第一章 LINUX系统基础

2022年9月14日 13:45

周炳烨

linux系统概述

GNU源代码开放计划
GNU is Not Unix
GPL通用公共许可协议
（权利：自由，义务：传染性）

linux 查看内核版本
uname -r

系统分层：
硬件交互层
内核层（进程管理子系统、文件子系统）
操作系统接口层（shell命令解释器、窗口系统、系统调用）
应用层

特点：
模块化，可移植性和灵活性好、稳定安全无需杀毒软件、分时多用户

应用：
互联网（集群服务器、镜像服务器）、大数据，云计算，人工智能（科学运算速度快）、物联网、嵌入式（小而精）

第二章 Linux管理基础

2022年9月20日 21:21

周炳烨

01 系统进入与退出

（1）本地登录

出现系统提示符 ” login:" 和 “ passwd:", 成功进入系统后出现提示符 ” $" ， root用户出现系统提示符 "#"

linux预设6个terminal终端控制台，分别命名为tty1~tty6,切换方式为alt+ctrl+F1至F6,图形终端默认在tty1

tty	查看当前控制台信息
uname -r	Linux内核版本及硬件等级
vim /etc/motd	用vi编辑器添加登录后提示信息

用户名@主机名 + 提示符，root用户和普通用户提示符分别为#和$。 ~ 表示用户家目录，root用户就是 / root, 其他用户就是 / home / 用户名

（2）远程登录

命令行直接登录:

Ifconfig	查看Linux系统的ip地址
ssh username@主机IP地址	在windonws终端上远程登录Linux主机

（3）切换用户

su -	切换到root用户
su zhangsan	切换到其他用户
su -zhangsan	切换到其他用户，家目录等环境信息也相应切换

（4）查看登录信息

root	查看各tty终端上登录的用户信息
whoami	查看当前用户名
who am i	本终端所有登录用户信息
last	曾经登录过的用户
lastlog	所有用户登录清况（包括即使未登录过的）

管理员UID为0，系统用户UID为1 ~999，普通用户的UID默认是从1000开始的

cat /etc/passwd	查看系统的所有用户信息
cat / etc/shadow	查看密码信息
cat /etc/group	查看所有组信息

（5）退出登录

Exit	退出这次shell进程（只能exit退回上一层）
Ctrl-D键	退出这次shell进程
logout	注销这次使用环境（文本环境下使用，不能切换账号）

基本用户组就像是原生家庭， 是在创建账号（出生）时就自动生成的；而扩展用户组则像工作单位，是需要手动添加的

（6）用户管理

id 用户名	显示用户的详细信息
useradd zhangsan	创建新的用户账户
useradd -e 2022-12-31 zhangsanl	创建新的用户账户以及账户的失效时间
useradd -G zhangsan zhangsan2	创建新的用户账户并加入扩展用户组
useradd -g zhangsan zhangsan2	创建新的用户账户并加入基本用户组
vim /etc/passwd或passwd zhangsan	修改用户密码
echo "123456" |passwd --stdin	通过格式化输入修改/设置密码
userdel -rf zhangsan 1	删除用户（r：删除用户主目录及其下文件、f:强制删除已登录用户）
usermod -G root zhangsan	变更扩展用户组至root
usermod -aG root zhangsan	增加zhangsan用户的名为root的扩展组
gpasswd -d zhangsan root	将zhangsan从root组中删除
gpasswd -a zhangsan root	将zhangsan添加到root组中

02执行命令的必备知识

（1）查询帮助信息

man passwd	查看命令的帮助信息（输入q退出】
man 5 passwd	查看配置文件(第五种类型)的帮助信息
passwd --help	简明的帮助
whatis	极简的命令作用
help exit	查看shell内置命令
type	查看命令是不是

（2）bash提供的小技巧

CTRL+C	终止当前命令
CTRL+L	清屏
SHIFT+PGUP/PGDN	上下翻查文本内容
tab	自动补齐

（3）查看并使用历史命令

history	显示历史输入的1000条命令
!5	执行历史中第五条命令
!!	执行上一条命令

（4）查看存放的位置目录

which useradd	查看命令的文件存放的目录
where is passwd	奎看该文件所在的目录

03常用工作命令

（1）echo显示

echo $...	显示字符串或变量提取后的值
echo $PATH	显示环境变量
echo $HOME	显示家目录
echo $SHELL	显示linux系统shell脚本

（2）日期时间（date、cal）

date	显示系统时间与日期
date +"%Y/%m/%d-%H:%M"	按想要的格式输出年月日时间
date +%H:%M	按想要的格式输出时间
date +%a	输出星期几
date -d "2023/1/1 12:00"	显示指定时间
date -s "2023/1/1 12:00"	重新设置系统时间
cal 2022	打印该年的日历
cal 01 2023	打印给出年、月的日历

（3）bc计算器

bc	进入计算器可进行+ - * /
scale=4	可以保留四位小数形式输出
Crtl+D或者quit	退出

（4）关机重启（reboot、poweroff、shutdown）

poweroff	立即关机
reboot	立即重启
Shutdown	1分钟以后会关机且所有用户都可看见该信息
shutdown -c	取消关机
shutdown 12:00	定时关机
shutdown -5	指定5分钟后关机
shutdown now	马上关机

04查看文件命令

（1）文件类型

标准文件【-】	目录文件【d】	链接文件【l】
特殊文件【b块文件c字符文件】	管道文件【p】	socket网络通信文件【s】

（2）查找文件

①pwd看家目录

pwd	查看当前用户的家目录

②cd进入目录

cd /etc	进入用户的根目录中的etc文件
cd ..	返回上一级目录
cd .	回到当前目录
cd	进入家目录（/home/zb）

③ls查看文件情况

ll 文件名	显示文件目录
ls -l 文件名	查找文件（显示结果：文件类型+3个wxr所有者、所属主、其他人读、写、执行的权限）
ls -al	查看所有文件，包括隐藏文件（文件名前面有一个.）
ls -il	查看文件inode号
ls -tl	按创建时间排序显示
ls -1 |wc	通过管道显示文件行数、字数、字节
ll -d	看目录本身

④tree查看树形结构

yum install tree	自动安装tree安装包
tree	查看目录树形结构
tree / -L 1或2…	查看根目录1/2/…层的树形结构
tree /	查看根目录的树形结构

⑤查找文件

find

find /etc -name …	在根目录的etc（不写etc就在全部根目录中查找）之中中查找）文件名为…的文件
find /etc -size+50k-a-size-60k	查找文件大小在50k-60k的文件
find /etc-user root	根据文件所有者查找

locate

locate+文件名	更快速查找已经更新的文件（可以找到文件在哪个目录）
updatedb+文件名	查用户自己创建的文件时需要先更新

which

Which+文件名	快速只能查命令文件（内嵌命令不能查找）

whereis

whereis	快速查找所有文件（只搜索几个特定目录）

（3）查看文件内容

①cat显示文件内容、向文件添加内容

cat -n /etc /passwd	显示加上行号的文件内容
cat >./文件名	在当前目录下添加文件，然后输入内容，ctrl+D结束并保存
cat >>./文件名	向文件添加内容
cat /etc/passwd 文件名 > 文件名	合并多个文件到新文件

②more less逐行搜索

more /etc/passwd	逐行搜寻文件
more /etc/passwd +10	从第10行开始显示
cat -n /etc/passwd |more +10	加上行号并从第10行开始显示
cat -n /etc/passwd |more -5	加上行号并仅显示5行
（然后）/关键字	可以搜索关键字 N上一个 n下一个
less	可前后翻页的逐行搜索
q	退出

③head tail看文本前/后n行

head -20 /etc/passwd	查看纯文本前n行
tail -20 /etc/passwd	查看纯文本后n行
cat -n /etc/passwd |head -20 |tail	显示文件带行号的第11到第20行
tail -20 /etc/passwd	查看文本第20行开始到最后一行

05 文件目录管理命令

/bin

/dev
/etc
/home
/usr
/var

绝对路径	（加了斜杠是在根目录）
相对路径	（相对于目前路径）

①touch新建文件

touch ./test/test1	修改文件时间/若不存在则创建该文件并设置时间

②mkdir新建目录

mkdir -p /test/test1	在根目录创建空白的目录
mkdir -p test/test1	在当前目录创建空白的目录

③cp复制拷贝文件
(/在根目录（绝对） ./在当前目录（相对））

cp /etc/test ./	复制文件到当前目录（不新建文件，不改变文件名字）
cp -p test test1	复制test文件到test1文件（保留原始时间属性）
cp ./test/* ./test2	复制test目录下所有文件到test2目录
cp -r test1 test2	递归复制目录且文件与子目录一并处理

拷贝结果与目录下是否已经有此文件有关

④mv剪切、重命名

mv test1 test2	剪切或重命名文件

⑤rm彻底删除

rm	文件名 删除文件或目录
rm -rf	确认直接删除文件与其下所有子目录

⑥rmdir删除空白文件目录

rmdir	文件名 只能删空目录及文件

⑦file查看文件类型

file test1/test2	查看文件的类型
区别：type	查看命令类型

⑧ln链接到

ln test1 test2	创建—个符号软链接文件（同一个inode）
ln -s test1 test2	创建—个符号硬链接文件（inode相同）

06 编辑器

vim

①一般指令模式

G	将光标移至最后一行
5G	将光标移至第5行
$	移到行尾
0	移到行尾
5dd	删除（剪切）从光标处开始的5行
6yy	复制从光标处开始的6行
p	将之前删除(dd)或复制(yy)过的数据粘贴到光标后面
CTRL+V然后y然后p	选择区块然后复制然后粘贴
u	撤销上—步的操作
CTRL+G	显示文件名、光标所在行与列号

②输入编辑模式

进入方法

i	光标前修改文本
a	光标后修改文本
A	在行末插入文本
I	在行首插入文本

o	光标下插入新行
O	光标上插入新行
r	修改光标处一个字符
R	一直处于替换一个字符的状态直到退出
x	删除光标所在位置字符

光标移动

上:k	nk:向上移动n行 9999k或gg可以移到第一行 G移到最后一行
下:j	nj:向下移动n行
左:h	nh:向左移动n列
右:l	nl:向右移动n列

③指令行模式（：进入）

退出

q!	强制退出（前面加个w才有保存）
wq	保存并退出

关于行

5	跳转到第5行
set nu	显示行号
set nonu	不显示行号

替换

s/one/two	将当前光标所在行的第一个one替换成two
s/one/two/g	将当前光标所在行的所有one替换成two
%s/one/two/g	将全文的所有one替换成two
4,10s/one/two/gc	4至10行的所有one替换成two, 逐一确认
4,$s/one/two/g	4至最后一行的所有one替换成two

查找

/ one（? one）	在全文中从上至下（? one是从下至上）搜索one
/^zby	查找行首是zby的行

注释

4,10s/^/#/g	注释掉第4-10行
%s/^#//g	去掉注释

执行命令

! 命令	执行该shell命令
! !	执行上次执行的命令

插入与合并

r test2	加入另一个文件（二合一）
r! date（或另外一个文件）	插入…到光标所在之处

④设置vi工作环境

~ / . viminfo	查看各用户vi的历史操作
vim ~ / . vimrc	vim环境参数

07 文件目录权限

①用户类型

u	文件所有者
g	文件所属组成员
o	其他用户

②权限类型

权限	对文件	对目录
r（读）	查看文件内容	列出目录中内容
w（写）	修改文件内容	在目录中删除、创建文件
x（执行）	可以执行文件	可以进入目录

具体表现为

权限（数字代码）	对文件	对目录

r（4）	cat、more、head、 tail	ls
w（2）	vi	touch、rm、mkdir
x（1）	./script	cd

注：要能够对文件进行操作，需先对目录有x的权限

③修改文件权限

（1）chown修改所有者

chown zby test1	将zby文件/目录的所有者修改为zby
chown -R zby test1	递 归修改其（对目录使用）下所有文件和目录
chown -R zby:zby test1	同时修改文件的所有者和所有组

（2）chgrp修改所属组

chgrp wuqiong test2	将test2文件/目录的所属组修改为wuqiong
chgrp -R wuqiong test2	递归修改具下所们 文件和目录

（3）chmod修改访问权限

chmod u+wx,g-r,o=x test	修改test文件或目录的u,g,o用户访问权限
chmod a=wrx test1	修改test1文件或目录的所有用户访问权限为wrx
chmod 777 test	修改test文件或目录的u,g,o用户访问权限均为wrx
chmod -R 664 passwd	对passwd这个目录,进行递归修改

08 文本文件操作命令

①comm比相同

comm test1 test2	（需先排序）比较两个已排序的文件是否有相同行（相同在右，相异在左）

②diff比不同

diff test1 test2	此较两个文件内容的差异/比较目录下是否存在该文件/比较两个目录下文件的差异（<第一个文件 >第二个文件 a第一个文件缺行 d第一个文件行数多了需要删掉）

③cmp按字节比较

cmp -l test1 test2	按字节比较两文件内容
cmp -l test1 test2	按字节比较并显示不同处的内容

④sort排序

sort /etc/passwd |more	分屏显示排序后的结果
sort -c test1	检查是否有序，若无序则显示第一个无序行
sort -m test1 test2 >test	将两个排好序的文件合并成一个大文件
sort -k 2 test1	按第2个关键字进行排序
sort -b -k 4.7,4.8 -k 4.1,4.2 test1	按第四个字段的第7个（若相同再以第8、1、2位排序）
参数 -n	按数字大小排序（按字典顺序）
参数 -u	排序时删去重复的内容cut

⑤cut提取

cut -d: -f1 /etc/passwd	按:作为分隔符提取第一字段内容
cut -d: -f3 /etc/passwd |sort -n	按照数字值排序

⑥grep查找

grep '12345' ./test1 ./test2	在test1文件和test2文件中查找12345
grep "123 45" test1	(搜索模式中有空格必须使用双引号)
grep '12345' ./test*.c	（通配符：?仅允许比12345多一个字符，而*没有）搜索当前目录下test开头的以.c结尾的所有文件
grep -e "123" -e "abc" ./test1?	在test1文件中查找有123和abc的字符的地方（两个-e匹配多个模式）
vi mypats 123 abc grep -if mypats ./test?	在test开头的文件名只有五个字母的文件中查找mypats文件中的模式(-i不区分大小写)（效果同-e，只是匹配模式先写在了文件里）
grep -n '12345' ./test1	显示行号
grep -v '12345' ./test1	只显示不符合匹配模式的行
grep -l '123' ./test1	只显示目录下内容含123的文件名

正则表达式

（正则表达式在“”里面表示匹配模式，只能用于文件，通配符只是在文件名后表示可以多一个或几个字符，只能用于目录）

.	匹配任意一个字符
*	重复*前面那个字符0次或无数次
^	匹配行首（在[]内使用表示）
$	匹配行尾
[ ]	集合字符
\	转义字符
'\.$'	以小数点结尾的行
'^$'	空白行
'[abc]'	含a或b或c的一个即可
'[^b]oo'	不以b在oo前面的即可（如bcdoo就可以）
'ooo*'	至少有oo即可
'b.*k'	只要一头一尾是b和k即可
'[0-9][0-9]*'	任意数字
'^[a-z]'	以小写字母开头
grep -v '[0-9]' test1	查找不含数字的行
grep '[^0-9]' test1	查找含有非数字内容的行（允许有数字）

⑦sed可修改的查找

sed -n	不会先显示一遍所有行再显示需要的内容
sed -e 'command' test1	在test1文件中查找有command的字符的地方（可在多个文件中查找）
sed -f mypats test1	在test1文件中查找满足mypats文件中的模式的地方
sed 'commands' test1 >test2	在test1文件中查找有command的字符的地方并将结果重定向到test2文件
sed -i 's/ \.$/\!/g' test1	将test1文件中结尾的 . 改为 ！
sed -i '$a # it is end' test1	在结尾行添加 # it is end

command动作

p	打印行	sed - n '5, 7p' test1 sed - n '2,5!p'不包含第2到5的行
=	显示匹配行的行号	sed -n '/1234/=' test1
a	行号后附加新文本信息	sed '2a 添加信息' test1
i	行号前插入新文本信息	sed '2i 添加信息' test1
d	删除	sed '4,$d' test1删除第四行到行尾
c	替换一/几整行	sed '2,5c 替换信息' test1
s	替换	sed 's/a/A/g'将test1中的全部a替换成A sed 's/a/A/' 将test1中的第一个a替换成A
/ /	里面可以正则表达式	sed -n '/[A-Z]/p' test1

⑧awk（可以处理字段，sed和grep只可对行操作）

last -n 5 lawk '{print $1 "\t" $2}'	显示最近登录的五次记录的第
awk '($0 ~ /abc/) {print $0}' test1	显示test1文件中含abc的所有行（~表示匹配正则表达式）（$0表示一整行）
awk -F: '($2= ='' ''){print $0}' /etc/passwd （或awk -F '{if($2=='' '') print $0)})	看是否有用户没有密码（awk -F: 或 FS=":" 表示以 : 为分隔符）
awk -F: '{print $1}' /etc/passwd	显示passwd文件中的第一字段内容(以：为分隔符，而不是空格)（或cut -d: -f1 /etc/passwd）
ifconfig |grep 'inet' |awk '($1=="inet"){print $2}'	获取ifconfig中的IP地址（剔除ipv6地址）
awk '($0!~/[a-z] . */) {print $0}' test1	显示不含小写字母的行
awk -f programme book （文件内 BEGIN {print "Price List"; print strftime()} (NR>1){sum += $2; print $1,"$"$2} END {print "Average price=$"sum/(NR-1)})	执行放在programme文件中的命令 （文件内: BEGIN{动作} （条件）{动作} （NR指当前记录数(≈行号)，从1开始，只是第一行不是价格） （条件）{动作} …… …… END{动作} ）
bash programme1	同上，但需要把所有内容放在文件中作为脚本文件

awk预定义变量

变量无须定义， 直接使用，依据变量第一次在awk语句中出现的 情况和上下文确定是数值、字符、数组，自动初始化为0或“ ”

预置变量

FILENAME	文件名	FS	输入字段分隔符(默认为空格/制表符)
NF	当前记录的字段数	NR	当前记录数（行号）
OFS	输出字段的分隔符(默认为空格/制表符)	ORS	输出记录的分隔符(默认为换行)

函数

awk 'gsub("486","586") {print $0}' test	把test中所有486换成586（或sed 's/486/586/g' test）

awk 'BEGIN {print match("ABCDXYZ", "x")}'	测试ABCDXYZ中是否含有匹配x的字符串（无则结果为0，有则结果为该字符u在字符串中的位置）
strftime	显示当前时间日期

控制语句next getline（不会返回到主体开始处，而是继续执行下一条）（成功读取到了下一行返回1，没有下一行返回0）
getline后可以加变量，表示将下一行的内容读取后储存到指针变量中，若没有，默认为$0

09系统引导、运行级、关闭

系统引导
开机进程：BIOS -> MBR (master boot record 磁盘主引导记录) 的boot loader -> 加载kernel (系统内核) -> 启动systemd进程并以default.target流程开机 (target表示文件包)

系统运行级
0 (关机) 1 (系统管理员状态) 3 (文本界面) 5 (启动图形化界面服务) 6 (重启)

who -r	查看现在的系统运行级
init 3 init 5 init 0 init 6	切换到某个运行级

系统关闭
halt（不关电源只关机）
shutdown
reboot init6

Shutdown	1分钟以后会关机且所有用户都可看见该信息
shutdown -c	取消关机
shutdown 12:00	定时关机
shutdown -5	指定5分钟后关机
shutdown now	马上关机

10系统初始化

系统初始化进程
systemd(是PID 1第一个进程)
(加d是daemon守护进程)

服务的启动

systemctl status sshd	查看ssh服务状态
将status换为start	启动……服务
restart	重新启动……服务
stop	终止……服务
reload	重新……服务的加载配置文件
disable	使……服务开机不自启

enable	使……服务开机自启
is-enabled	查看……服务是否为开机自启

11 用户及组管理

用户管理文件：

/etc/passwd

/etc/shadow

/etc/group

passwd文件行格式：login_name:password:uid:gid:user info(用户注释信息):home_directory:default_shell(登录时默认执行的shell)

cat /etc/shells	查看可使用的shell有哪些
echo $SHELL	查看默认的shell版本（一般为bash）

group文件行格式：groupname:password:gid:user-list（同组的用户清单）

用户管理命令

id 用户名	显示用户的详细信息
useradd zhangsan	创建新的用户账户
useradd -e 2022-12-31 zhangsanl	创建新的用户账户以及账户的失效时间
useradd -G zhangsan zhangsan2	创建新的用户账户并加入扩展用户组
useradd -g zhangsan zhangsan2	创建新的用户账户并加入基本用户组
vim /etc/passwd或passwd zhangsan	修改用户密码
userdel -rf zhangsan 1	删除用户（r：删除用户主目录及其下文件、f:强制删除已登录用户）
usermod -G root zhangsan	变更扩展用户组至root
usermod -aG root zhangsan	增加zhangsan用户的名为root的扩展组
gpasswd -d zhangsan root	将zhangsan从root组中删除
gpasswd -a zhangsan root	将zhangsan添加到root组中

组管理命令

groupadd testgroup	添加名为testgrouop的组
usermod -aG testgroup zhangsan gpasswd -a zhangsan testgroup	将zhangsan的扩展组加入testgroup中
groupmod -g 1005 testgroup	将testgroup的组序号修改为1005
groups zby	查看zby用户所属组信息
groupdel testgroup	删除组
gpasswd -d zhangsan testgroup	将zhangsan从testgroup组中删除
gpasswd -A zhangsan testgroup	将zhangsan设为组管理员

管理员与用户通信

write zby + CTRL-D (表示结束)	发消息给zby
write zby < filetest	把filetest文件的内容发给zby用户 【注：重定向符 >>添加 >直接把原文件覆盖掉 < 】
wall + CTRL-D	发消息给所有用户
mail xxc	发邮件给xxc
mail 然后输入序号	查看第几封邮件

mail d1	删除第一封邮件
mail -u xxc	（作为root）查看1某用户的邮件

计划任务服务程序

周期性任务：cron
一次性任务：指定时间：at
自动在系统低负荷时做：batch

cron

系统守护进程crond每一分钟被唤醒一次并扫描一次crontab文件

系统crontab文件	/etc/crontab
用户crontab文件	/var/spool/cron/username
cron进程执行的记录日志	/var/log/cron

crontab文件

若有/etc/cron.allow，则
列出的用户可以执行自己的crontab文件
若无/etc/cron.allow，则
/etc/cron.deny文件不为空， 不在该文件中的用户可以执行自己的crontab文件
/etc/cron.deny文件内容为空，则表示所有用户都可以使用cron
若/etc/cron.allow和/etc/cron.deny文件都不存在，则
所有普通用户都不能执行自己的crontab文件， 只有超级用户才有权执行系统的crontab文件
(系统默认只有/etc/cron.deny)

crontab文件的参数格式

普通用户	分钟 小时 日期 月份 星期 命令
root用户	分钟 小时 日期 月份 星期 用户名 命令

字段内符号

*	所有可能的值
1,5	1和5
1-5	从1到5的整数范围
/	指定时间的间隔频率（如在分钟字段*/5：每五分钟执行一次）

（修改后执行任务时需要reload有的系统甚至要restart）

crontab文件修改方式

root用户	crontab -u zhangsan -e(给用户创建任务) crontab -e(给root自己创建任务)
普通用户	crontab -e

注：执行结果通过mail保存到 /var/spool/mail/用户名
-e参数始终是用户的crontab文件，系统的文件只能用vim编辑器

at

进入at模式：

at now +5 months
at tomorrow
at 0309am July 30
at Sunday
at 23:00 2022-09-01 +3 days

进入at模式后的任务编辑：

at> /bin/date	（使用命令需要用其绝对路径）
at> /usr/sbin/shutdown now	
at> ctl+d	
at now +1 minutes <(或-f) /home/zby/job1	若需要执行的任务较多，可以先写在文件里（要用文件的绝对路径位置）

at> cat /dev/zero > /dev/null &	（加上&可以让指令在后台执行）

对已经存在的作业进行处理：

at -l	查看有哪些计划作业
at -c job1	查看作业内容
at -d job1	删除作业

对at的访问权限：

/etc/at.allow 和 /etc/at.deny文件，同crontab

batch

同at一样，只是自动在系统负荷低的时候进行作业

系统性能管理

ps

显示正在进行的进程信息

ps	仅显示当前shell下的进程
ps -e	显示所有shell解释器下进程
ps -aux	显示所有进程（含和终端无关的）

然后可以kill 2927（PID进程号）杀死该进程

top

实现对处理器实时状态监控

h	帮助
k 2034	杀死第2034号进程
s	改变系统刷新的延迟时间

jobs

查看当前终端放入后台的工作

周炳烨

第三章 SHELL

2022年11月15日　　19:43

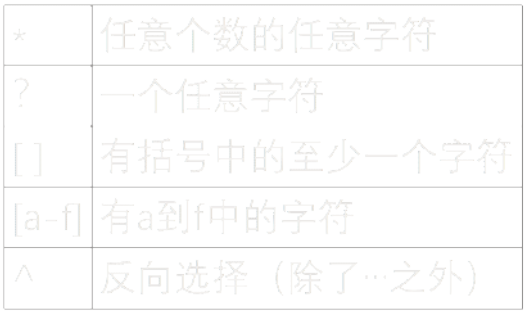
周炳烨

SHELL脚本概述

1 查看shell种类

cat /etc/shells
echo $SHELL

2 Bash的功能

history	显示1000条历史命令
!2	执行第二条命令
look="ls -al"	设置别名（等号两端不能有空格） 注:　（管理员）放在/etc/bashrc （用户）放在~/.bashrc（有.是代表隐藏文件） （重启后或运行该文件后即可生效）
可以使用通配符	 如ls -d /etc/cron* ls- al /dev/[a-zA-Z]*[0-9]

3 切换shell

chsh -s zsh	
vim /etc/passwd	修改zby:x:1000:1000:ZBY:/home/zby:/bin/bash 或修改root:x:0:0:root:/root:/bin/bash

4 编写shell脚本

脚本中的命令应当用绝对路径　如查看当前工作目录/bin/pwd

5 执行shell脚本

sh test或bash test	对文件只需要r不需要x权限，但是对文件所在目录需要有x权限（只要是在当前目录下就不需要写路径）	启动了新的shell
chomd u+x test然后直接输入文件名test	此方法先要获取文件的执行权限，运行文件时需输入路径，如./	启动了新的shell
source ./test或. ./test	.之后要跟一个空格	在原shell下执行

SHELL脚本变量

1 shell变量类型

shell变量	环境变量（全局变量） 系统环境变量：位于/etc/profile 用户环境变量：位于~/.bash_profile 临时环境变量：未写入配置文件（仅在当前shell及其子shell中才有效） 本地变量（仅在当前shell中才有效） 系统特殊变量（不能修改）

	用户自定义变量

注：echo显示变量时，需在变量前加$
export时，变量前也需要加$
~/.bash_profile文件内容格式为 name=zby
export $name
(等号两边不能有空格,否则是判等不是赋值)
$$可以查看当前shell脚本的进程号

2 shell下输入输出命令

read -p （显示提示内容的）输入	read -p "input four varys:" var1 var2 var3 var4（输入值太多多余的会全部赋给最后一个变量，输入值太少后面的变量会被赋空值）
echo 输出 echo -e转义字符有效（\c抑制后面的输出\\显示反斜线\n换行输出） echo -n不输出行尾的换行符	（1）输出有多个空格时需要用引号括起来，否则被认为只有一个 （2）echo可以匹配通配符（尤其是通过管道关联到cat、grep等命令时）（*任意字符1？一个字符[0-9]数字） （3）

3 系统环境变量
主要系统环境变量

PATH	shell所要查找的路径，以：分隔（每次输入命令，shell就可以自动从这些路径中查找，因此不用输绝对路径就能执行命令）
HOME	用户的家目录（根目录）
PS1	prompt string系统提示符 \u 当前用户名 \h 主机名的前一部分 \W 目录中的最后一级 \$ 提示字符（root# 其他$） \d 星期月日 \t 显示24小时制时间
PS2	系统的次提示符，默认为>
LANG	显示当前语言编码(可在/etc/locale.conf中修改)

环境变量查看
查看环境变量的命令

env	显示所有环境变量	查找某个变量 env |grep PATH
export	显示所有环境变量	查找某个变量 export |grep PATH
set	显示1所有环境变量和本地变量 set -u 使用未定义的变量时提示错误 set +u 关闭提示功能 set -x （用在脚本的首行）可显示命令执行情况，方便排错	查找某个变量 set |grep PATH

环境变量的主要配置文件

/etc/profile
~/.bash_profile
~/.bashrc

4 系统特殊变量
（1）位置参数变量

$0 $1 $2 $3 $4 $5 $6 $7 $8 $9	$0存放命令名，其他1存放命令参数 在echo后添加shift命令，实现移位（从右到左前移一位，但$0始终不动） 移位后原始数据将不被保留，只剩下移位后参数数据
$#	向脚本提供的位置参数个数
$*	位置参数字符串（所有位置参数组成的大的字符串）
$@	位置参数字符串（每个位置参数都分别是一个字符串）

（2）其他特殊变量

$$	当前shell脚本的进程号
$!	上一个后台命令的进程号
$?	上一条指令的回传值（成功为0 失败为127）

5 用户自定义变量
定义与使用格式：（1）定义等式中等号两端不能有空格
（2）变量的值有空格时需要将它放在“”之内
（3）echo时，如前变量后字符($***myname***hahaha)
表示方法：

${myname}hahaha
"$myname"hahaha

测试命令

myname=${name:-zby}	如果name未被定义或为空，则将“zby”赋给myname
echo "Everything was in a state of ${state:-discirder}"	可以在字符串内使用

定义命令变量
命令放在` `或$( )中
例如，echo "Now it is `pwd`"
echo "Now is $(pwd)"
time=`date`，echo "Now is $time"
注：“ ”中遇到$或' '会做变量替换，遇到` `或$( )会做命令替换
' '中完全为普通字符，不做命令替换
对变量添加内容时，myname="$myname":added_continent
若希望该变量在其他子程序中执行，export myname变成环境变量
6 算数运算
（1）expr语句
expr 24 / 6
expr 24 * 2
value=`expr 5 + 3`
注：只能处理整数，运算符前后要有空格，乘法要用/*
（2）$(( ))
echo $((24*2/4))
echo $(($num1*$num2))
value=$((24*2/4))
注：只能处理整数
（3）bc
echo 1.25*6.25 |bc
value=`echo (3.33+1.25)*6.25 |bc`
（4）declare -i
（declare宣告变量类型-a数组-i整数-x变为环境变量）
declare -i sum=100+30*50

SHELL控制结构

（一） if - then条件语句
（1）基本格式：
if test 条件（或if [条件]）
then 执行的命令
fi
嵌套的if
if test 条件（或if [条件]）
then 执行的命令
else if 条件（或elif 条件）
then 执行的命令
else 执行的命令
fi
fi
【注：if后可以是命令，不是看命令结果，而是是否成功执行，如 if grep -q "root" /etc/passwd，找到了返回值就为1】
（2）测试条件
（单独执行test时，可以通过echo $?查看结果）
[0判断为真 127为假]
数字类
test 5 -ne 8
-eq 判等
-ne 判不等
-lt 小于
-gt大于
-le 小于等于
-ge 大于等于
文件类
test -f /etc/passwd
-d 判断1是否存在且是目录
-e 判断存在
-f 判断存在且是文件

字符类
== 判等（或(空格)=(空格)）
!= 判不等
-z string 判断字符串长度为0
-n 判断字符串长度为非0
逻辑类
!expression 取非
expression1 -a expression2 与
expression1 -o expression2 或
[cmd1 && cmd2：cmd1正确($?=0)执行则继续执行cmd2，cmd1执行结果错误($?=127)就不执行cmd2]
[cmd1 || cmd2：cmd1正确则不执行cmd2，cmd1错误才执行cmd2]

（二）case选择语句
case variable in
mode1）command1;;
mode2）command2;;
*）command;;
esac
注：mode1,mode2均可匹配正则表达式，也可以是简单的a,b,c,1,2,3
在命令结尾处需加 ;;
（三）for循环语句
for variable in list
do
commands
done
注：list列表可以是a b c（等价于letter="a b c"，$letter）,也可以是"a b c"【前者被认为是多个字符串，后者被认为是一个】
list列表可以
list列表可以是命令结果，如`ls`
list列表可以是存放在一个变量中的cut / cat命令结果
list列表可以是文件（可以匹配正则表达式），如test*（以test开头的所有文件）
list列表可以是位置参数，$1、$*、$@等，需要在执行是给予参数
for可以多重循环

（三）while语句

有终止条件的循环
while test 条件（或while 命令或while [条件]）
do
commands
done
例如，while read variable，若输入EOF则read失败，停止循环，可以用sleep命令（休息几秒）实现无限占用

无终止条件的循环
while:
do
commands
sleep 1
done

（四）until语句
until test 条件（或until 命令或until [条件]）
do
commands
done
条件满足时就停止，而while是条件满足时就继续

（五）break语句与continue语句

SHELL函数

function_name()
{
commands
}
函数没有值的传递，可以直接用外部变量

2022-12-21