应时间，一般地系统IO响应时间应该低于5ms，如果大于10ms就比较大了。**

**%util：在统计时间内所有处理IO时间，除以总共统计时间。例如，如果统计间隔1秒，该 设备有0.8秒在处理IO，而0.2秒闲置，那么该设备的%util = 0.8/1 = 80%，所以该参数暗示了设备的繁忙程度。一般地，如果该参数是100%表示设备已经接近满负荷运行了（当然如果是多磁盘，即使%util是100%，因为磁盘的并发能力，所以磁盘使用未必就到了瓶颈）。**

**2）使用iotop命令**

**要安装iotop软件包**

**iotop命令是一个用来监视磁盘I/O使用状况的top类工具。iotop具有与top相似的UI。Linux下的IO统计工具如iostat，nmon等大多数是只能统计到per设备的读写情况，如果你想知道每个进程是如何使用IO的就比较麻烦，使用iotop命令可以很方便的查看。**

**怎么查看当前网络的IO：**

**iftop 查看网络带宽情况(必须从epel源安装)**

**sar看看当前网络流量 ,sar -n**[**DEV**](https://www.baidu.com/s?wd=DEV&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YvPj0YuWN9uWR3uW6znW030ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHn3nWDzP1DYP1nvPH64rjTsn0)**1 999 表示取样间隔为1秒,取样999次**

**5.写一个脚本，实现批量添加20个用户，用户名为：user1-20，密码为user后面跟着5个随机字符或数字**

**答：**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | [root@localhost ~]# cat useradd.sh  #!/bin/bash  name="user"  for number in $(seq 1 20)  do     password=$(cat /dev/urandom | head -1 | md5sum | head -c 5)     useradd $name$number     echo "user$password" | passwd --stdin user$number &> /dev/null     echo "$name$number user$password" >> userinfo.txt  done |

**6.dmesg命令中看到ip-conntrack：table full，dropping packet，如何解决？**

**答：**

**线上web服务器在访问量很大时，就会出现网络连接丢包的问题，通过dmesg命令查看日志，发现如下信息：**

**ip\_conntrack: table full, dropping packet.**

**为什么会出现 ip\_conntrack: table full, dropping packet 呢？这里面关键的信息是"ip\_conntrack: table full, dropping packet",从这里可以判断出这跟iptables有关系了，因为iptables防火墙使用了ip\_conntrack内核模块实现连接跟踪功能，所有的进出数据包都会记录在连接跟踪表中，包括tcp，udp，icmp等，一旦连接跟踪表被填满以后，就会发生丢包，导致网络不稳定。当这张表满了，就会在日志里面写入该信息（执行命令dmesge | more查看）。**

**而当我们的服务器确实打开了iptables防火墙，并且都是在网站流量非常高的时候经常会出现这个问题。这个问题的原因是由于web服务器收到了大量的连接，在启用了iptables的情况下，iptables会把所有的连接都做链接跟踪处理，这样iptables就会有一个链接跟踪表，当这个表满的时候，就会出现上面的错误。**

**查看当前连接数：**

|  |  |
| --- | --- |
| 1  2 | [root@localhost ~]# cat /proc/sys/net/netfilter/nf\_conntrack\_count  0 |

**iptables的链接跟踪表最大容量配置文件如下：**

|  |  |
| --- | --- |
| 1  2 | [root@localhost ~]# cat /proc/sys/net/netfilter/nf\_conntrack\_max  65536 |

**注：**

**由于nf\_conntrack 工作在3层，支持IPv4和IPv6，而ip\_conntrack只支持IPv4，因此nf\_conntrack模块在**[**Linux**](http://www.2cto.com/os/linux/)**的2.6.15内核中被引入，而ip\_conntrack在Linux的2.6.22内核被移除（centos6.x版本），因此不同版本的**[**系统**](http://www.2cto.com/os/)**，配置文件也就不尽相同了。目前大多的ip\_conntrack\_\*已被 nf\_conntrack\_\* 取代，很多ip\_conntrack\_\*仅仅是个软链接，原先的ip\_conntrack配置目录/proc/sys/net/ipv4/netfilter/ 仍然存在，但是新的nf\_conntrack在/proc/sys/net/netfilter/中，这样做是为了能够向下的兼容。**

**这个问题该如何解决，解決方法一般有两个：**

**方法1：**

|  |  |
| --- | --- |
| 1  2  3  4  5 | [root@localhost ~]# sysctl -w net.nf\_conntrack\_max=100000  net.nf\_conntrack\_max = 100000  [root@localhost ~]# cat /proc/sys/net/netfilter/nf\_conntrack\_max  100000  sysctl-p |

**注：**

**严格来看重启 iptables/ip6tables 会重新加载 nf\_conntrack 相关的模块, 引起参数恢复成默认值。**

**以免重启 iptables 引起参数失效. 也可以在 /etc/sysconfig/iptables-config 文件中开启选项 IPTABLES\_SYSCTL\_LOAD\_LIST=”.nf\_conntrack”, iptables 重启后会进行 sysctl 操作.**

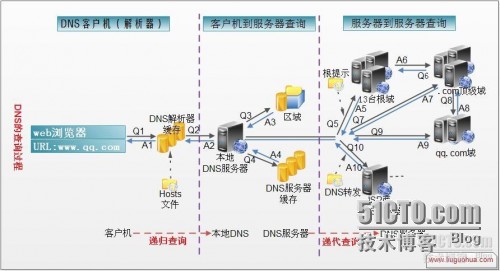
**方法2：**

**不使用ip\_conntrack，nf\_conntrack\_ipv4，xt\_state模块**

7、varnish  nginx squid各自缓存的优缺点

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | 要做cache服务的话，我们肯定是要选择专业的cache服务，优先选择squid和varnish。  Varnish  高性能、开源的反向代理服务器和内存缓存服务器。  优点：  1. 高性能；  2. 多核支持；  3. 支持0-60秒的精确缓存时间。  缺点：  1. 不具备自动容错和恢复功能，重启后数据丢失；  2. 在线扩容比较难。  3. 32位机器上缓存文件大小为最大2GB；  4. 不支持集群。  应用场景：  并发要求不是很大的小型系统和应用  nginx  1不支持带参数的动态链接  2Nginx缓存内部没有缓存过期和清理的任何机制，这些缓存的文件会永久性地保存在机器上，如果要缓存的东西非常多，那就会撑暴整个硬盘空间。  3只能缓存200状态码，因此后端返回301/302/404等状态码都不会缓存，假如恰好有一个访问量很大的伪静态链接被删除，那就会不停穿透导致后端承载不小压力  4Nginx不会自动选择内存或硬盘作为存储介质，一切由配置决定，当然在当前的操作系统里都会有操作系统级的文件缓存机制，所以存在硬盘上也不需要过分担心大并发读取造成的io性能问题。  Squid  Squid，很古老的反向代理软件，拥有传统代理、身份验证、流量管理等高级功能，但是配置太复杂。它算是目前互联网应用得最多的反向缓存代理服务器，工作于各大古老的cdn上  squid的优势在于完整的庞大的cache技术资料，和很多的应用生产环境 |

8、查询DNS的深层结构

当 DNS 客户机需要查询程序中使用的名称时，它会查询本地DNS 服务器来解析该名称。客户机发送的每条查询消息都包括3条信息，以指定服务器应回答的问题。  
● 指定的 DNS 域名，表示为完全合格的域名 (FQDN) 。  
● 指定的查询类型，它可根据类型指定资源记录，或作为查询操作的专门类型。  
● DNS域名的指定类别。  
    对于DNS 服务器，它始终应指定为 Internet 类别。例如，指定的名称可以是计算机的完全合格的域名，如im.qq.com，并且指定的查询类型用于通过该名称搜索地址资源记录。  
    DNS 查询以各种不同的方式进行解析。客户机有时也可通过使用从以前查询获得的缓存信息就地应答查询。DNS 服务器可使用其自身的资源记录信息缓存来应答查询，也可代表请求客户机来查询或联系其他 DNS 服务器，以完全解析该名称，并随后将应答返回至客户机。这个过程称为递归。  
    另外，客户机自己也可尝试联系其他的 DNS 服务器来解析名称。如果客户机这么做，它会使用基于服务器应答的独立和附加的查询，该过程称作迭代，即DNS服务器之间的交互查询就是迭代查询。  
DNS 查询的过程如下图所示。  
[](https://s3.51cto.com/wyfs02/M02/8F/1C/wKiom1jTpgnCcXHRAADb3ZnfOSM830.jpg-wh_500x0-wm_3-wmp_4-s_613059812.jpg)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | 1、在浏览器中输入www.qq.com域名，操作系统会先检查自己本地的hosts文件是否有这个网址映射关系，如果有，就先调用这个IP地址映射，完成域名解析。  2、如果hosts里没有这个域名的映射，则查找本地DNS解析器缓存，是否有这个网址映射关系，如果有，直接返回，完成域名解析。  3、如果hosts与本地DNS解析器缓存都没有相应的网址映射关系，首先会找TCP/ip参数中设置的首选DNS服务器，在此我们叫它本地DNS服务器，此服务器收到查询时，如果要查询的域名，包含在本地配置区域资源中，则返回解析结果给客户机，完成域名解析，此解析具有权威性。  4、如果要查询的域名，不由本地DNS服务器区域解析，但该服务器已缓存了此网址映射关系，则调用这个IP地址映射，完成域名解析，此解析不具有权威性。  5、如果本地DNS服务器本地区域文件与缓存解析都失效，则根据本地DNS服务器的设置（是否设置转发器）进行查询，如果未用转发模式，本地DNS就把请求发至13台根DNS，根DNS服务器收到请求后会判断这个域名(.com)是谁来授权管理，并会返回一个负责该顶级域名服务器的一个IP。本地DNS服务器收到IP信息后，将会联系负责.com域的这台服务器。这台负责.com域的服务器收到请求后，如果自己无法解析，它就会找一个管理.com域的下一级DNS服务器地址(qq.com)给本地DNS服务器。当本地DNS服务器收到这个地址后，就会找qq.com域服务器，重复上面的动作，进行查询，直至找到www.qq.com主机。  6、如果用的是转发模式，此DNS服务器就会把请求转发至上一级DNS服务器，由上一级服务器进行解析，上一级服务器如果不能解析，或找根DNS或把转请求转至上上级，以此循环。不管是本地DNS服务器用是是转发，还是根提示，最后都是把结果返回给本地DNS服务器，由此DNS服务器再返回给客户机。  从客户端到本地DNS服务器是属于递归查询，而DNS服务器之间就是的交互查询就是迭代查询。 |

9、lvs的三种模式

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | 一、NAT模式（VS-NAT）   原理：就是把客户端发来的数据包的IP头的目的地址，在负载均衡器上换成其中一台RS的IP地址，并发至此RS来处理,RS处理完成后把数据交给经过负载均衡器,负载均衡器再把数据包的原IP地址改为自己的IP，将目的地址改为客户端IP地址即可期间,无论是进来的流量,还是出去的流量,都必须经过负载均衡器    优点：集群中的物理服务器可以使用任何支持TCP/IP操作系统，只有负载均衡器需要一个合法的IP地址。    缺点：扩展性有限。当服务器节点（普通PC服务器）增长过多时,负载均衡器将成为整个系统的瓶颈，因为所有的请求包和应答包的流向都经过负载均衡器。当服务器节点过多时，大量的数据包都交汇在负载均衡器那，速度就会变慢！    二、IP隧道模式（VS-TUN）    原理：首先要知道，互联网上的大多Internet服务的请求包很短小，而应答包通常很大。那么隧道模式就是，把客户端发来的数据包，封装一个新的IP头标记(仅目的IP)发给RS,RS收到后,先把数据包的头解开,还原数据包,处理后,直接返回给客户端,不需要再经过负载均衡器注意,由于RS需要对负载均衡器发过来的数据包进行还原,所以说必须支持IPTUNNEL协议所以,在RS的内核中,必须编译支持IPTUNNEL这个选项    优点：负载均衡器只负责将请求包分发给后端节点服务器，而RS将应答包直接发给用户。所以，减少了负载均衡器的大量数据流动，负载均衡器不再是系统的瓶颈，就能处理很巨大的请求量，这种方式，一台负载均衡器能够为很多RS进行分发。而且跑在公网上就能进行不同地域的分发。    缺点：隧道模式的RS节点需要合法IP，这种方式需要所有的服务器支持”IP Tunneling”(IP Encapsulation)协议，服务器可能只局限在部分Linux系统上。    三、直接路由模式（VS-DR）    原理：负载均衡器和RS都使用同一个IP对外服务但只有DR对ARP请求进行响应,所有RS对本身这个IP的ARP请求保持静默也就是说,网关会把对这个服务IP的请求全部定向给DR,而DR收到数据包后根据调度算法,找出对应的RS,把目的MAC地址改为RS的MAC（因为IP一致）并将请求分发给这台RS这时RS收到这个数据包,处理完成之后，由于IP一致，可以直接将数据返给客户，则等于直接从客户端收到这个数据包无异,处理后直接返回给客户端由于负载均衡器要对二层包头进行改换,所以负载均衡器和RS之间必须在一个广播域,也可以简单的理解为在同一台交换机上    优点：和TUN（隧道模式）一样，负载均衡器也只是分发请求，应答包通过单独的路由方法返回给客户端。与VS-TUN相比，VS-DR这种实现方式不需要隧道结构，因此可以使用大多数操作系统做为物理服务器。    缺点：（不能说缺点，只能说是不足）要求负载均衡器的网卡必须与物理网卡在一个物理段上。 |

10、lvs/nginx/haproxy优缺点

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | Nginx的优点是：   1、工作在网络的7层之上，可以针对http应用做一些分流的策略，比如针对域名、目录结构，它的正则规则比HAProxy更为强大和灵活，这也是它目前广泛流行的主要原因之一，Nginx单凭这点可利用的场合就远多于LVS了。  2、Nginx对网络稳定性的依赖非常小，理论上能ping通就就能进行负载功能，这个也是它的优势之一；相反LVS对网络稳定性依赖比较大，这点本人深有体会；  3、Nginx安装和配置比较简单，测试起来比较方便，它基本能把错误用日志打印出来。LVS的配置、测试就要花比较长的时间了，LVS对网络依赖比较大。  3、可以承担高负载压力且稳定，在硬件不差的情况下一般能支撑几万次的并发量，负载度比LVS相对小些。  4、Nginx可以通过端口检测到服务器内部的故障，比如根据服务器处理网页返回的状态码、超时等等，并且会把返回错误的请求重新提交到另一个节点，不过其中缺点就是不支持url来检测。比如用户正在上传一个文件，而处理该上传的节点刚好在上传过程中出现故障，Nginx会把上传切到另一台服务器重新处理，而LVS就直接断掉了，如果是上传一个很大的文件或者很重要的文件的话，用户可能会因此而不满。  5、Nginx不仅仅是一款优秀的负载均衡器/反向代理软件，它同时也是功能强大的Web应用服务器。LNMP也是近几年非常流行的web架构，在高流量的环境中稳定性也很好。  6、Nginx现在作为Web反向加速缓存越来越成熟了，速度比传统的Squid服务器更快，可以考虑用其作为反向代理加速器。  7、Nginx可作为中层反向代理使用，这一层面Nginx基本上无对手，唯一可以对比Nginx的就只有lighttpd了，不过lighttpd目前还没有做到Nginx完全的功能，配置也不那么清晰易读，社区资料也远远没Nginx活跃。  8、Nginx也可作为静态网页和图片服务器，这方面的性能也无对手。还有Nginx社区非常活跃，第三方模块也很多。  Nginx的缺点是：  1、Nginx仅能支持http、https和Email协议，这样就在适用范围上面小些，这个是它的缺点。  2、对后端服务器的健康检查，只支持通过端口来检测，不支持通过url来检测。不支持Session的直接保持，但能通过ip\_hash来解决。   LVS  LVS：使用Linux内核集群实现一个高性能、高可用的负载均衡服务器，它具有很好的可伸缩性（Scalability)、可靠性（Reliability)和可管理性（Manageability)。    LVS的优点是：  1、抗负载能力强、是工作在网络4层之上仅作分发之用，没有流量的产生，这个特点也决定了它在负载均衡软件里的性能最强的，对内存和cpu资源消耗比较低。  2、配置性比较低，这是一个缺点也是一个优点，因为没有可太多配置的东西，所以并不需要太多接触，大大减少了人为出错的几率。  3、工作稳定，因为其本身抗负载能力很强，自身有完整的双机热备方案，如LVS+Keepalived，不过我们在项目实施中用得最多的还是LVS/DR+Keepalived。  4、无流量，LVS只分发请求，而流量并不从它本身出去，这点保证了均衡器IO的性能不会收到大流量的影响。  5、应用范围比较广，因为LVS工作在4层，所以它几乎可以对所有应用做负载均衡，包括http、数据库、在线聊天室等等。    LVS的缺点是：  1、软件本身不支持正则表达式处理，不能做动静分离；而现在许多网站在这方面都有较强的需求，这个是Nginx/HAProxy+Keepalived的优势所在。  2、如果是网站应用比较庞大的话，LVS/DR+Keepalived实施起来就比较复杂了，特别后面有Windows  Server的机器的话，如果实施及配置还有维护过程就比较复杂了，相对而言，Nginx/HAProxy+Keepalived就简单多了。  HAProxy  HAProxy的特点是：  1、HAProxy也是支持虚拟主机的。  2、HAProxy的优点能够补充Nginx的一些缺点，比如支持Session的保持，Cookie的引导；同时支持通过获取指定的url来检测后端服务器的状态。  3、HAProxy跟LVS类似，本身就只是一款负载均衡软件；单纯从效率上来讲HAProxy会比Nginx有更出色的负载均衡速度，在并发处理上也是优于Nginx的。  4、HAProxy支持TCP协议的负载均衡转发，可以对MySQL读进行负载均衡，对后端的MySQL节点进行检测和负载均衡，大家可以用LVS+Keepalived对MySQL主从做负载均衡。  5、HAProxy负载均衡策略非常多，HAProxy的负载均衡算法现在具体有如下8种：  ①roundrobin，表示简单的轮询，这个不多说，这个是负载均衡基本都具备的；  ② static-rr，表示根据权重，建议关注；  ③leastconn，表示最少连接者先处理，建议关注；  ④ source，表示根据请求源IP，这个跟Nginx的IP\_hash机制类似，我们用其作为解决session问题的一种方法，建议关注；  ⑤ri，表示根据请求的URI；  ⑥rl\_param，表示根据请求的URl参数’balance url\_param’ requires an URL parameter name；  ⑦hdr(name)，表示根据HTTP请求头来锁定每一次HTTP请求；  ⑧rdp-cookie(name)，表示根据据cookie(name)来锁定并哈希每一次TCP请求。 |

11、linux系统监控命令，查看cpu负载内存等情况

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | top命令是Linux下常用的性能分析工具，比如cpu、内存的使用，能够实时显示系统中各个进程的资源占用状况，类似于Windows的任务管理器。 top显示系统当前的进程和其他状况,是一个动态显示过程,即可以通过用户按键来不断刷新当前状态.如果在前台执行该命令,它将独占前台,直到用户终止该程序为止.  比较准确的说,top命令提供了实时的对系统处理器的状态监视.它将显示系统中CPU最“敏感”的任务列表.该命令可以按CPU使用.内存使用和执行时间对任务进行排序；而且该命令的很多特性都可以通过交互式命令或者在个人定制文件中进行设定。  top命令参数  d 指定每两次屏幕信息刷新之间的时间间隔。当然用户可以使用s交互命令来改变之。  p通过指定监控进程ID来仅仅监控某个进程的状态。  q该选项将使top没有任何延迟的进行刷新。如果调用程序有超级用户权限，那么top将以尽可能高的优先级运行。  S 指定累计模式  s 使top命令在安全模式中运行。这将去除交互命令所带来的潜在危险。  i使top不显示任何闲置或者僵死进程。  c 显示整个命令行而不只是显示命令名  常用操作  top   //每隔5秒显式所有进程的资源占用情况  top -d2//每隔2秒显式所有进程的资源占用情况  top -c//每隔5秒显式进程的资源占用情况，并显示进程的命令行参数(默认只有进程名)  top -p12345-p6789//每隔5秒显示pid是12345和pid是6789的两个进程的资源占用情况  top -d2-c-p123456//每隔2秒显示pid是12345的进程的资源使用情况，并显式该进程启动的命令行参数 |

三、查看网络流量的命令

|  |  |
| --- | --- |
| 1 | watch -n 1 "/sbin/ifconfig eth0 | grep bytes" |

四、docker四中网络模式

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | 1 host模式  众所周知，Docker使用了Linux的Namespaces技术来进行资源隔离，如PID Namespace隔离进程，Mount  Namespace隔离文件系统，Network Namespace隔离网络等。一个Network Namespace提供了一份独立的网络环境，包括网卡、路由、Iptable规则等都与其他的Network Namespace隔离。一个Docker容器一般会分配一个独立的Network  Namespace。但如果启动容器的时候使用host模式，那么这个容器将不会获得一个独立的Network Namespace，而是和宿主机共用一个Network Namespace。容器将不会虚拟出自己的网卡，配置自己的IP等，而是使用宿主机的IP和端口。  例如，我们在10.10.101.105/24的机器上用host模式启动一个含有web应用的Docker容器，监听tcp80端口。当我们在容器中执行任何类似ifconfig命令查看网络环境时，看到的都是宿主机上的信息。而外界访问容器中的应用，则直接使用10.10.101.105:80即可，不用任何NAT转换，就如直接跑在宿主机中一样。但是，容器的其他方面，如文件系统、进程列表等还是和宿主机隔离的。   2 container模式  在理解了host模式后，这个模式也就好理解了。这个模式指定新创建的容器和已经存在的一个容器共享一个Network  Namespace，而不是和宿主机共享。新创建的容器不会创建自己的网卡，配置自己的IP，而是和一个指定的容器共享IP、端口范围等。同样，两个容器除了网络方面，其他的如文件系统、进程列表等还是隔离的。两个容器的进程可以通过lo网卡设备通信。   3 none模式  这个模式和前两个不同。在这种模式下，Docker容器拥有自己的Network Namespace，但是，并不为Docker容器进行任何网络配置。也就是说，这个Docker容器没有网卡、IP、路由等信息。需要我们自己为Docker容器添加网卡、配置IP等。   4 bridge模式  bridge模式是Docker默认的网络设置，此模式会为每一个容器分配Network Namespace、设置IP等，并将一个主机上的Docker容器连接到一个虚拟网桥上。下面着重介绍一下此模式。host模式  使用Docker run时使用–net=host指定 Docker使用的网络实际上和宿主机一样，在容器内看到的网卡ip是宿主机上的ip。  docker常用命令  1. docker version  查看docker的版本号，包括客户端、服务端、依赖的Go等  2. docker info  查看系统(docker)层面信息，包括管理的images, containers数等  3. docker search <image>在docker index中搜索image  4. docker pull <image>从docker registry server 中下拉image  5. docker push <image|repository>推送一个image或repository到registry  6. docker push <image|repository>:TAG  同上，指定tag  7. docker inspect <image|container>查看image或container的底层信息  8. docker images  查看本机images  9. docker images –a  列出所有的images  10. dockerps默认显示正在运行中的container  六、docker怎么实现容器间的独立 |

12、实现高可用工具有哪些

|  |  |
| --- | --- |
| 1  2 | hearbeat  keepalived |

13、mysql数据备份工具

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | mysqldump工具  Mysqldump是mysql自带的备份工具，目录在bin目录下面：/usr/local/mysql/bin/mysqldump，支持基于innodb的热备份。但是由于是逻辑备份，所以速度不是很快，适合备份数据比较小的场景。Mysqldump完全备份+二进制日志可以实现基于时间点的恢复。  基于LVM快照备份  在物理备份中，有基于文件系统的物理备份（LVM的快照），也可以直接用tar之类的命令对整个数据库目录进行打包备份，但是这些只能进行泠备份，不同的存储引擎备份的也不一样，myisam自动备份到表级别，而innodb不开启独立表空间的话只能备份整个数据库。  tar包备份  percona提供的xtrabackup工具  支持innodb的物理热备份，支持完全备份，增量备份，而且速度非常快，支持innodb存储引起的数据在不同数据库之间迁移，支持复制模式下的从机备份恢复备份恢复  ，为了让xtrabackup支持更多的功能扩展，可以设立独立表空间，打开 innodb\_file\_per\_table功能，启用之后可以支持单独的表备份。 |

14、awk和sed的区别

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | awk是一种程序语言，对文档资料的处理具有很强的功能。awk擅长从格式化报文或从一个大的文本文件中抽取数据。  awk的命令格式为：  awk [-F filed-separator] “commands” input-file(s)  awk将一行文字按分隔符（filed-separator）分为多个域，依次记为$ 1，$ 2 . . . $ n。$0代表所有域值。因此awk更适合于以域为单位来处理文件。加之ARGIND等内置变量，使awk能处理多个文件。典型的应用为查找一个文件中的某个字段是否在另一个文件中出现  sed是一个精简的、非交互式的编辑器。它能执行与编辑vi和emacs相同的编辑任务，但sed编辑器不提供交互使用方式，只能在命令行下输入编辑命令。  sed的命令格式为：  sed [options] 'command' file(s)  sed [options] -f scrīptfile file(s)    作为编辑器，当然少不了插入（a/、i/）、删除（d）、查找替换(s)等命令。    如果文件是格式化的，即由分隔符分为多个域的，优先使用awk  awk适合按列（域）操作，sed适合按行操作  awk适合对文件的抽取整理，sed适合对文件的编辑。 |

15、介绍CDN的作用

|  |  |
| --- | --- |
| 1  2  3 | CDN加速简单的来说，就是把原服务器上数据复制到其他服务器上，用户访问时，那台服务器近访问到的就是那台服务器上的数据。CDN加速优点是成本低，速度快。可以用CDN best的CDN进行加速，免费，可部署私有，公有CDN系统。可以实现宕机检测，自动切换ip，分线路，分组解析。也就是  CDN加速  的主要作用就是保证网站的正常访问，及加快网站访问速度和响应速度，防止网站因黑客攻击，DNS解析劫持故障等导致的网站服务器的宕机状况的出现。 |

16、 keepalive的工作原理和如何做到健康检查

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | keepalived是以VRRP协议为实现基础的，VRRP全称Virtual Router Redundancy Protocol，即虚拟路由冗余协议。  虚拟路由冗余协议，可以认为是实现路由器高可用的协议，即将N台提供相同功能的路由器组成一个路由器组，这个组里面有一个master和多个backup，master上面有一个对外提供服务的vip（该路由器所在局域网内其他机器的默认路由为该vip），master会发组播，当backup收不到vrrp包时就认为master宕掉了，这时就需要根据VRRP的优先级来选举一个backup当master。这样的话就可以保证路由器的高可用了。  keepalived主要有三个模块，分别是core、check和vrrp。core模块为keepalived的核心，负责主进程的启动、维护以及全局配置文件的加载和解析。check负责健康检查，包括常见的各种检查方式。vrrp模块是来实现VRRP协议的。  Keepalived健康检查方式配置  HTTP\_GET|SSL\_GET  HTTP\_GET | SSL\_GET  {  url {  path /# HTTP/SSL 检查的url可以是多个  digest <STRING> # HTTP/SSL 检查后的摘要信息用工具genhash生成  status\_code 200# HTTP/SSL 检查返回的状态码  }  connect\_port 80 # 连接端口  bindto<IPADD>  connect\_timeout 3 # 连接超时时间  nb\_get\_retry 3 # 重连次数  delay\_before\_retry 2 #连接间隔时间  } |

17、 squid的默认缓存大小是多少？

|  |  |
| --- | --- |
| 1 | 1024MB |

18、查看Linux系统的所有配置的命令（除了cpuinfo和meinfo）

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | 查看主板的序列号: dmidecode | grep -i ’serial number’  用硬件检测程序kuduz探测新硬件：service kudzu start ( or restart)  查看CPU信息：cat /proc/cpuinfo [dmesg | grep -i 'cpu'][dmidecode -t processor]  查看内存信息：cat /proc/meminfo [free -m][vmstat]  查看板卡信息：cat /proc/pci  查看显卡/声卡信息：lspci |grep -i ‘VGA’[dmesg | grep -i 'VGA']  查看网卡信息：dmesg | grep -i ‘eth’[cat /etc/sysconfig/hwconf | grep -i eth][lspci | grep -i 'eth']  www.2cto.com  <!--more-->  查看PCI信息：lspci (相比cat /proc/pci更直观）  查看USB设备：cat /proc/bus/usb/devices  查看键盘和鼠标:cat /proc/bus/input/devices  查看系统硬盘信息和使用情况：fdisk& disk – l &df  查看各设备的中断请求(IRQ):cat /proc/interrupts  查看系统体系结构：uname -a |

19、用shell脚本或命令将/opt目录下的大于15KB的文件都移到/tmp目录下

find /opt -size +15k -exec mv {} /tmp/ \;

20、写过哪些python脚本？

这个程序的作用是跳过最新的7\*24个备份，删除其他的时间更长的

          # -\*- coding: cp936 -\*-  
  
import os  #os是python自带的一个扩展库，用来进行跟系统相关的操作  
  
skipNum = 24 \* 7    #保留文件夹的数目（小时 \* 天）  
  
print '\n\*\*\*\*\*\*\*\*\*\*\*\*\*清除时间超过 ' + str(skipNum /24) + ' 天的自动备份文件\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n'  
  
backupDir = os.path.abspath(\_\_file\_\_) + '\\..\\..' + '\\servedatabase\_autobackup'  
for dir in os.listdir(backupDir):   #os.listdir(xxx)函数返回的是一个文件列表  
    if dir[-4:] == '.map':  
        mapDir = backupDir + '\\' + dir  
        print '正在清除备份目录：' + os.path.abspath(mapDir)  
        subDirs = os.listdir(mapDir) #获取文件列表  
        subDirs.reverse()            #逆序  
        i = 0  
        for subDir in subDirs:  
            i+=1  
            if i>skipNum:  
                print '正在清除备份: ' + subDir   
                os.system('rmdir ' + mapDir + '\\' + subDir + ' /s/q') #相当于执行控制台命令

防伪码：不要仰望别人，自己亦是风景。

1、写一个脚本查找最后创建时间是三天前，后缀是\*.log的文件并删除

|  |  |
| --- | --- |
| 1 | find .-ctime +3 -name '\*.log' | rm -rf |

2、统计ip访问情况，要求分析nginx访问日志，找出访问页面数量在前十位的ip

|  |  |
| --- | --- |
| 1 | cataccess.log | awk '{print $1}' | uniq -c | sort -rn | head -10 |

3、使用tcpdump监听主机为192.168.1.1，tcp端口为80的数据，同时将输出结果保存输出

到tcpdump.log。

|  |  |
| --- | --- |
| 1 | tcpdump 'host 192.168.1.1 and port 80' > tcpdump.log |

4、利用Python打印前一天的本地时间，格式为'2016-03-29 13:58:34'

|  |  |
| --- | --- |
| 1 | time.strftime('%y-%m-%d%H:%M%:%S' time.locatime(time.time() -86400)) |

5、用Python将‘123456’反转为‘654321’

|  |  |
| --- | --- |
| 1 | ‘123456’[::-1] |

6、利用Python执行shell命令并取得返回结果

|  |  |
| --- | --- |
| 1  2  3  4  5 | >>> import subprocess  >>> a=subprocess.popen('ls',shell=True,stdout=subprocoss.PIPE,  stderr=subprocess.PIPE)  >>>stdout,sterr = a.communicate()  >>>print stdout |

8、请用Python继承process，并写一个并行执行的类，并写出使用过程。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | import multiprocessing  import time  class ClockProcess(multiprocessing.Process):      def \_\_init\_\_(self, interval):          multiprocessing.Process.\_\_init\_\_(self)          self.interval = interval      def run(self):          n = 5          while n > 0:              print("the time is {0}".format(time.ctime()))              time.sleep(self.interval)              n -= 1 |

9、有文件file1

a、请用shell查询file1 里面空行的所在行号

|  |  |
| --- | --- |
| 1 | #awk '$0 ~ /^$/ {print NR}' file1 |

或

|  |  |
| --- | --- |
| 1 | grep -n "^$" file1 |

b、编写ShellScript查询file1 以abc结尾的行

|  |  |
| --- | --- |
| 1 | grep "abc$" file1 |

或

|  |  |
| --- | --- |
| 1 | sed -n '/abc$/ p' file1 |

c、打印出file1 文件第1 到第3 行

|  |  |
| --- | --- |
| 1 | sed  -n '1,3p' file1 |

10、如何将本地80 端口的请求转发到8080 端口，当前主机IP 为192.168.2.1

|  |  |
| --- | --- |
| 1 | Iptables –t nat -A PREROUTING -d 192.168.2.1 -p tcp  –dport 80 -j REDIRECT --to 8080 |

11、在11 月份内，每天的早上6 点到12 点中，每隔2 小时执行一次/usr/bin/httpd.sh 怎么

实现

|  |  |
| --- | --- |
| 1 | 0 6-12/2 \* 11 \* /usr/bin/httpd.sh |

12、编写个shell 脚本将/usr/local/test 目录下大于100K 的文件转移到/tmp目录下

|  |  |
| --- | --- |
| 1 | find /usr/local/test -type f -size +100k -exec mv {} /tmp/ |

或

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | #!/bin/bash  filelist=$(ls  /usr/local/test -l | grep "^-" | awk '{print $9}')  filepath="/usr/local/test"  for file in $filelist  do      filesize=$(du -s $filepath$file |awk '{print $1}')      if [ $filesize -gt 100]      then      mv $filepath$file /tmp/  fi  done |

13、简述raid0 raid1 raid5 三种工作模式的工作原理及特点。

|  |  |
| --- | --- |
| 1  2 | RAID 0：带区卷，连续以位或字节为单位分割数据，并行读/写于多个磁盘上，因此具有很高的数据传输率，但它没有数据冗余，  RAID 0 只是单纯地提高性能，并没有为数据的可靠性提供保证，而且其中的一个磁盘失效将影响到所有数据。因此，RAID 0 不能应用于数据安全性要求高的场合。 |

|  |  |
| --- | --- |
| 1  2 | RAID 1：镜像卷，它是通过磁盘数据镜像实现数据冗余，在成对的独立磁盘上产生互为备份的数据，不能提升写数据效率。当原始数据繁忙时，可直接从镜像拷贝中读取数据，因此RAID  1 可以提高读取性能。RAID 1 是磁盘阵列中单位成本最高的，镜像卷可用容量为总容量的1/2，但提供了很高的数据安全性和可用性。当一个磁盘失效时，系统可以自动切换到镜像磁盘上读写，而不需要重组失效的数据。 |

|  |  |
| --- | --- |
| 1 | RAID5：至少由3块硬盘组成，分布式奇偶校验的独立磁盘结构，它的奇偶校验码存在于所有磁盘上，任何一个硬盘损坏，都可以根据其它硬盘上的校验位来重建损坏的数据（最多允许1块硬盘损坏）。所以raid5可以实现数据冗余，确保数据的安全性，同时raid5也可以提升数据的读写性能。 |

14、oracle 数据库备份方式

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | 从物理与逻辑的角度分类：  物理备份：对数据库操作系统的物理文件（如数据文件、控制文件、日志文件等）的备份，物理备份又分为冷备份和热备份，前者是在关闭数据库的时候进行的，后者对正以归档日志方式进行的数据库备份，可以使用RMAN进行数据库的物理备份  逻辑备份：对数据库逻辑组件（如表、表空间、存储过程等数据库对象）的备份逻辑备份的手段很多，如exp、数据泵expdp、数据库闪回技术等  从数据库的备份策略角度分类：  完全备份：每次对数据库进行完整备份  增量备份：只有那些在上次完全备份或者增量备份后被修改的文件才会被备份。  差异备份：备份那些自从上次完全备份之后被修改过的文件。 |

15、如何查看占用端口8080 的进程

|  |  |
| --- | --- |
| 1  2  3 | netstat -anpt | grep 8080  或   lsof -i :8080 |

八、请写出apache2.X 版本的两种工作模式，以及各自工作原理。如何查看apache 当前所支持的模块，并且查看是工作在哪种模式下？

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | prefork(多进程，每个子进程产生一个线程)和worker（多进程多线程，每个进程生成多个线程)  　　其主要工作方式是：当Apache服务器启动后，mpm\_prefork模块会预先创建多个子进程(默认为5个)，每个子进程只有一个线程，当接收到客户端的请求后，mpm\_prefork模块再将请求转交给子进程处理，并且每个子进程同时只能用于处理单个请求。如果当前的请求数将超过预先创建的子进程数时，mpm\_prefork模块就会创建新的子进程来处理额外的请求。Apache总是试图保持一些备用的或者是空闲的子进程用于迎接即将到来的请求。这样客户端的请求就不需要在接收后等候子进程的产生。  由于在mpm\_prefork模块中，每个请求对应一个子进程，因此其占用的系统资源相对其他两种模块而言较多。不过mpm\_prefork模块的优点在于它的每个子进程都会独立处理对应的单个请求，这样，如果其中一个请求出现问题就不会影响到其他请求。Prefork在效率上要比Worker要高，但是内存使用大得多不擅长处理高并发的场景。  　　Worker模式(多线程多进程)：  和prefork模式相比，worker使用了多进程和多线程的混合模式，worker模式也同样会先预派生一些子进程，然后每个子进程创建一些线程，同时包括一个监听线程，每个请求过来会被分配到一个线程来服务。线程比起进程会更轻量，因为线程是通过共享父进程的内存空间，因此，内存的占用会减少一些，在高并发的场景下会比prefork有更多可用的线程，表现会更优秀一些；另外，如果一个线程出现了问题也会导致同一进程下的线程出现问题，如果是多个线程出现问题，也只是影响Apache的一部分，而不是全部。  总的来说，prefork方式速度要稍高于worker，然而它需要的cpu和memory资源也稍多于woker。  可以通过命令httpd  -M  或 httpd -l 可以查看apache 当前的模块  可以通过命令httpd  -V查看是工作在哪种模式下 |

16、你使用过监控软件吗？说说其特点

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | 使用nagios对服务器进行监控，其特点侧重于对检测项的状态监控，主要通过nrpe实现对远程主机的监控，但也可以通过snmp对设备（如路由器、交换机）进行监控，可实时实现手机短信、电子邮件、MSN、飞信报警。  使用cacti 对服务器进行监控,其特点侧重性能和流量监控并通过图表显示，主要通过snmp协议收集监测项数据，可实时实现手机短信、电子邮件、  使用zabbix对服务器进行监控，zabbix是完全开源的工具，整合了cacti和nagios等特性。  zabbix可以对主机的性能监控、网络设备性能监控、数据库、FTP 等通用协议监控、多种告警方式、详细的报表图表绘制  支持自动发现网络设备和服务器（可以通过配置自动发现服务器规则来实现）  支持分布式，能集中展示、管理分布式的监控点，扩展性强  可以自己开发完善各类监控（根据相关接口编写程序实现），编写插件容易，可以自定义监控项，报警级别的设置。  数据收集，支持 snmp(包括 trapping and polling )，IPMI，JMX，SSH，TELNET； |

17、你对现在运维工程师的理解和以及对其工作的认识

|  |  |
| --- | --- |
| 1 | 运维工程师在公司当中责任重大，需要保证时刻为公司及客户提供最高、最快、最稳定、最安全的服务。运维工程师的一个小小的失误，很有可能会对公司及客户造成重大损失，因此运维工程师的工作需要严谨及富有创新精神。 |

18、第一部分 基础题

a. 仅列出 /test目录下的所有目录，请写出完整命令

|  |  |
| --- | --- |
| 1 | ls /test -l | grep "^d" |

b. umask 022 ，请描述该命令的含义

|  |  |
| --- | --- |
| 1 | umask设置文件或目录的缺省权限，umask 022表示目录的缺省权限为755，文件的缺省权限为644 |

c. 在排除 /test/a目录的条件下，归档并gzip压缩整个/test目录，请写出完整的操作命令t

|  |  |
| --- | --- |
| 1 | tar zcvf backup.tar.gz --exclude /test/a /test |

d. 查询并列出 test进程所打开的当前所有文件，请写出完整的操作命令

|  |  |
| --- | --- |
| 1 | lsof -c test |

e. 查询并列出test.rpm包里所包含的所有文件，请写出完整的操作命令

|  |  |
| --- | --- |
| 1  2 | rpm -qpl    test.rpm |

f. 将 /test/a目录建立软链接到 /test/b目录，请写出完整的操作命令

|  |  |
| --- | --- |
| 1 | ln -s /test/a test/b |

g. 设置当前用户环境中 test变量为 123并使之立即生效，请写出完整操作命令

|  |  |
| --- | --- |
| 1 | export test=123 |

h. 设置系统当前运行级别中 test服务状态为启动时自动加载，请写出完整操作命令

假设运行级别为3级别

|  |  |
| --- | --- |
| 1 | chkconfig  --level  3  test  on |

k. 列出当前系统中所有的网络连接（包含进程名），请写出完整操作命令

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | netstat -antuple  （-a 显示所有socket，包括正在监听的。  　　-n 以网络IP地址代替名称，显示出网络连接情形。  　　-t 显示TCP协议的连接情况  　　-u 显示UDP协议的连接情况。  　　-p 显示指定协议信息。      -l或--listening 显示监控中的服务器的Socket。  　　-e 显示以太网统计。此选项可以与 -s 选项结合使用。  ） |

j. 实时抓取并显示当前系统中tcp 80端口的网络数据信息，请写出完整操作命令

|  |  |
| --- | --- |
| 1 | tcpdump -nn tcp port 80 |

19、第二部分综合应用  
a. 请写出5个你常用的系统或网络维护工具的名称

|  |  |
| --- | --- |
| 1  2  3  4  5 | rhel  centos  ubuntu  iotop用于检查 I/O 的使用情况  htop实质上是 top 的一个增强版本。它更容易对进程排序。  ping  traceroute |

b.在nslookup使用中，请问如何查看test.com域中的MX记录

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | #nslookup  >set type=mx  >test.comc. ftp ，ssh，smtp，pop3，https，请写出这些服务的默认端口ftp:21  ssh:22  smtp:25  pop3:110  https:443.25G） |

d. 请问如何用iptables工具阻断来自 192.168.0.1的所有网络连接

|  |  |
| --- | --- |
| 1 | iptable -I INPUT -s 192.168.0.1 -j DROP |

e. 请简要描述Linux系统下源代码编译方式安装软件的大致步骤

|  |  |
| --- | --- |
| 1  2  3  4  5 | tar释放源码包  cd 切换到解压目录  ./configure 配置  make 编译  make  install  安装 |

f. 请写出Tomcat程序的主配置文件名

|  |  |
| --- | --- |
| 1 | server.xml |

g. Mysql客户端工具中，请问如何查询当前所有的连接进程信息

|  |  |
| --- | --- |
| 1  2 | mysql -u user -p password  -e "show processlist" |

20. 服务器开不了机怎么解决一步步的排查

|  |  |
| --- | --- |
| 1  2  3 | 硬件有无报警灯提示  主面板液晶面板有没有提示什么报错信息，例如raid错误的提示信息  先排除硬件还是软件问题 |

十四. Linux开机模式

|  |  |
| --- | --- |
| 1  2 | 0 - 表示关闭系统（千万不要把默认设置成0哦！）1 - 单用户模式2 - 多用户模式，没有NFS服务3 - 多用户文本模式5 - 图形模式6 - 系统重新启动  过修改/etc/inittab，使系统在启动时进入不同模式。 |

21. Linux系统中病毒怎么解决

|  |  |
| --- | --- |
| 1  2  3  4  5 | 找到病毒文件然后删除；中毒之后一般机器cpu、内存使用率会比较高，机器向外发包等异常情况，排查方法：  linux服务器流量剧增,用iftop查看有连接外网的情况。netstat连接的外网ip和端口。#top命令找到cpu使用率高的进程，一般病毒文件命名都比较乱#可以用ps aux 查看是否有不明进程，找出病毒文件的位置#rm  -f 命令删除病毒文件#检查计划任务、开机启动项和病毒文件目录有无其他可疑文件等  chkconfig    --list     |  grep  3:on  服务器启动级别是3的，检查一下了开机启动项，没有特别明显的服务。然后检查了一下开机启动的一个文件，more   /etc/rc.local |

22. 发现一个病毒文件你删了他又自动创建怎么解决

|  |  |
| --- | --- |
| 1  2 | ps axu一个个排查，方法是查看可疑的用户和系统相似而又不是的进程找出进程可疑。  杀掉所有与病毒相关的进程，然后删掉病毒这个可执行文件，最后删除病毒创建的文件 |

十七. 日志文件很大，怎么把他们切分

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | 针对这些日志按每或每周进行分割，例如只保留一周的数据，用logrotate来实现日志的轮替。或者编写日志文件大小监控脚本，定期检查该日志文件的大小，接近设定大小时，进行轮换。  如果日志文件存在并且很大，可以用Linux下的split进行文件分割：  模式一：指定分割后文件行数  Split:按指定的行数截断文件格式: split [-n] file [name]参数说明：-n: 指定截断的每一文件的长度，不指定缺省为1000行file: 要截断的文件name： 截断后产生的文件的文件名的开头字母，不指定，缺省为x，即截断后产生的文件的文件名为xaa,xab....直到xzz    模式二：指定分割后文件大小  命令：split -b 10m server.log server\_part\_  其中server.log是要分割的文件，server\_part\_是分割文件的前缀。  对二进制文件我们同样也可以按文件大小来分隔 |

23. tcp/ip七层模型

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | 应用层 (Application)：  网络服务与最终用户的一个接口。  协议有：HTTP FTP TFTP SMTP SNMP DNS TELNET HTTPS POP3 DHCP  表示层（Presentation Layer）：  数据的表示、安全、压缩。（在五层模型里面已经合并到了应用层）  格式有，JPEG、ASCll、DECOIC、加密格式等  会话层（Session Layer）：  建立、管理、终止会话。（在五层模型里面已经合并到了应用层）  对应主机进程，指本地主机与远程主机正在进行的会话  传输层 (Transport)：  定义传输数据的协议端口号，以及流控和差错校验。  协议有：TCP UDP，数据包一旦离开网卡即进入网络传输层  网络层 (Network)：  进行逻辑地址寻址，实现不同网络之间的路径选择。  协议有：ICMP IGMP IP（IPV4 IPV6） ARP RARP  数据链路层 (Link)：  建立逻辑连接、进行硬件地址寻址、差错校验等功能。（由底层网络定义协议）  将比特组合成字节进而组合成帧，用MAC地址访问介质，错误发现但不能纠正。  物理层（Physical Layer）： |

24. 你常用的Nginx模块，用来做什么

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | rewrite模块，实现重写功能  access模块：来源控制  ssl模块：安全加密  ngx\_http\_gzip\_module：网络传输压缩模块  ngx\_http\_proxy\_module 模块实现代理  ngx\_http\_upstream\_module模块实现定义后端服务器列表  ngx\_cache\_purge实现缓存清除功能 |

25、请列出你了解的web服务器负载架构。

|  |  |
| --- | --- |
| 1  2  3  4 | Nginx  haproxy  keepalived  lvs |

1、docker命令

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | 容器生命周期管理 — docker [run|start|stop|restart|kill|rm|pause|unpause]  容器操作运维 — docker [ps|inspect|top|attach|events|logs|wait|export|port]  容器rootfs命令 — docker [commit|cp|diff]  镜像仓库 — docker [login|pull|push|search]  本地镜像管理 — docker [images|rmi|tag|build|history|save|import]  其他命令 — docker [info|version] |

2、docker怎样实现容器件的独立

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | 1)pid namespace  不同用户的进程就是通过pid namespace 隔离开的，且不同  namespace  中可以有相同pid。所有的LXC进程在docker中的父进程为docker进程，每个lxc进程具有不同的 namespace 。  2) net namespace  有了pid namespace,  每个 namespace 中的pid能够相互隔离，但是网络端口还是共享 host 的端口。网络隔离是通过 net namespace 实现的，每个 net namespace 有独立的  network devices, IP addresses, IP routing tables, /proc/net  目录。这样每个 container 的网络就能隔离开来。docker默认采用veth的方式将 container 中的虚拟网卡同 host 上的一个docker bridge: docker0 连接在一起。  3) ipc namespace  container 中进程交互还是采用linux常见的进程间交互方法 (interprocess communication - IPC),包括常见的信号量、消息队列和共享内存。container  的进程间交互实际上还是host 上具有相同pid namespace 中的进程间交互。  4) mnt namespace  类似chroot，将一个进程放到一个特定的目录执行。mnt namespace 允许不同 namespace 的进程看到的文件结构不同，这样每个  namespace  中的进程所看到的文件目录就被隔离开了。在container里头，看到的文件系统，就是一个完整的linux系统，有/etc、/lib 等，通过chroot实现。  5) uts namespace  UTS("UNIX Time-sharing System") namespace 允许每个 container 拥有独立的 hostname 和 domain name,  使其在网络上可以被视作一个独立的节点而非 Host 上的一个进程。  6) user namespace  每个 container 可以有不同的  user  和  group id,  也就是说可以在 container 内部用 container 内部的用户执行程序而非 Host 上的用户。  有了以上 6 种 namespace 从进程、网络、IPC、文件系统、UTS和用户角度的隔离，一个 container 就可以对外展现出一个独立计算机的能力，并且不同 container 从 OS 层面实现了隔离。然而不同 namespace 之间资源还是相互竞争的，仍然需要类似ulimit来管理每个 container 所能使用的资源 - -cgroup。  cgroups（Control groups）实现了对资源的配额和度量。 |

3、Linux如何挂载windows下的共享目录

|  |  |
| --- | --- |
| 1  2 | mount.cifs //IP地址/server /mnt/server -o user=administrator,password=123456  linux 下的server需要自己手动建一个  后面的user与pass 是windows主机的账号和密码 注意空格 和逗号 |

4、查看http的并发请求数与其TCP连接状态

|  |  |
| --- | --- |
| 1 | netstat -n | awk '/^tcp/ {++b[$NF]}' END {for(a in b) print a,b[a]}' |
| 1  2  3  4  5 | 还有ulimit -n 查看linux系统打开最大的文件描述符，这里默认1024，不修改这里web服务器修改再大也没用。若要用就修改很几个办法，这里说其中一个：  修改/etc/security/limits.conf  \* soft nofile 10240  \* hard nofile 10240  重启后生效 |

5、用tcpdump嗅探80端口的访问看看谁最高。

|  |  |
| --- | --- |
| 1 | tcpdump -i eth0 -tnn dst port 80 -c 1000 | awk -F"." '{print $1"."$2"."$3"."$4"."}' | sort |uniq -c | sort -nr | head-5&nbsp; |

6、查看/var/log目录下的文件数

|  |  |
| --- | --- |
| 1 | ls /var/log/ -1R | grep "-" | wc -l |

7、查看当前系统每个ip的连接数

|  |  |
| --- | --- |
| 1 | netstat -n | awk '/^tcp/ {print $5}' | awk -F: '{print $1}' | sort | uniq -c | sort -rn |

8、shell下32位随机密码生成。

|  |  |
| --- | --- |
| 1 | cat /dev/urandom | head -1 | md5sum | head -c 32 >> /pass |

9、统计出apache的access.log中访问量最多的5个ip。

|  |  |
| --- | --- |
| 1 | cat access.log | awk '{print $1}' | sort | uniq -c | sort -n -r | head -5 |

10**.如何查看二进制文件的内容**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | 我们一般通过hexdump命令 来查看二进制文件的内容。  hexdump -C XXX(文件名)  -C是参数 不同的参数有不同的意义  -C  是比较规范的 十六进制和ASCII码显示  -c  是单字节字符显示  -b  单字节八进制显示  -o  是双字节八进制显示  -d  是双字节十进制显示  -x  是双字节十六进制显示  ... ... |

**11.ps aux 中的VSZ代表什么意思，RSS代表什么意思**

|  |  |
| --- | --- |
| 1  2 | VSZ:虚拟内存集,进程占用的虚拟内存空间  RSS:物理内存集,进程战用实际物理内存空间 |

**10.检测并修复/dev/hda5**

|  |  |
| --- | --- |
| 1 | fsck用来检查和维护不一致的文件系统。若系统掉电或磁盘发生问题，可利用fsck命令对文件系统进行检查 |

**12.Linux系统的开机启动顺序**

|  |  |
| --- | --- |
| 1  2  3 | 加载BIOS–>读取MBR–>Boot Loader–>加载内核–>用户层init一句inittab文件来设定系统运行的等级(一般3或者  5，3是多用户命令行，5是界面)–>init进程执行rc.syninit–>启动内核模块–>执行不同级别运行的脚本程序–>执行/etc/rc.d/rc.local(本地运行服务)–>  执行/bin/login,就可以登录了。 |

**13.符号链接与硬链接的区别**

|  |  |
| --- | --- |
| 1  2  3  4 | 我们可以把符号链接，也就是软连接 当做是 windows系统里的 快捷方式。  硬链接 就好像是 又复制了一份.  ln 3.txt 4.txt   这是硬链接，相当于复制，不可以跨分区，但修改3,4会跟着变，若删除3,4不受任何影响。  ln -s 3.txt 4.txt  这是软连接，相当于快捷方式。修改4,3也会跟着变，若删除3,4就坏掉了。不可以用了。 |

**14.保存当前磁盘分区的分区表**

|  |  |
| --- | --- |
| 1  2 | dd 命令是以个强大的命令，在复制的同时进行转换  dd if=/dev/sda of=./mbr.txt bs=1 count=512 |

**15.如何在文本里面进行复制、粘贴，删除行，删除全部，按行查找和按字母查找。**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | 以下操作全部在命令行状态操作，不要在编辑状态操作。  在文本里 移动到想要复制的行  按yy  想复制到哪就移动到哪，然后按P  就黏贴了  删除行  移动到改行 按dd  删除全部  dG  这里注意G一定要大写  按行查找  :90 这样就是找到第90行  按字母查找 /path  这样就是 找到path这个单词所在的位置，文本里可能存在多个,多次查找会显示在不同的位置。 |

**16.手动安装grub**

|  |  |
| --- | --- |
| 1 | grub-install /dev/sda |

**17.修改内核参数**

|  |  |
| --- | --- |
| 1  2 | vi /etc/sysctl.conf  这里修改参数  sysctl -p  刷新后可用 |

**18.在1-39内取随机数**

|  |  |
| --- | --- |
| 1  2  3 | expr $[RANDOM%39] +1  RANDOM随机数  %39取余数范围0-38 |

**19.限制apache每秒新建连接数为1，峰值为3**

|  |  |
| --- | --- |
| 1  2 | 每秒新建连接数 一般都是由防火墙来做，apache本身好像无法设置每秒新建连接数，只能设置最大连接：  iptables -A INPUT -d 172.16.100.1 -p tcp --dport 80 -m limit --limit  1/second -j ACCEPT |

**20.FTP的主动模式和被动模式**

|  |  |
| --- | --- |
| 1  2  3  4 | FTP协议有两种工作方式：PORT方式和PASV方式，中文意思为主动式和被动式。  PORT（主动）方式的连接过程是：客户端向服务器的FTP端口（默认是21）发送连接请 求，服务器接受连接，建立一条命令链路。当需要传送数据时，客户端在命令链路上用PORT 命令告诉服务器：“我打开了XX端口，你过来连接我”。于是服务器从20端口向客户端的 XX端口发送连接请求，建立一条数据链路来传送数据。  PASV（被动）方式的连接过程是：客户端向服务器的FTP端口（默认是21）发送连接请 求，服务器接受连接，建立一条命令链路。当需要传送数据时，服务器在命令链路上用PASV 命令告诉客户端：“我打开了XX端口，你过来连接我”。于是客户端向服务器的XX端口 发送连接请求，建立一条数据链路来传送数据。  从上面可以看出，两种方式的命令链路连接方法是一样的，而数据链路的建立方法就完 全不同。 |

**21.显示/etc/inittab中以#开头，且后面跟了一个或者多个空白字符，而后又跟了任意非空白字符的行。**

|  |  |
| --- | --- |
| 1 | grep "^#\{1,\}[^]" /etc/inittab |

22.显示/etc/inittab中包含了:一个数字:(即两个冒号中间一个数字)的行

|  |  |
| --- | --- |
| 1 | grep "\:[0-9]\{1\}:" /etc/inittab |

**23.怎么把脚本添加到系统服务里，即用service来调用**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | 在脚本里加入  #!/bin/bash  # chkconfig: 345 85 15  # description: httpd  然后保存  chkconfig httpd –add  创建系统服务  现在就可以使用service 来 start or restart |

**24.写一个脚本，实现批量添加20个用户，用户名为user01-20，密码为user后面跟5个随机字符**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | #!/bin/bash  #description: useradd  for i in `seq -f"%02g" 1 20`;do  useradd user$i  echo "user$i-`echo $RANDOM|md5sum|cut -c 1-5`"|passwd –stdinuser$i >/dev/null 2>&1  done |

**25.写一个脚本，实现判断192.168.1.0/24网络里，当前在线的IP有哪些，能ping通则认为在线**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | #!/bin/bash  for ip in `seq 1 255`  do  ping -c 1 192.168.1.$ip > /dev/null 2>&1  if [ $? -eq 0 ]; then  echo 192.168.1.$ip UP  else  echo 192.168.1.$ip DOWN  fi  }&  done  wait |

**26.写一个脚本，判断一个指定的脚本是否是语法错误；如果有错误，则提醒用户键入Q或者q无视错误并退出其它任何键可以通过vim打开这个指定的脚本**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | [root@localhost  tmp]# cat checksh.sh   #!/bin/bash   read -p "please input check script-> " file   if [ -f $file ]; then      sh -n $file > /dev/null 2>&1      if [ $? -ne 0 ]; then          read -p "You input $file syntax error,[Type q to exit or Type vim to  edit]" answer          case $answer in          q | Q)             exit 0             ;;          vim )             vim $file             ;;          \*）           exit 0           ;;          esac     fi   else      echo "$file not exist"      exit 1   fi |

**27、写一个脚本：(26包括3个小题)**

1、创建一个函数，能接受两个参数：

1)第一个参数为URL，即可下载的文件；第二个参数为目录，即下载后保存的位置；

2)如果用户给的目录不存在，则提示用户是否创建；如果创建就继续执行，否则，函数返回一个51的错误值给调用脚本；

3)如果给的目录存在，则下载文件；下载命令执行结束后测试文件下载成功与否；如果成功，则返回0给调用脚本，否则，返回52给调用脚本；

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | [root@localhost tmp]# cat downfile.sh  #!/bin/bash  url=$1  dir=$2  download()    {      cd $dir >> /dev/null 2>&1      if [ $? -ne 0 ];then          read -p "$dir No such file or directory,create?(y/n)" answer          if [ "$answer" == "y" ];then              mkdir -p $dir              cd $dir              wget $url 1> /dev/null 2>&1          else              return "51"          fi      fi      if [ $? -ne 0 ]; then          return "52"      fi  }  download $url $dir  echo $? |

**28、写一个脚本：（27包括2个小题）**

1、创建一个函数，可以接受一个磁盘设备路径（如/dev/sdb）作为参数;在真正开始后面步骤之前提醒用户有危险，并让用户选择是否继续；而后将此磁盘设备上的所有分区清空（提示，使用命令dd if=/dev/zero of=/dev/sdb bs=512 count=1实现，注意其中的设备路径不要写错了；

如果此步骤失败，返回67给主程序；

接着在此磁盘设备上创建两个主分区，一个大小为100M，一个大小为1G；如果此步骤失败，返回68给主程序；

格式化此两分区，文件系统类型为ext3；如果此步骤失败，返回69给主程序；

如果上述过程都正常，返回0给主程序；

2、调用此函数；并通过接收函数执行的返回值来判断其执行情况，并将信息显示出来；

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | local Darray=(`ls /dev/sd[a-z]`)  for i in ${Darray};do    [[ "$i" == "$1" ]] && Sd=$i &&break  done    else    return66    fi |

#当匹配成功，进入选择，告诉用户，是否继续，输错的话进入无限循环，当用户选择Y,则清空目标分区，且跳出while循环

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | while :;do      read -p "Warning!!!This operation will clean $Sd data.Next=y,Quit=n [y|n]:" Choice      case $Choice in  y)     dd if=/dev/zero of=$Sd bs=512 count=1 &> /dev/null &&break || return 67 ;;  n)     exit 88 ;;  \*)     echo "Invalid choice,please choice again." ;;  esac  done |

#使用echo传递给fdisk进行分区，如果此命令失败，则跳转出去，错误值68，需要注意的是，有时候这个返回值很诡异，笔者之前成功与否都是返回的1，后来重启之后，就好了，如果慎重的话，可以对创建的分区，进行判断，不过就需要使用其他工具截取相关字段了，虽有些小麻烦，但无大碍

|  |  |
| --- | --- |
| 1 | echo-e "n\np\n1\n\n+100M\nn\np\n2\n\n+1024M\nw\n"|fdisk /dev/sdb&> /dev/null || || return 68 |

#格式化之前，让内核重新读取磁盘分区表，值得注意的是，有的系统版本，使用partprobe无效，譬如笔者的环境是rhel5.8，而rhel6.0以后，这个命令就很危险了，而使用partx -a /dev/sdb则效果更好…此项需慎重，如果格式化失败，则告知把失败的分区定义成变量，且跳出函数，并带出错误值69

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | `partprobe`  Part=`fdisk -l /dev/$Sd|tail -2|cut -d” ” -f1`  for M in ${Part};do     mke2fs -j $M &> /dev/null && ErrorPart=$M &&return 69  done    return 0  } |

#下面代码，调用函数，接收函数返回值，根据返回值进行判断哪里出错。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | Disk\_Mod $1  Res=$?  [ $Res-eq 0 ] && exit 0  [ $Res-eq 66 ] && echo "Error! Invalid input."  [ $Res-eq 67 ] && echo "Error! Command -> dd <- Faild."  [ $Res-eq 68 ] && echo "Error! Command -> fdisk <- Faild."  [ $Res-eq 69 ] && echo "Error! Command -> mke2fs <- Faild." |

28、如何让history命令显示具体时间？

|  |  |
| --- | --- |
| 1  2  3 | HISTTIMEFORMAT="%Y-%m-%d %H:%M:%S"  export HISTTIMEFORMAT  重新开机后会还原，可以写／etc／profile |