**目标每天200行代码**

**认识shell,什么是shell? 功能就是把命令翻译给机器的工具---解释器**

**企业的服务器很可能是在虚拟机上进行的**

**Day 1 shell脚本编程 2019-7-5**

**# Pstree(意思查看进程数)**

**# cat /etc/shells #查看解释环境**

**不想用之前的解释器可以新装解释器:**

**# yum -y install ksh #装新的解释器ksh**

**进入解释器环境直接输入解释器名: # ksh**

**Ctrl+a 行首 ctrl+e 行尾**

**企业安全考虑人离开服务器正确操作用exit 登出服务器,进入再输入密码即可**

**Bash的优点:历史记录,快捷键,tab键 ,管道,重定向**

**标准的脚本构成**

1. **声明解释器**

**#!/bin/bash**

1. **注释脚本功能,变量含义,每个步骤要解决的问题等等**
2. **具体代码**

**脚本执行方式:**

1. **赋予x执行权限,然后使用绝对路径或相对路径运行该文件**
2. **使用解释器直接执行脚本,及时没有x权限也可以(新建的一个解释器,解释完退回之前的解释器,解释完就跑)**

**Bash test1.sh**

1. **使用source命令执行脚本 (使用当前的解释器,解释完不跑)**

**Source test1.sh 雷同 (. test2.sh)**

**编写一键部署yum脚本**

**#!/bin/bash**

**Rm -rf /etc/yum.repos.d/\*.repo**

**echo “**

**[abc]**

**name=test**

**baseurl=http://172.25.254.254/content/rhel7.0/x86\_64/dvd/**

**enabled=1**

**gpgcheck=0” > /etc/yum.repos.d/abc.repo**

**编写一键部署vsftpd脚本**

**#! /bin/bash**

**yum -y install vsftpd &> /dev/null**

**systemctl start vsftpd**

**systemctl enable vsftpd**

**# Systemctl stop firewalld //脚本写完关闭防火墙测试服务**

**常量 固定不变的量**

**变量 储存可能会发生变化的内容,增加脚本灵活**

**变量类型:常用变量四种**

1. **自定义变量 (叫什么名字给什么内容,赋予一个值)**

**# $a 调用a变量**

**# Unset a 取消变量**

**# echo ${a}RNB =10RNB**

1. **环境变量**

**#echo $USER**

**#echo $HOME (家目录)**

**#echo $UID (编号)**

**#echo $SHELL (解释器)**

**#echo $PS1 (输出一级提示符)**

**#echo $PS2 (输出二级提示符)**

**#echo $PATH (命令程序存放路径)**

1. **位置变量 $1 $2 $3......**
2. **预定义变量 ($0 $$ $! $# $? $\* )**

**[root@server0 opt]# cat test5.sh**

**#!/bin/bash**

**echo $1**

**echo $2**

**echo $3**

**echo $4**

**echo $$　　　当前执行的程序进程号**

**echo $#　　　所有位置变量的个数**

**echo $\*　　　　所有位置变量**

**echo $?　　　判断上一条指令是否执行成功**

**echo $0　　　执行的程序名**

**Echo　$!　　最后一条放入后台的程序进程号**

**[root@server0 opt]# bash test5.sh a b c d**

**a**

**b**

**c**

**d**

**1398**

**4**

**a b c d**

**0**

**test5.sh**

**$$ -->当前脚本的进程号**

**$# -->显示位置变量的个数**

**$? -->检测上一条指令是否成功.成功是0**

**$0 -->执行的程序名**

**$\* -->显示所有位置变量的值**

**$! -->最后一条放入后台的程序进程号**

**# env #查看所有环境变量**

**# set #查看所有变量**

**# set | grep xxoo #显示 xxoo这行的信息**

**“” 双引 -->界定范围**

**‘’ 单引 -->界定范围 ,使特殊符号失效,有特殊符号不能用.没有的情况下可以用**

**`` 反撇号 等同于 $() 效果一样**

**a=date #定义变量a,内容是date**

**a=`date` 将date的内容输出赋值给变量a, # echo $a 是将date的内容输出**

**Stty -echo //关闭回显 在输入密码的时候可以使用**

**Stty echo //恢复回显**

**Export a=10 #发布全局变量**

**export a #对已有的变量,发布全局变量**

**Export -n a #取消全局变量,恢复局部变量**

**计算数字**

**1. # Expr 2 - 1**

**+ - \\* / %(余数)**

**\转译符号,屏蔽之后一个字符**

**a=10**

**Expr $a + 5 --->15**

**2. # echo $[4+8] -->12**

**# echo $[15\*5+52\*52+95\*6-5+6/5]**

1. **Let 可以改变变量本身的值,不显示结果**

**加1非主流写法# let a=a+1**

**let a=a-1**

**Echo $a**

**加1主流写法 let a++ let a--**

**加2 let a+=2 let a/=4 let a-=3**

1. **Bc 计算器**

**一般使用非交互式 echo “10/3” | bc**

**# echo "scale=3;10/3" | bc //显示3位小数点 定义小数点后面长度是3位**

**# echo "scale=4;10/3" | bc //显示4位小数点**

**Day 2 shell 脚本条件测试 if for whiel 循环**

**--管理员的UID等于0,普通用户的UID都是1000以上的**

**条件测试常用两种 1.test 2.加表达式**

**==等于 !=不等于 -z判断是否为空(!-z取反等同于 (-n,测试时,变量加引号)) &&并且 ||或 ;分开执行不同**

**逻辑判断 &&(并且) ||(或) 可以合并使用**

**--# [ $USER == root ] && echo 123 //如果前面的条件为真则执行 &&后面的命令,相反则不执行**

**--# [ $USER != root ] || echo 123 //如果前面的条件为假则执行后面的命令**

**--# [ $USER != root ] && echo 123 || echo 123**

**--# [ -z "$a" ] -z 判断变量a的值是否为空,空就是正确的**

**--# [ -z "$a" ] && exit**

**--# [ ! = $a ] != 意思是不等于**

**新命令: yum list installed 检查是否安装某个安装包**

**yum list installed | grep vsftpd**

**纯数字比较: 而==是比较字符串的,两者有不同的区别**

**-eq(等于)**

**-gt(大于)**

**-lt(小于)**

**-le(小于等于)**

**-ge(大于等于)**

**-ne(不等于)**

**案例:**

**编写脚本,每2分钟检查系统登录人数,如果超过三人,则警报并发邮件给管理员**

**#Vim test5.sh**

**#! /bin/bash**

**n=`who | wc -l`**

**[ $n -gt 3 ] && echo "有人入侵" | mail -s test root**

**注意:邮件发送有两种方式**

**第二种 mail -s test2.sh < a.txt /需要提取准备a.txt,文件的内容就是邮件,a.txt需要新建**

**Echo “有人侵犯” | mail -s test2.sh < a.txt**

**# Crontab -e**

**\*/2 \* \* \* \* /opt/test5.sh**

**加！可取反测试**

**-e 判断是否有这文件(目录或文件)**

**六个文件测试 -e -f -d 　　　　　　　 -r -w -x　　可读可写可执行**

**-f是否存在，部分文件类别**

**-d 是否存在，且是目录**

**可以结合 ! 使用 ! -e 叹号后有空格**

**# [ -e /opt/a ] # [ ! -e /opt/a ]**

**逻辑符号汇总:**

**== != -z ! -z(-n,测试时变量加引号)**

**-eq(等于) -gt(大于) -lt(小于) -le(小于等于) -ge(大于等于) -ne(不等于)**

**-e -f -d -r -w -x (管理员不受-rwx的限制)**

**&& || ;**

**逻辑符号在脚本中一行代码中尽量用不超过三个,太丑,可以用if for 等循环语法**

**If 单分支**

**If 条件测试:then**

**命令序列**

**Fi**

**if多分支**

**If 条件测试 ;then**

**命令序列1**

**Elif**

**命令序列2**

**Elif**

**命令序列3**

**Elif**

**命令序列4**

**Else**

**fi**

**案例走起: 测试本机ping 通不通**

**# Ping -c 3 172.25.0.11 ping三次-c 3**

**# ping -c 3 -i 0.2 -W 1 172.25.0.11 -c ping的次数 -i 频次 -W不管通不通都把信息反馈回来的时间调节**

**#!/bin/bash**

**ping -c 3 -i 0.2 -W 1 $1 &> /dev/null**

**if [ $? -eq 0 ];then**

**echo "$1通了"**

**else**

**echo "$1不通"**

**fi**

**新命令: # echo $RANDOM //随机数**

**案例走起:猜数大小**

**#!/bin/bash**

**x=$[RANDOM%11]**

**read -p "请输入一个您的数" n**

**if [ $x -eq $n ];then**

**echo "猜对了"**

**elif [ $n -gt $x ];then**

**echo "猜大了"**

**else**

**echo "猜小了"**

**fi**

**For循环**

**For 变量名 in 列表值**

**Do**

**命令序列**

**Done**

**{1..100} 等同于 seq $a**

**#!/bin/bash**

**for i in {1..15}**

**do**

**ping -c 3 -i 0.2 -W 1 172.25.0.$i &> /dev/null**

**if [ $? -eq 0 ];then**

**echo "$i通了"**

**else**

**echo "$i不通"**

**fi**

**done**

**案例:**

**#!/bin/bash**

**a=0**

**b=0**

**for i in {1..15}**

**do**

**ping -c 3 -i 0.2 -W 1 172.25.0.$i &>/dev/null**

**if [ $? -eq 0 ];then**

**echo "通了"**

**let a++**

**else**

**echo "不通"**

**let b++**

**fi**

**done**

**echo "$a台通了,$b台不通"**

**While 循环**

**While 条件测试**

**Do**

**任务序列**

**Done**

**#!/bin/bash**

**x=$[RANDOM%101]**

**a=0**

**while :**

**do**

**let a++**

**read -p "请输入一个数字(0-100)" n**

**if [ $x -eq $n ];then**

**echo "猜对了!猜了$a次"**

**exit**

**elif [ $n -lt $x ];then**

**echo "猜小了"**

**else**

**echo "猜大了"**

**fi**

**done**

**Day 3 case\函数**

**Case 分支,是简化版本的if,代码编写比if精简,但功能没有if强大,case可以替代if**

**case 结构**

**Case 变量 in**

**模式1)**

**命令序列;;**

**模式2)**

**命令序列2;;**

**模式3)**

**命令序列3;;**

**Esac**

**一键部署nginx软件脚本**

**真机:**

**1.$ cd linux-soft**

**$ cd 02**

**2.$ tar -xf lnmp\_soft.tar.gz -C /home/student**

**$ scp lnmp\_soft/nginx-1.12.2.tar.gz [root@172.25.0.10:/opt](mailto:root@172.25.0.10:/opt)**

**3. Vim test1.sh**

**#!/bin/bash**

**yum -y install gcc openssl-devel prce-devel**

**tar -xf /opt/nginx-1.12.2.tar.gz**

**cd nginx-1.12.2**

**./configure**

**make**

**make install**

1. **运行nginx服务**

**Cd /usr/local/nginx/sbin**

**./nginx**

**关闭防火墙后,打开火狐浏览器测试**

**# netstat -ntulp | grep -q nginx //在一堆的服务中寻找nginx 服务 -q沉默不输出**

**#echo $? //查看nginx**

**编写运行nginx 软件脚本**

**Vim test3.sh**

**#!/bin/bash**

**case $1 in**

**st|start)**

**/usr/local/nginx/sbin/nginx;;**

**stop)**

**/usr/local/nginx/sbin/nginx -s stop;;**

**rest)**

**/usr/local/nginx/sbin/nginx -s stop**

**/usr/local/nginx/sbin/nginx;;**

**stat)**

**/netstat -ntulp | grep -q nginx**

**[ $? -eq 0 ] && echo "服务已启动" || echo "服务未启动";;**

**\*)**

**echo "st|stop";;**

**esac**

**# echo -e "\033[32mABC\033[0m"**

**# echo -e "\033[35mHELLO\033[0m" //更改颜色**

**函数: 可以定义公共的语句块,在脚本中反复调用,达到精简脚本的目的**

**格式:**

**函数名()**

**{**

**}**

**案例:1**

**#!/bin/bash**

**abc(){ //定义函数,名字是abc**

**mkdir $1 // 函数中的任务是创建并进入目录**

**cd $1**

**}**

**abc abc**

**案例2:**

**#!/bin/bash**

**a(){**

**echo -e "\033[$1m$2\033[0m"**

**}**

**a 32 hello**

**a 33 hello**

**a 34 hello**

**执行测试 . Test4.sh 或者a 43 hello**

**下午**

**Exit 终止脚本程序**

**Break 跳出循环,执行循环后台任务**

**Continue 结束本次循环,继续下一次循环**

**案例:**

**#!/bin/bash**

**x=0**

**while :**

**do**

**read -p "请输入数字(0是结束)" n**

**[ -z $n ] || [ $n -eq 0 ] && break**

**let x+=n**

**done**

**echo "所有数字之和是$x"**

**字串截取的方法**

**a=abcde**

**截取de**

**echo ${a:3:2}**

**RANDOM //随机数**

**案例:**

**#!/bin/bash**

**a=abcdefghijklmnopqrst**

**x=$[RANDOM%21]**

**echo ${a:x:1}**

**案例:**

**#!/bin/bash**

**#c=''**

**a=abcdefghijklmnopqrst**

**for i in {1..8}**

**do**

**x=$[RANDOM%21]**

**# b=$[$x:$[RANDOM]:1]**

**pass=${a:x:1}**

**pass2=$pass2$pass**

**done**

**echo $pass2**

**执行的时候用bash 解释器**

**字串替换**

**Echo ${变量名/old/new}**

**字串删除**

**##\* echo ${变量名#\*关键词}**

**echo ${变量名##\*关键词}**

**%% \***

**echo ${变量名%关键词\*}**

**echo ${变量名%%关键词\*}**

**案例:更改文件名**

**定义变量初值(备用值)**

**Echo ${变量名:-初值}**

**Echo ${a:-123}**

**案例:**

**#!/bin/bash**

**read -p "请输入用户名:" u**

**useradd $u**

**stty -echo**

**read -p "请输入密码:" p**

**stty echo**

**echo**

**echo ${p:-123} | passwd --stdin $u**

**echo "输入的用户名是$u, 输入的密码是$p"**

**Day4**

**Exit**

**Break**

**Continue**

**字串处理**

**${a:0:1} 截取**

**${a/} 替换**

**${a#} 掐头**

**${a%} 去尾**

**${a:-123} 赋初值**

**正则表达式 基本正则列表**

**以下所有的字符在正则表达式里用,有些在linux 环境中表示的意思不一样**

**概念:使用一串字符来描述有共同属性的数据**

**基本正则表达式符号**

**符号:^ 以什么开头 $匹配结尾字串 ^$匹配空行**

**[root@server0 ~]# grep ^root txt**

**root:x:0:0:root:/root:/bin/bash**

**Head -2 /etc/passwd > txt**

**Grep root txt //找有root 的行**

**Grep ^root txt //找有以root 开头的行**

**Grep nash$ txt // 找有以bash 结尾的行**

**Grep ^$ txt // 找空行**

**Grep bash$ /etc/passwd | wc -l //统计以bash 结尾的行数 --wc统计行数**

**Grep bas[hg] //查找有字母b或者i或者n的行**

**Grep “[root]” txt //找匹配有有字母r或者让o或者t里任意字符的行**

**Grep “[^rot]” txt //正则表达式里的取反,找出除了r或者o或者t的行**

**Grep “[a-z]” /etc/passwd // 匹配所有小写字母的行**

**Grep “[^a-z]” /etc/passwd //找拥有除了小写字母之外的行**

**Grep “[A-Z]” /etc/passwd // 匹配所有大写字母的行**

**Grep “[0-9]” /etc/passwd // 匹配有所有数字的行**

**Grep “rot..” txt // 找root,后面追加2个任意字符的行**

**grep “ro\*” txt //匹配\*号前面的任意字符**

**grep “bo\*i” txt //找boi,o可以出现任意次数**

**bin:x:1:1:bin:/bin:/sbin/nologin**

**Grep “ro\*ot” txt //找root,第一个o可以出现任意次数**

**grep “.\*” txt //找所有字符**

**Grep “\*” txt //不能单独使用,不然什么也找不到**

**Grep “.” txt //找任意单个字符**

**Grep “.oot” //找某字符开头后面是oot的行**

**[root@server0 opt]# grep "ro\{2,3\}t" txt //范围 找root 或者rooot**

**root:x:0:0:root:/root:/bin/bash**

**[root@server0 opt]# grep "ro\{2\}" txt**

**root:x:0:0:root:/root:/bin/bash //精准 找root**

**grep "ro\{2\}t" txt //找root**

**grep "ro\{1\}t" txt //找rot**

**扩展正则列表,由1升级而来**

**Egrep “ro+t” txt 等同于”ro\{2,3\}t”**

**Egrep “ro?ot” txt //找root,第一个o可以实现0次或1次**

**Egrep “ro?t” txt //找rot,o可以出现0次或1次**

**Grep “ro\{0,1\}ot” txt //使用基本正则实现相同效果**

**egrep "ro{0,1}ot" txt //扩展正则更精简**

**egrep "r|o|t" txt //寻找有r或者o或者t的行**

**egrep "\bbin\b" txt //找bin,前后不能是数字或者是字母,下划线**

**\b 空格.空.tab,特殊符号 可以找出来**

**Egrep 等同于grep -E ／／grep过滤工具　的扩展正则**

**下午**

**Sed 流式编辑器,可以非交互式修改文本,逐行操作**

**增删改查**

**两种用法**

**1 前置命令 | sed 选项 (定址符)指令**

**2 sed | 选项 (定址符)指令 文本**

**选项:三个常用的**

**-n 屏蔽默认输出**

**-r 支持扩展正则**

**-i 写入文件**

**指令**

**p 输出指定内容**

**# df | sed -n 'p' //输出df指令生成的文本中的第一行**

**# sed -n '1,2p' txt //用p指定输出1到2行的内容**

**# sed -n ‘p’ txt //输出所有行一次**

**Sed ‘p’ txt //输出所有行2次**

**Sed -n ‘1p’ txt //输出第一行**

**# sed -n "1,+2p" txt //输出1行加后面的两行,123行**

**# sed -n '1p;3p' txt //输出1行和3行**

**# sed -n '1~2p' txt //输出1,3,5行 输出1然后隔两行再输出也就是到了3,然后再隔2就是5**

**# sed -n "$=" txt //显示最后一行的行号 $是最后一行**

**# sed -n "=" txt //显示每一行的行号**

**查看总行数的两种方法:**

**[root@server0 opt]# sed -n '$=' /etc/passwd**

**39**

**[root@server0 opt]# cat /etc/passwd | wc -l**

**39**

**指令d 删除**

**Sed -n “5d” txt //用法跟p差不多**

**sed "5d" txt //删除第五行**

**Sed “1~2d” txt //删除奇数行**

**sed "2~4d" txt //删除偶数行**

**# sed -i '5d' txt //删除第五航,并写入文件 i能 删源文件**

**# sed -r "/bash|nologin/d" txt //删除以root开头的文件**

**# sed "/^root/d" txt //删除有bash或nologin的行**

**指令s 替换**

**sed "s/2017/oooo/" test.txt**

**sed "s/2017/xxoo/g" test.txt**

**sed "3s/2017/xxoo/2;3s/2017/xxoo/2" test.txt //**

**sed "s#/nologin#/sbin/sh#" txt //**

**# sed "s/.//2;s/.$//" test3.sh //删除最后一个和第二个字符**

**# sed "s/[0-9]//g" test3.sh**

**\(\) 保留 \1\2\3**

**() 划分范围**

**sed -r "s/^(.)(.\*)(.)$/\3\2\1/" test3.sh //将文本中的第一个字符和最后一个字符互换**

**# sed -r "s/^(.)(.\*)(.)(.$)/\3\2\1\4/" test3.sh //将文本中的第一个字符和倒数第二个字符互换**

**Day5**

**扩展正则继续**

**指令a行后追加 i行前追加 c替换行**

**# sed '1,3a 8888' t1.txt**

**# sed '2a 8888' t1.txt**

**# sed 'i 8888' t1.txt**

**# sed '1c 9999' t1.txt**

**Awk 工具的概述(搜索　过滤文本)**

**跟sed同样也有编辑的功能 最大的功能是搜索**

**格式有两种:**

**1 前置命令**

**2 awk [选项] ‘’ 注意一定要使用单引号,双引号是把整行的信息输出**

**# yum provides awk //查看软件包是否有awk这个包**

**$1 $2 $3 ......NR-行 NF-列**

**Awk -F [:/] ‘{print 9$}’ txt1 //使用多个分隔符后显示第9列**

**Awk -F [:/] ‘{print NF}’ txt1 //显示第几列**

**Awk -F [:/] ‘{print NR}’ txt1 //显示第几行**

**Awk -F [:/] ‘{print NR NF}’ txt1 //都显示,看不清**

**Awk -F [:/] ‘{print NR,NF}’ txt1 //加逗号有空格效果**

**Awk -F [:/] ‘{print NR “”NF}’ txt1 //或把空格引起来**

**Awk -F [:/] ‘{print “当前行有” NF”列”}’ txt1 //常量配合变量输出**

**# ifconfig eth0 | awk '/RX p/{print "接收的流量是" $5"字节"}' //**

**# df -h | awk '/vda1/{print "可用大小是"$4}' //**

**# awk '/Failed/{print $11}' /var/log/secure**

**逐行任务**

**Awk [选项] ‘BEGIN{print }{指令}{指令}’ # \t tab空格键**

**三大任务 :前置任务 逐行任务 结束任务**

**案例:**

**#awk-F: 'BEGIN{print "User\tUID\tHome"}{print $1"\t"$3"\t"$6}END{print "总计 "NR" 行"}' txt**

**User UID Home**

**root 0 /root**

**bin 1 /bin**

**daemon 2 /sbin**

**adm 3 /var/adm**

**lp 4 /var/spool/lpd**

**总计 5 行**

**//按需求的格式输出文档的内容,有表头,有结尾统计 空格可以使用 \t , 空格等**

**Awk 流程控制**

**BEGIN任务 执行1次**

**逐行任务 执行多次,与文本有关**

**END任务 执行1次**

**Awk 处理条件**

**1.正则**

**# awk -F: '$1!~/root/' txt //　~号是包含 !~是不包含**

**//输出第七列不包含nologin 的行的第一列和第七列**

1. **使用数值/字符串比较设置条件**

**比较符号 ==(等于) !=(不等于) >(大于)**

**# awk 'NR==4{print}' txt //查看第五行**

**# awk 'NR!=4{print}' txt //不看第五行**

**# awk -F: '$1=="root"' txt //查找第一例列等于root 的行,注意是字母的要用引号引起来**

**# awk -F: '$3==0' txt //找第三列等于0的行**

**Awk -F: ‘$3<=10’ /etc/passwd //找第三列小于等于10 的行**

**Awk -F: ‘$3>=1000’ /etc/passwd //找第三列大于等于1000的列**

1. **使用逻辑符号 && ||**

**Awk -F: ‘$3>10 || $<1000’ /etc/passwd //找大于10 或者小于1000的行**

**Awk -F: ‘$<10 && $>1000’ /etc/passwd //逻辑错误找不到**

**案例:**

**#!/bin/bash**

**u=`awk -F: '/bash$/{print $1}' /etc/passwd`**

**for i in $u**

**do**

**grep $i /etc/shadow | awk -F: '{print $1 ,"-->", $2}'**

**Done**

**扩展**

**Sort排序命令中常用的三个选项:-n按数字升序排序 -k针对指定的字段进行排序 -r反向排序**

**Cut做切割的时候,指定分割符的选项是 -d -f是指定选取的字段**

**$0 是所有列**

**Day 6**

**Awk　数组**

**数组可以使用一个数组名称储存多个变量**

**数组名［下标］＝元素名**

**# awk 'BEGIN{a[0]=1;a[1]=20;print a[0],a[1]}'**

**注：　字符加双引号**

**# awk '{a[$1]++}END{for(i in a){print i,a[i]}}' a.txt**

**准备测试环境：**

**Yum -y install httpd //安装服务**

**Netstat -ntulp | grep :80　　　／／检查哪个服务占用了８０端口**

**/usr/local/nginx/sbin/nginx -s stop　　　／／如果是ｎｇｉｎｘ需要关闭**

**Killall nginx**

**Systemxtl srestart httpd　　　／／开启服务**

**Echo 123 > /var/www/html/index.html 　创建ｈｔｔｐｄ的默认页面**

**Curl <http://172.25.0.11/index.html>　　　／／测试**

**Cat /var/log/httpd/access\_log //ｈｔｔｐｄ的日志，每次被访问，都会添加新纪录**

**# systemctl stop firewalld.service 　　　／／关闭防火墙**

**# firewall-cmd --reload　　　　　　　　　　　　　　／／重新加载防火墙**

**# awk '{ip[$1]++}END{for(i in ip) {print i,ip[i]}}' /var/log/httpd/access\_log　　　　　　　／／用ａｗｋ统计日志中的访问量**

**ab**

**Yum -y install httpd-tools //ab工具的安装包**

**# ab -c 1 -n 10000 <http://172.25.0.11/index.htnl> //　使用ａｂ工具测试网站访问量，－ｃ　是用户名　－ｎ是访问次数**

**Ll -lh 文件路径　　　／／查看文件大小**

**排序：**

**sort -r txt //sort 排序默认升序，-r 降序　　　注：排序是按首个数字进行排序**

**Sort -n txt //sort //升序**

**Sort -rn txt //按整个数字大小进行排序**

**在脚本中不能使用top 查看排序，top直接进入到一个程序里了**

**[root@server0 opt]# uptime | awk '{print "cpu的平均负载是"$10,$11,$12}'cpu的平均负载是0.05**

**[root@server0 opt]# df -h | awk '/vda1/{print "硬盘剩余的空间是"$4}'**

**硬盘剩余的空间是7.0G**

**[root@server0 opt]# p=`rpm -qa | wc -l`**

**[root@server0 opt]# echo "当前主机安装的软件包数量是$p"**

**当前主机安装的软件包数量是1213**

**[root@server0 opt]# x=`who | wc -l`**

**[root@server0 opt]# echo "当前登录账户数量是"$x**

**当前登录账户数量是2**

**[root@server0 opt]# psn=`ps aux | wc -l`**

**[root@server0 opt]# echo "当前运行的进程数是"$psn**

**当前运行的进程数是127**

**[root@server0 opt]# free -m | awk '/^Mem/{print "主机剩余内存是"$4"M"}'**

**主机剩余内存是120M**

**[root@server0 opt]# awk '{x++}END{print "主机拥有的账户总数是"x"个"}' /etc/passwd**

**主机拥有的账户总数是40个**

**Sleep 2 睡眠２秒**

**清除屏幕信息 clear**

**查看服务器实时状态信息的脚本案例：**

**Vim test.sh**

**#!/bin/bash**

**while :**

**do**

**uptime | awk '{print "cpu的平均负载是"$10,$11,$12}'**

**df -h | awk '/vda1/{print "硬盘剩余的空间是"$4}'**

**p=`rpm -qa | wc -l`**

**echo "当前主机安装的软件包数量是$p"**

**x=`who | wc -l`**

**echo "当前登录账户数量是"$x**

**psn=`ps aux | wc -l`**

**echo "当前运行的进程数是"$psn**

**free -m | awk '/^Mem/{print "主机剩余内存是"$4"M"}'**

**awk '{x++}END{print "主机拥有的账户总数是"x"个"}' /etc/passwd**

**ifconfig eth0 | awk '/RX p/{print "网卡eth0接收的数据量是"$5"字节"}'**

**sleep 3**

**clear**

**Done**

**# awk '/Failed/{ip[$11]++}END{for(i in ip){print i,ip[i]}}' /var/log/secure**

**案例**

**编写监控脚本，如果查看有人访问本机密码输入错误超过三次则发邮件通知管理员**

**#!/bin/bash**

**x=`awk '/Failed/{ip[$11]++}END{for(i in ip){print i","ip[i]}}' /var/log/secure`**

**for i in $x**

**do**

**p=${i%,\*}**

**s=${i#\*,}**

**[ $s -gt 3 ] && echo "报警！$p访问本机失败了$s次，赶紧去灭火"**

**done**

# **NSD Operation DAY01**

**Day1 Nginx**

**Setip ／／在虚拟机直接陪网卡ｉｐ**

**实验１**

**搭建nginx**

**Gcc　　 c语言的解释器**

**Openssl -devele ／加密和解密用，不会直接用，在用的时候nginx会自动调用**

**指定用户和组的目的是为了安全**

**Ｒｏｏｔ起服务很危险，为什么呢?因为ｒｏｏｔ起的服务直接授予访问权限给ＨＴＴＰ，相当于ｈｔｔｐ可以共享ｒｏｏｔ里的所有数据**

**生产环境里绝大多数的软件包是用源码包安装的－目的是为了安全，版本的更新，ｙｕｍ是四年以前的方法，现在是ｃｅｎｔｏｎ７．０了**

**＼　号是为了换行**

**# netstat -anptu | grep nginx　　查看是否有某个软件**

**:99,115s/#// //vim末行模式里把99到115行前面的#替换成空**

**Timeout //超时时间**

**Day 2 nginx 高级运维**

**SSL 虚拟主机 配置基于加密网站的虚拟主机**

**服务端实现目标:**

**1.域名为www.c.com**

1. **该站点通过https访问**
2. **通过私钥,证书对该站点所有数据加密**

**客户端一定要改配置文件 vim /etc/hosts //本地域名解析文件**

**客户端访问的时候一定要加htpps**

**LNMP (linux,nginx,mysql,php) 任何一步骤少了都不能正常访问网页**

**静态页面,动态网页,现在基本是两者相结合**

**模块化设计:**

**PHP ------>mysl**

**PHP ------->oracle**

**PHP -------->sqlserver**

**PHP----------->NB**

**||**

**数据库的公司比较多,用什么安装什么,例如用sqlserver 就安装sqlserver的扩展服务包**

**LNMP**

**Nginx配置动静分离**

**如果用户访问的是静态数据,则到html目录找,直接返回,找不到则报错404 文件找不到**

**如果用户访问的是动态数据,到html目录找,转发给9000端口去匹配**

**用户[php,java,python,perl,basic,.net,c**

**Fpm 的端口是9000 fpm是个多进程软件 指的是客户打开一个软件,但进程上显示打开多个软件 ,啥好处呢? 可以同时干多个事情,干活的人多.缺点:占用能存空间比较大,一个进程百分之三左右的空间**

**Nginx 里配置location,相当于if语句,但是不能用if (支持正则) 没有顺序要求,/根的优先级最低,但是根能匹配一切**

**Location 匹配用户的地址栏**

**Location /abc { deny all; }**

**Location /ttt { allow 11111; }**

**Location /test { allow all; }**

**Firefox <http://www.a.com/abc>**

**Firefox <http://www.a.com/ttt>**

**;号代表注释**

**<?php php的脚本开头**

**在nginx中 ~ 表示正则匹配的意思 \.屏蔽转意的意思**

**更改php主配置文件改三个地方:**

**1.67-73行注释去掉**

1. **最长那行是错的删掉或注释掉**
2. **Include fastcgi.后面的名字是错的改为.conf**

**LNMP 常见问题**

**Nginx的默认访问日志文件为/usr/local/nginx/logs/access.log**

**Nginx的默认错误日志文件为/usr/local/nginx/logs/error.log**

**PHP默认的错误日志文件为/var/log/php-fpm/www-error.log**

**用日志可以精确的查找问题,精确到第几行**

**User --->nginx---->php---->mysql**

**没有启动服务器**

**没有安装**

**Nginx 高级技术 地址重写(地址栏被重写)**

**rewrite regex replacement flag**

**rewrite 旧地址 新地址 [选项]**

**Regex 正则表达式的意思**

**Day 3**

**Nginx 代理服务器(调度器)**

**nginx除了当web服务器用 还可以当代理服务器使用**

**代理服务器有两个功能:**

**一个是调度器(轮询算法)**

**一个是健康检查**

**#weight设置服务器权重值，默认值为1**

**#max\_fails设置最大失败次数**

**#fail\_timeout设置失败超时时间，单位为秒**

**#down标记服务器已关机，不参与集群调度**

**Ip\_hash; =5mub**

**Nginx 基本上可以调度一切服务**

**----Nginx.org nginx官网 可以查看所有课件上的源代码**

**中国GB2312**

**编码ASCII 计算机编码起源**

**Apple (万国编码utf8)**

**200 一切正常**

**报错400以上 代表客户端错**

**500以上代表服务器错**

**Day4**

**查看状态信息:**

**Pv (page view页面量) uv(user view 用户量)**

**Tcp协议(3次握手建立连接,4次断开)**

**Client ----------------------baidu**

**3次握手(syn,syn+ack,ack) 30k**

**建立连接**

**请求(看图片) 1k**

**4次断开 40k**

**现在基本上是1次连接多次请求**

**Active connections：当前活动的连接数量。**

**Accepts：已经接受客户端的连接总数量。**

**Handled：已经处理客户端的连接总数量。**

**（一般与accepts一致，除非服务器限制了连接数量）。**

**Requests：客户端发送的请求数量。**

**Reading：当前服务器正在读取客户端请求头的数量。**

**Writing：当前服务器正在写响应信息的数量。**

**Waiting：当前多少客户端在等待服务器的响应。**

**ab -n 2000 -c 2000 <http://192.168.4.5/> ab 工具末尾必须加/号**

**# ab -n 2000 -c 2000 <http://192.168.4.5/>**

**N100 c100**

**Ulimit u是用户 limit是限制意思**

**# ulimit -Hn 100000 //设置硬限制（临时规则)不能超过**

**# ulimit -Sn 100000 //设置软限制（临时规则）警告值**

**# kill -USR1 `cat nginx.pid`**

**# mv access.log access2.log**

**并发量 缓存 日志切割 服务状态信息 日志切割脚本 对网页进行压缩处理**

**服务器内存缓存**

**# kill -l**

**Kill 的作用:传递信号用的**

**Linux 里文件压缩比例用数字表示,数字越大压缩越好**

**多媒体文件a.MP3 MP4 文**

**1 http\_stub\_status\_module**

**2 /usr/local/nginx/logs**

**3 并发量应考虑服务器网络带宽,系统内存,web容器,程序代码**

**Day4**

**网站是有session保存在服务器上,**

**Cookies保存在客户端上,要有这两种概念**

**硬盘储存有一个特点:只要死机了数据全部丢失,有个优点就是速度快**

**Client -- proxy -- lnmp(web1)**

**lnmp(web2)**

**内存**

**数据库储存对比**

**性能 --CPU>内存>磁盘>数据库**

**价格 --CPU>内存>磁盘>数据库**

**Memcached是高性能的分布式缓存服务器,存数据没有数据库,没有数据表**

**Memcached 的储存方式 (key数据库),诶呦表的概念.只有写数据和查数据**

**set name 0 180 3 3表要储存的数据3个字节 set写数据**

**Get 查数据**

**提示：0表示不压缩，180为数据缓存时间，3为需要存储的数据字节数量**

**LNMP +memcached 通过php页面实现对memcached 服务器的数据操作**

**Php-session共享**

**Day5**

**常见的servlet容器**

**-IBM**

**-Oracle**

**-Apache**

**-RedHat**

**Java :写游戏软件/网页/财务软件/面包机(芯片)**

**面包机:烤面包,监测温度**

**Andriod(java写的)**

**Widows ios andriod linux windows(java)**

**Java是内存的消耗大户**

**Qq在andriod 里不是直接运行的,是在jvm也就是java虚拟机上运行的,一个虚拟机2个G,两个就四个G了,**

**但ios不是用java开发的,所以可以直接在ios平台运行,也就是ios的sigeG相当于andriod的16个G了**

**8080端口是tomcat**

**9000 -php-fpm**

**实际生产环境里开发会把网页打包成var或者jar包给到运维部,tomcat可以直接解压var包,直接丢到webapp网页存放路径下就可以,前提是var格式的包**

**-cf 压**

**-zf 解**

**影响网速最核心的因素是距离**

**卫星最大的特点是带宽高**

**CDN内容分发网络**

**Nslookup [www.tmooc.cn](http://www.tmooc.cn) 查询服务器**

**Day5总结**

**首先三个案例:**

**1安装部署tomcat 服务器**

**2 使用tomcat部署虚拟机**

**3 使用varnish加速web**

**1安装tomcat服务器**

**1)使用rpm安装JDK环境**

**2使用tomcat部署虚拟主机**

**一个host name 等于一个虚拟主机**

**端口集合:**

**http超文本传输协议 默认端口80**

**https安全超文本传输协议 默认端口 443**

**FTP:文件传输协议 默认端口21**

**TFTP:简单的文件传输协议 默认端口 69**

**telnet:远程管理协议 默认端口23**

**DNS: 域名解析协议 默认端口53**

**snmp:简单的网络管理协议 默认端口161**

**smtp:邮件协议(发邮件 默认端口 25**

**pop3:邮件协议(收邮件 默认端口110**

**Mysql:数据库 3306**

**Tomcat: 8080**

**Php: 9000**

**gcc readline-devel //安装软件依赖包**

**ncurses-devel //安装软件依赖包**

**pcre-devel //安装软件依赖包**

**python-docutils //安装软件依赖包**

**Day6**

**案例1：Git基本操作**

**案例2： HEAD指针操作**

**案例3：Git分支操作**

**案例4：Git服务器**

**案例5：制作nginx的RPM包**

**Git :版本控制软件**

**延伸:参与linux的开发着有两万多人**

**Windows只有两千人**

**# git clone [root@192.168.2.100:/var/git/project](mailto:root@192.168.2.100:/var/git/project)**

**Ls -a 查看空文件**

**Windows-------------->linux**

**Windows需要安装git和tortoiseGit**

**课外扩展知识:注册使用Github 非常有用,面试的时候 https://github.com**