Linux运维工程师：面试题整理

1.linux如何挂在windows下的共享目录

Shell

|  |  |
| --- | --- |
| 1 | mount.cifs //192.168.1.124/server /mnt/server -o user=administrator,pass=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

2.查看http的并发请求数与其TCP连接状态

Shell

|  |  |
| --- | --- |
| 1 | netstat-n| awk '/^tcp/ {++b[$NF]} END {for(a in b) print a, b[a]}' |

[root@node1 ~]# netstat -n |awk '/^tcp/ {++b[$NF]} END {for (a in b) print a,b[a]}'

ESTABLISHED 1

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

Shell

|  |  |
| --- | --- |
| 1  2 | \* soft nofile 10240  \* hard nofile 10240 |

重启后生效

3.用tcpdump嗅探80端口的访问看看谁最高

Shell

|  |  |
| --- | --- |
| 1 | tcpdump-ieth0-tnndst port80-c1000| awk -F"."'{print $1"."$2"."$3"."$4}'| sort | uniq-c| sort -nr|head-5&nbsp; |

查看/var/log目录下文件数

-R -------》同时显示目录下的文件

|  |  |
| --- | --- |
| 1 | ls /var/log/ -lR| grep "^-" |wc -l |

5.查看当前系统每个IP的连接数

Shell

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

6.shell下32位随机密码生成

Shell

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

将生成的32位随机数 保存到/pass文件里了

7.统计出apache的access.log中访问量最多的5个IP

Shell

|  |  |
| --- | --- |
| 1 | Cat access\_log| awk  '{print $1}'| sort | uniq-c| sort -n-r| head -5 |

Sort 排序默认从小到大 -n按数值大小 -r以相反的顺序排列

Uniq 删除文中重复行

8.如何查看二进制文件的内容

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

等等等等

9.ps aux 中的VSZ代表什么意思，RSS代表什么意思

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

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,用法：

11.Linux系统的开机启动顺序

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。

12.符号链接与硬链接的区别

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。

13.保存当前磁盘分区的分区表

dd 命令是以个强大的命令，在复制的同时进行转换

Shell

|  |  |
| --- | --- |
| 1 | dd if=/dev/sda of=./mbr.txt bs=1 count=512 |

14.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。

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。

15.手动安装grub

Shell

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

16.修改内核参数

vi /etc/sysctl.conf  这里修改参数

sysctl -p  刷新后可用

17.在1-39内取随机数

Shell

|  |  |
| --- | --- |
|  | expr $[$RANDOM%39] + 1 |
|  | [root@node1 ~]# expr $[$RANDOM%39] + 1  21 |

RANDOM 随机数

%39 取余数 范围 0-38

18.限制apache每秒新建连接数为1，峰值为3

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：

Shell

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -AINPUT -d 192.168.9.110 -p tcp --dport 80 -mlimit --limit 1/second -j ACCEPT |

硬件防火墙设置更简单，有界面化，可以直接填写数字。。。

最大连接 apache本身可以设置

MaxClients 3  ,修改apache最大连接 前提还是要修改系统默认tcp连接数。

19.FTP的主动模式和被动模式

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。

20.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

Shell

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

21.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

Shell

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

22.怎么把脚本添加到系统服务里，即用service来调用

在脚本里加入

Shell

|  |  |
| --- | --- |
| 1  2  3 | #!/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

23.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

Shell

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | #!/bin/bash  #description: useradd  fori in `seq-f"%02g"1 20`;do  useradd user$i  echo"user$i:`echo $RANDOM|md5sum|cut -c 1-5`"|passwd–stdinuser$i>/dev/null2>&1  done |

24.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

Shell

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | #!/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 |

25.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

Shell

|  |  |
| --- | --- |
| 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/null2>&1      if[ $?-ne0 ];then          read-p"You input $file syntax error,[Type q to exit or Type vim to  edit]"answer          case$answerin          q| Q)             exit0            ;;          vim)            vim$file             ;;          \*）           exit0          ;;          esac    fi  else      echo"$file not exist"      exit1  fi |

26、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；

Shell

|  |  |
| --- | --- |
| 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 $? |

27、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；

Shell

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | localDarray=(`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  while:;do      read-p"Warning!!!This operation will clean $Sd data.Next=y,Quit=n [y|n]:"Choice      case$Choicein  y)    dd if=/dev/zeroof=$Sdbs=512count=1&>/dev/null&&break|| return 67 ;;  n)    exit88;;  \*)    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

Shell

|  |  |
| --- | --- |
| 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

Shell

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | `partprobe`  Part=`fdisk-l/dev/$Sd|tail-2|cut-d”” -f1`  forM in ${Part};do     mke2fs-j$M &>/dev/null&& ErrorPart=$M&&return69  done    return0  } |

#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。

Shell

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | Disk\_Mod$1  Res=$?  [$Res-eq0 ] && exit 0  [ $Res-eq66] && echo "Error! Invalid input."  [$Res-eq67] && echo "Error! Command -> dd <- Faild."  [ $Res-eq68] && echo "Error! Command -> fdisk <- Faild."  [$Res-eq69] && echo "Error! Command -> mke2fs <- Faild." |

1为什么要有swap分区？工作原理是什么？

 Linux内核为了提高读写效率与速度，会将文件在内存中进行缓存，这部分内存就是Cache Memory(缓存内存)。即使你的程序运行结束后，Cache Memory也不会自动释放。这就会导致你在Linux系统中程序频繁读写文件后，你会发现可用物理内存变少。当系统的物理内存不够用的时候，就需要将物理内存中的一部分空间释放出来，以供当前运行的程序使用。那些被释放的空间可能来自一些很长时间没有什么操作的程序，这些被释放的空间被临时保存到Swap空间中，等到那些程序要运行时，再从Swap分区中恢复保存的数据到内存中。这样，系统总是在物理内存不够时，才进行Swap交换。

2.多线程和多进程的区别，喜欢用哪个？为什么？

（思路：先对比线程和进程的优缺点，再结合具体场景介绍，不同场景下喜欢用不同方法。）

进程是资源分配的最小单位，线程是CPU调度的最小单位。

我们按照多个不同的维度，来看看多线程和多进程的对比（注：因为是感性的比较，因此都是相对的，不是说一个好得不得了，另外一个差的无法忍受）。

|  |  |  |  |
| --- | --- | --- | --- |
| 对比维度 | 多进程 | 多线程 | 总结 |
| 数据共享、同步 | 数据共享复杂，需要用IPC；数据是分开的，同步简单 | 因为共享进程数据，数据共享简单，但也是因为这个原因导致同步复杂 | 各有优势 |
| 内存、CPU | 占用内存多，切换复杂，CPU利用率低 | 占用内存少，切换简单，CPU利用率高 | 线程占优 |
| 创建销毁、切换 | 创建销毁、切换复杂，速度慢 | 创建销毁、切换简单，速度很快 | 线程占优 |
| 编程、调试 | 编程简单，调试简单 | 编程复杂，调试复杂 | 进程占优 |
| 可靠性 | 进程间不会互相影响 | 一个线程挂掉将导致整个进程挂掉 | 进程占优 |
| 分布式 | 适应于多核、多机分布式；如果一台机器不够，扩展到多台机器比较简单 | 适应于多核分布式 | 进程占优 |

1）需要频繁创建销毁的优先用线程

原因请看上面的对比。

这种原则最常见的应用就是Web服务器了，来一个连接建立一个线程，断了就销毁线程，要是用进程，创建和销毁的代价是很难承受的

2）需要进行大量计算的优先使用线程

所谓大量计算，当然就是要耗费很多CPU，切换频繁了，这种情况下线程是最合适的。

这种原则最常见的是图像处理、算法处理。

3）强相关的处理用线程，弱相关的处理用进程

什么叫强相关、弱相关？理论上很难定义，给个简单的例子就明白了。

一般的Server需要完成如下任务：消息收发、消息处理。“消息收发”和“消息处理”就是弱相关的任务，而“消息处理”里面可能又分为“消息解码”、“业务处理”，这两个任务相对来说相关性就要强多了。因此“消息收发”和“消息处理”可以分进程设计，“消息解码”、“业务处理”可以分线程设计。

当然这种划分方式不是一成不变的，也可以根据实际情况进行调整。

4）可能要扩展到多机分布的用进程，多核分布的用线程

原因请看上面对比。

5）都满足需求的情况下，用你最熟悉、最拿手的方式

至于“数据共享、同步”、“编程、调试”、“可靠性”这几个维度的所谓的“复杂、简单”应该怎么取舍，我只能说：没有明确的选择方法。但我可以告诉你一个选择原则：如果多进程和多线程都能够满足要求，那么选择你最熟悉、最拿手的那个。

3./etc/sysctl.conf 这个文件的意义？列举一些常见的kernel参数和作用？

/etc/sysctl.conf这个目录主要是配置一些系统信息,而且它的内容全部是对应于/proc/sys/这个目录的子目录及文件。

（1）shmmax：该参数定义了共享内存段的最大尺寸（以字节为单位）。缺省为32M，对于oracle来说，该缺省值太低了，通常将其设置为2G。

   （2）shmmni：这个内核参数用于设置系统范围内共享内存段的最大数量。该参数的默认值是 4096 。通常不需要更改。     （3）shmall：该参数表示系统一次可以使用的共享内存总量（以页为单位）。缺省值就是2097152，通常不需要修改。     （4）sem：该参数表示设置的信号量。     （5）file-max：该参数表示文件句柄的最大数量。文件句柄设置表示在linux系统中可以打开的文件数量。

4.nginx的应用场景？

nginx应用场景概括

      1、http服务器。Nginx是一个http服务可以独立提供http服务。可以做网页静态服务器。

      2、虚拟主机。可以实现在一台服务器虚拟出多个网站，例如个人网站使用的虚拟机。

      3、反向代理，负载均衡。当网站的访问量达到一定程度后，单台服务器不能满足用户的请求时，需要用多台服务器集群可以使用nginx做反向代理。并且多台服务器可以平均分担负载，不会应为某台服务器负载高宕机而某台服务器闲置的情况。

5.lvs脑裂如何解决，为什么会产生双master？双master时vip通不通？

作者：灯下黑  
链接：https://www.zhihu.com/question/50997425/answer/136178424  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

可以采用第三方仲裁的方法。由于keepalived体系中主备两台机器所处的状态与对方有关。如果主备机器之间的通信出了网题，就会发生脑裂，此时keepalived体系中会出现双主的情况，产生资源竞争。

一般可以引入仲裁来解决这个问题，即每个节点必须判断自身的状态。最简单的一种操作方法是，在主备的keepalived的配置文件中增加check配置，服务器周期性地ping一下网关，如果ping不通则认为自身有问题 。

最容易的是借助keepalived提供的vrrp\_script及track\_script实现。如下所示。

初始的配置:

global\_defs {

router\_id LVS\_DEVEL

}

vrrp\_sync\_group VG\_1 {

group {

VI\_1

}

notify\_master /root/slave2master.py

notify\_backup /root/master2slave.py

}

vrrp\_instance VI\_1 {

vrrp\_unicast\_bind 192.168.150.21

vrrp\_unicast\_peer 192.168.150.20

state BACKUP

interface manbr

virtual\_router\_id 51

nopreempt

priority 100

advert\_int 10

authentication {

auth\_type PASS

auth\_pass XXXX

}

virtual\_ipaddress {

192.168.150.23 dev manbr

}

}

增加仲裁的配置：

global\_defs {

router\_id LVS\_DEVEL

}

track\_script {

check\_local

}

vrrp\_script check\_local {

script "/root/check\_gateway.sh"

interval 5

}

vrrp\_sync\_group VG\_1 {

group {

VI\_1

}

notify\_master /root/slave2master.py

notify\_backup /root/master2slave.py

}

vrrp\_instance VI\_1 {

vrrp\_unicast\_bind 192.168.150.21

vrrp\_unicast\_peer 192.168.150.20

state BACKUP

interface manbr

virtual\_router\_id 51

nopreempt

priority 100

advert\_int 10

authentication {

auth\_type PASS

auth\_pass XXXX

}

virtual\_ipaddress {

192.168.150.23 dev manbr

}

}

check\_gateway.sh 就是我们的仲裁逻辑，发现ping不通网关，则关闭keepalived service keepalived stop。

该方法在服务器通信正常之后无法再将keepalived启动起来