**linux平台**

**1查看进程**

1.1查看系统进程ps aux

1.2pstree -p： 显示进程的PID -u： 显示进程的所属 查看进程树

1.3kill -1 22354 # 重启进程 按照进程号杀死进程

kill -9 22368 # 强制杀死进程

killall [选项][信号] 进程名 按照进程名称杀死进程

pkill –t 按照终踢出用户pkill -t -9 pts/1# 强制杀死从 pts/1 虚拟终端登录的进程

查看进程网络 netstat -tlun

**2系统资源查看**

**2.1 top -d 秒数： 指定top命令每隔几秒更新。默认是3秒**

top命令的交互模式当中可以执行的命令：

？或h： 显示交互模式的帮助

P： 以CPU使用率排序，默认就是此项

M： 以内存的使用率排序

N： 以PID排序

q： 退出top查看系统占用资源排名

**2.2 使用 vmstat [刷新延时 刷新次数]**

简单来说缓存（cache）是用来加速数据从硬盘中“读取”的，而缓冲（buffer）是用来加速数据“写入”硬盘的。

**2.3 w # 使用 w 命令查询本机已经登录的用户以及系统负载**

**2.4 free [-b|-k|-m|-g]：#查看内存信息**

-b： 以字节为单位显示

-k： 以KB为单位显示，默认就是以

KB为单位显示

-m： 以MB为单位显示

-g： 以GB为单位显示

（5）dmesg | grep CPU 查看启动自检信息中cpu 的自检信息。或者cat /proc/cpuinfo

（6）uname -a： 查看系统所有相关信息；-r： 查看内核版本；-s： 查看内核名称。

（7）file /bin/ls 查看操作系统位数

（8）lsb\_release –a 查看发行版本

（9）lsof [选项] #列出进程调用或打开的文件的信息

选项：

-c 字符串： 只列出以字符串开头的进程打开

的文件

-u 用户名： 只列出某个用户的进程打开的文

件

-p pid： 列出某个PID进程打开的文件

**3 定时任务crontab [选项]：**

-e： 编辑crontab定时任务

-l： 查询crontab任务

-r： 删除当前用户所有的crontab任务

第一个“\*” 一小时当中的第几分钟0-59

第二个“\*” 一天当中的第几小时0-23

第三个“\*” 一个月当中的第几天1-31

（）

第四个“\*” 一年当中的第几月 1-12

第五个“\*” 一周当中的星期几 0-7（0和7都代表星期日）

4 文件的重定向

nohup jupyter notebook > jupyter.log 2>&1 &

解释: 用&让命令后台运行, 1>或者>标识正确结果输出到某个文件，2>表示错误结果输出到某个文件。2>&1表示错误输出也输出到正确输出的文件中。

su - jupyter -c "nohup /usr/local/anaconda3/bin/jupyter notebook &>/dev/null &"设置开机启动，正确结果和错误结果都不输出。nohup表示no hang up, 就是不挂起, 于是这个命令执行后即使终端退出, 也不会停止运行.

5.history，查看所有的历史命令。存储在文件~/.bash\_hostory文件中。“！字符串”来搜索字符串开头的历史命令。

**命令别名**

alias :查看所有别名 ； alias ll=’ls –l –color=auto’；永久生效 vim /root/.bashrc 配置文件；删除别名 unalias

命令别名执行顺序：

1.绝对路径或者相对路径

2.执行别名

3.执行bash内部命令,注意这里的内部命令是shell自带的。并没有执行文件和ls不同。Ls属于第四种是有执行文件的。

4.执行按照$PATH环境变量定义的目录查找顺序找到的第一个命令。指的的/bin；/sbin目录下的一些软件命令。查看$PATH的内容在用echo $PATH

**Bash 快捷键**

ctrl+U 清除当前行 ctrl + Y 粘贴当前剪切的部分 ctrl + R搜索历史记录 ctrl + D 退出当前bash。

**BASH编程**

**bash多命令连续的执行。**

命令1；命令2 顺序执行。命令1&&命令2，命令1正确执行命令2才会执行。命令1||命令2 命令1不正确执行，命令2才会执行。

**三种引号的作用**

单引号里面的‘/’，‘$’等特殊格式字符都会变成普通字符，双引号会认这些字符。

反引号` `和$()一样，先执行其中的bash命令的结果。比如`date`或者$(date)。引用变量加$。

**Bash中的变量**

Bash中默认变量是字符型，输入不打引号的字符，字符串变量叠加直接连写即可。不需要像python 写 ‘+’。等号两边不能有空格，中间的空格用引号括起来。环境变量最好大写。set命令显示所有的变量.unset 变量名就是删除某个变量。Env 专门查看环境变量的命令。

declare给变量申明类型。Declare –p 变量名；表示查看变量名类型。Declare -i 变量名（不用写$）；表示给变量赋予整数类型; cc=$(($aa + $bb))用双小括号括起来强制规定为数值运算。

本地变量就是自定义变量，变量名称和变量内容随便写。可以新增变量.自定义变量只在当前shell中生效。

环境变量有部分固定名称是系统调用的不能更改。环境变量用export申明。参数变量不能修改变量名只能赋值。系统变量是bash预定义也是如此不能修改名称不能新增。只能赋值。环境变量在当前shell和子shell中都生效。写入配置文件后在所有的shell中生效。永久生效记得也要写入配置文件。path定义系统命令路径的变量。ps1 提示符格式的变量。echo ps1 即可知道目前系统提示符格式。

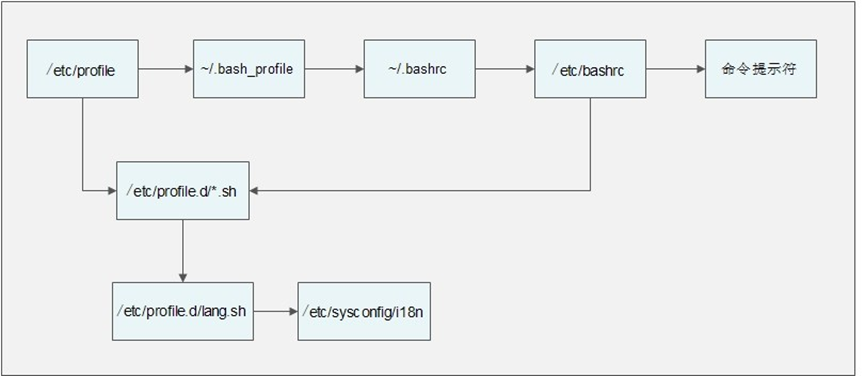
位置参数变量。是向shell脚本中传递参数的变量。在脚本中占位用$0-9，10以上用大括号，${11}。其中$0表示shell脚本本身的路径字符串。在脚本外部用空格跟随需要传入的参数。./chuancanshu.sh 11 22 33。以上都显示。$\* 是把脚本外所有参数作为一个整体传入进来。$#代表所有参数的个数。$@ 也是把所有参数一起传入。但是和$\*的区别是里面是区分个数的。也就是说可以在脚本中用for i in $@:这种循环去遍历每个参数。

位置参数变量的替代函数是read ，类似readline。直接给提示read -s -t 30 –p “please input your name” name ;隐私接受 30秒 提示

预定义变量本质上和位置参数变量是一样的。不同的是它的赋值也不能定义。$?变量表示上次命令执行的结果。0为正确执行。其他数值为错误代码。$$变量表示当前进程的PID。$!变量代表放入后台执行（语句最后带 $）的最后一个进程的PID。

**环境变量配置文件**

* 让配置文件生效用 source配置文件或者. 配置文件。
* 保存在etc中的文件对所有登录用户都生效。保存在家目录下只对当前用户生效。
* /etc/profile 定义USER变量：LOGNAME变量：MAIL变量：PATH变量： HOSTNAME变量： HISTSIZE变量：umask：调用/etc/profile.d/\*.sh文件 有语言文件
* ~/.bash\_profile 调用了~/.bashrc文件。在PATH变量后面加入了“:$HOME/bin”这个目录
* ~/.bashrc 定义默认别名调用/etc/bashrc
* /etc/bashrc 定义了PS1 定义了nologin shell下的变量umask；PATH变量；用/etc/profile.d/\*.sh文件



**其他配置文件**

* ~/.bash\_logout用户登出时执行的命令脚本。一般是一些清空或者保存的收尾命令。支持引用环境变量。如echo "Logout, `date`"，输出登出的时间。
* ~/bash\_history；记录历史命令的配置文件。HISTSIZE环境变量就是控制这个文件的记录数
* /etc/issue；本地终端欢迎信息：

转义符作用

\d 显示当前系统日期

\s 显示操作系统名称

\l 显示登录的终端号，这个比较常用。

\m 显示硬件体系结构，如i386、i686等

\n 显示主机名

\o 显示域名

\r 显示内核版本

\t 显示当前系统时间

\u 显示当前登录用户的序列号

* /etc/issue.net；远程终端欢迎信息是否显示此欢迎信息，由ssh的配置文件/etc/ssh/sshd\_config决定，加入“Banner /etc/issue.net”行才能显示（记得重启SSH服务）service ssh restart。
* /etc/motd 登陆后欢迎信息：不管是本地登录，还是远程登录，都可以显示此欢迎信息

**Bash中的运算命令**

cc=$(($aa + $bb))用双小括号括起来强制规定为数值运算。一个$和|是位与和位或

逻辑与用$$ 逻辑或用||

**Bash中的日期命令**

date +%y%m%d;#date是获得当前时间. +%y%m%d的意思是按照年月日输出.

**Shell中常用文字处理命令正以及grep中正则表达式**

* 正则表达式是搜索文件中的字符串。bash通配符\*?[]只是匹配文件名称。在正则中\*表示前一个字符出现的次数包含0次。？重复零次或一次，.表示出了换行之外的一个字符。
* Bash提取文件中的字符串是按照行列定位的。行提取用grep，文件中匹配是按照包含正则表达式的整行匹配的。（相当与python中的多行匹配。且加上了^ ……$；）列定位提取用cut提取符合条件的列，cut –d ‘:’ –f 2,4 student.txt 提取：分隔符的第2，4列 。但是在一些文件中列未必按照固定的符号来分割。比如空格数量不一致。因此需要akw函数。
* Printf函数的用法，print输出带回车，printf输出不带回车。

Printf和python最大的区别是 printf ‘%s ’ 格式后面的参数用空格隔开即可。但是pyhon是需要 % (my\_eyes, my\_hair)

* awk ‘条件1{动作1} 条件2{动作2}…’ 文件名

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

动作（Action）：格式化输出流程控制语句

awk '{printf $2 "\t" $6 "\n"}' student.txt 没有条件，动作为printf，$2 $6 为awk的格式代表文件的第二列和第6列。主要的作用是awk能识别空格和tab混合而成的分隔符。指定“：”分隔符为awk 'BEGIN {FS=":"} {printf $1 "\t" $3 "\n"}'

* 在bash中{}是有特殊含义的。所以用在正则表达式中要加转义符/.
* sed [选项] ‘[动作]’是一个用于流数据的行处理命令。

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

动作：

a \： 追加，在当前行后添加一行或多行。添加多行时，除最后 一行外，每行末尾需要用“\”代表数据未完结。

c \： 行替换，用c后面的字符串替换原数据行，替换多行时，除最后一行外，每行末尾需用“\”代表数据未完结。

i \： 插入，在当期行前插入一行或多行。插入多行时，除最后 一行外，每行末尾需要用“\”代表数据未完结。

d： 删除，删除指定的行。

p： 打印，输出指定的行。

s： 字串替换，用一个字符串替换另外一个字符串。格式为“行范围s/旧字串/新字串/g”（和vim中的替换格式类似） 。

sed -n '2,4p' student.txt 输出文件的第二第四行

* sort /etc/passwd# 排序用户信息文件。sort [选项] 文件名选项：-f： 忽略大小写-n： 以数值型进行排序，默认使用字符串型排序-r： 反向排序-t： 指定分隔符，默认是分隔符是制表符；-k n[,m]： 按照指定的字段范围排序。从第n字段开始，m字段结束（默认到行尾）
* wc [选项] 文件名选项：-l： 只统计行数。-w： 只统计单词数。-m： 只统计字符数

**BASH中的条件判断**

* [ -e /root/install.log ] 判断文件是否存在

-d 文件

判断该文件是否存在，并且是否为目录文件（是目录为真）

-e 文件

判断该文件是否存在（存在为真）

-f 文件

判断该文件是否存在，并且是否为普通文件（是普通文件为真）

-L 文件

判断该文件是否存在，并且是否为符号链接文件（是符号链接

文件为真）

* [ -w student.txt ] && echo "yes" || echo "no"；判断文件是否存在并是否具有写权限。判断部分所有者所属组和其他人。值要具有即有写权限。

-r 文件。判断该文件是否存在，并且是否该文件拥有读权限（有读权限为真）

-w文件。判断该文件是否存在，并且是否该文件拥有写权限（有写权限为真）

-x 文件。判断该文件是否存在，并且是否该文件拥有执行权限（有执行权限为真）

* [ /root/student.txt -ef /tmp/stu.txt ] && echo "yes" || echo "no" # 比较文件是否硬链接。

文件1 -nt 文件2；判断文件1的修改时间是否比文件2的新（如果新则为真）

文件1 -ot 文件2；判断文件1的修改时间是否比文件2的旧（如果旧则为真）

文件1 -ef 文件2；判断文件1是否和文件2的Inode号一致，可以理解为两个文件是否为同一个文件。这个判断用于判断硬链接是很好的方法

* [ 23 -ge 22 ] && echo "yes" || echo "no"；23是否大于等于22

整数1 -eq 整数2，判断整数1是否和整数2相等（相等为真）

整数1 -ne 整数2，判断整数1是否和整数2不相等（不相等位置）

整数1 -gt 整数2，判断整数1是否大于整数2（大于为真）

整数1 -lt 整数2，判断整数1是否小于整数2（小于位置）

整数1 -ge 整数2，判断整数1是否大于等于整数2（大于等于为真）

整数1 -le 整数2，判断整数1是否小于等于整数2（小于等于为真）

* [ -z "$name" ] && echo "yes" || echo "no"# 判断 name 变量是否为空，因为不为空，所以返回 no

-z 字符串，判断字符串是否为空（为空返回真）

-n 字符串，判断字符串是否为非空（非空返回真）

字串1 ==字串2，判断字符串1是否和字符串2相等（相等返回真）

字串1 != 字串2，判断字符串1是否和字符串2不相等（不相等返回真）

* 组合判断：aa=11；[ -n "$aa" -a "$aa" -gt 23 ] && echo "yes" || echo "no"# 判断变量 aa 是否有值，同时判断变量 aa 的是否大于 23# 因为变量 aa 的值不大于 23 ，所以虽然第一个判断值为真，返回的结果也是假

判断1 -a 判断2；逻辑与，判断1和判断2都成立，最终的结果才为真

判断1 -o 判断2；逻辑或，判断1和判断2有一个成立，最终的结果就为真

！判断逻辑非，使原始的判断式取反.

**BASH中的流程控制**

* **单分支if条件语句**

if [ 条件判断式 ];then #中括号内侧的空格一定不能少.否则出错.

程序

fi

或者

if [ 条件判断式 ]

then

程序

fi

* **多分支if条件语句**

if [ 条件判断式1 ]

then

当条件判断式1成立时，执行程序1

elif [ 条件判断式2 ]

then

当条件判断式2成立时，执行程序2

„省略更多条件 …

else

当所有条件都不成立时，最后执行此程序

fi

* **多分支case条件语句**

case语句只能判断一种条件关系，而if语句可以判断多种条件关系。

case $变量名 in

"值1"）

如果变量的值等于值1，则执行程序1

;;

"值2"）

如果变量的值等于值2，则执行程序2

;;

…省略其他分支…

\*）

如果变量的值都不是以上的值，则执行此程序

;;

esac

* **FOR循环**

第一种格式

for time in morning noon afternoon evening#空格区别各变量

do

echo "This time is $time!"

done

第二种格式

s=0

for (( i=1;i<=100;i=i+1 )) #初始值;循环控制条件;变量变化

do

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

done

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

* **while循环与until循环**

i=1

s=0

while [ $i -le 100 ]

#如果变量 i 的值小于等于 100 ，则执行循环

do#类似括号

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

i=$(( $i+1 ))

done#类似括号

echo "The sum is: $s"

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

i=1

s=0

until [ $i -gt 100 ]

#循环直到变量 i 的值大于 100 ，就停止循环

do

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

i=$(( $i+1 ))

done

echo "The sum is: $s"

* **安装虚拟机**

安装图形界面最小内存628M。

自定义安装和标准安装区别不打。标准安装在装好后也可以调整虚拟机各项硬件指标。

**Linux的硬件都是以文件形式表示的，叫硬件文件名。一般放在/dev/文件夹下面**

比如stata硬盘的的文件/dev/sd[a-p],注意是一块物理硬盘而非分区。cdrom 光驱的硬件文件名为/dev/cdrom

**硬盘分区**

硬盘最多只有4个主分区，如果不够用则可以把一个主分区改成扩展分区，扩展分区不能直接写入数据。下面只能放逻辑分区。逻辑分区可以多个。主分区是sda[1-4],逻辑分区是sda5开始。即使前面主分区没用满编号。逻辑分区也必须从5开始。扩展分区不一定是4，排在主分区后面该多少是多少。分区的的大小是挂载文件的时候指定的。分区硬件文件名是自动的。用户能控制的是依次挂载几个文件目录。

**挂载点是系统内真正的文件访问地址。**

硬件分区必须有挂载点才可以访问文件。一般来说sda[1-3]需要挂/，/swap，/boot(一般200m)。/swap分区时交换内存，是指内存不够用的时候用硬盘来当内存。一般为物理内存的两倍。不超过2G。挂载文件的时候就要设定文件格式和分区大小。

**分区完成之后对分区进行格式化之后才能装文件。**

一般用ext4文件格式进行格式化，ext4的意思是把磁盘的文件分成每个4kb的文件块。每个数据块属于哪个文件会有一个索引。文件的索引称之为inode，也交I节点号。

**自定义安装**

选择中文安装，美国英语键盘。这样可以把中文字符装上。如果选择英文安装，以后要补安装中文字符。更改主机名为repo作为克隆hadoop集群的主机名。

选择自定义磁盘布局-标准分区

先分配boot分区，200m，然后swap分区，剩下的分到根分区/,如果服务器是文件服务器则单独分一个/home分区。

软件包的选择一般用最小化或者基本服务器。如果是英文安装的至少要选择中文支持的包。

700m为图形安装界面最小内存

组建集群可以用更NAT模式单独组网。分配IP更干净。无干扰。原型机为100，集群机器从101开始。Win主机ip地址为1，网关和dns地址为2.实际上用的win的实际ip作为出口。

克隆集群机器最好用ifcfg-eth0文件中的uuid和mac地址删掉。同时删掉/etc/udev/rules.d/70-persistent-net.rules文件。防止产生两个uuid和mac地址发生冲突。这个文件删除后会计算一个新的uuid和mac地址。

DNS服务器ip保存再 resolv.conf文件中。但是再ifcfg-eth0中也可以指定。Peerdns = yes，dns1=；dns2= ； 然后系统会写入resolv.conf中。

**关闭防火墙命令**

关闭命令： service iptables stop

永久关闭防火墙：chkconfig iptables off

两个命令同时运行，运行完成后查看防火墙关闭状态

service iptables status

1 关闭防火墙-----service iptables stop

2 启动防火墙-----service iptables start

3 重启防火墙-----service iptables restart

4 查看防火墙状态--service iptables status

5 永久关闭防火墙--chkconfig iptables off

6 永久关闭后启用--chkconfig iptables on

vi /etc/selinux/config # 把 SELINUX 的值改为 disabled

**大数据软件**