第二十四章：Linux信号及Expect自动化交互式程序

**一、Linux信号的概述；**

**二、使用trap控制信号；**

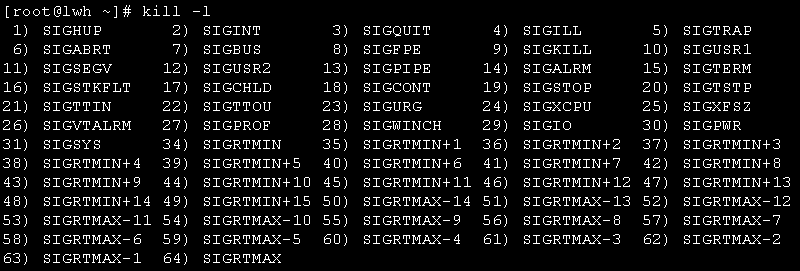
**三、Expect自动化交互式程序；**

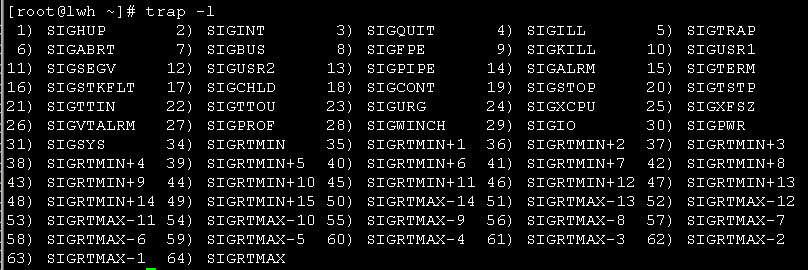
**一、Linux信号的概述：**

**概述：**运行shell脚本时，ctrl c或者ctrl z等快捷键可以使程序终止，这种终止就是采用快捷键调用信号进行异步操作来完成的；

**场景：**但是我们在有些情况下，并不想使脚本被信号所影响，此时就需要屏蔽信号的手段，让shell解释器忽略信号的指令，继续运行shell脚本程序；

**信号列表：**使用信号时去掉信号名前的SIG；





**常见的信号对应关系：**



**二、使用trap控制信号：**

**概述：**trap命令用于设置系统在接收到信号后所采取的行动；

**语法：**trap 'command;command' 信号编号或信号名称

**案例：**

**1.设置ctrl c快捷键的提示；**

[root@lwh ~]# trap "echo exit" 2

[root@lwh ~]# ^Cexit

[root@lwh ~]# trap "echo exit" INT

[root@lwh ~]# ^Cexit

**2.取消ctrl c快捷键的退出作用；**

[root@lwh ~]# stty -a ##查看键盘与信号的对应关系

speed 38400 baud; rows 29; columns 80; line = 0;

intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;

eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R;

werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0;

-parenb -parodd cs8 -hupcl -cstopb cread -clocal -crtscts -cdtrdsr

-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff

-iuclc -ixany -imaxbel -iutf8

opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0

isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt

echoctl echoke

[root@lwh ~]# ^C

[root@lwh ~]# trap "" 2 ##取消ctrl c的功能

[root@lwh ~]# ##此时按下ctrl c则无反应

[root@lwh ~]# trap ":" 2 ##恢复ctrl c的功能

[root@lwh ~]# ^C

**3.屏蔽多个信号和所有信号；**

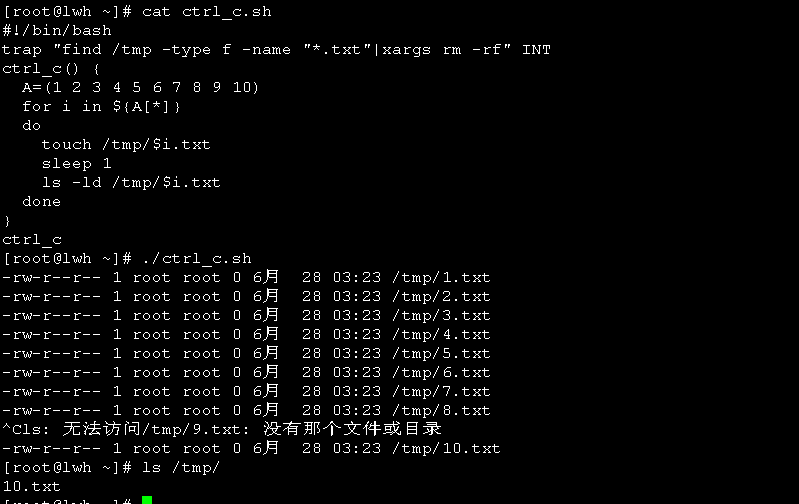
[root@lwh ~]# trap "" 1 2 3 20 15

[root@lwh ~]# trap ":" 1 2 3 20 15

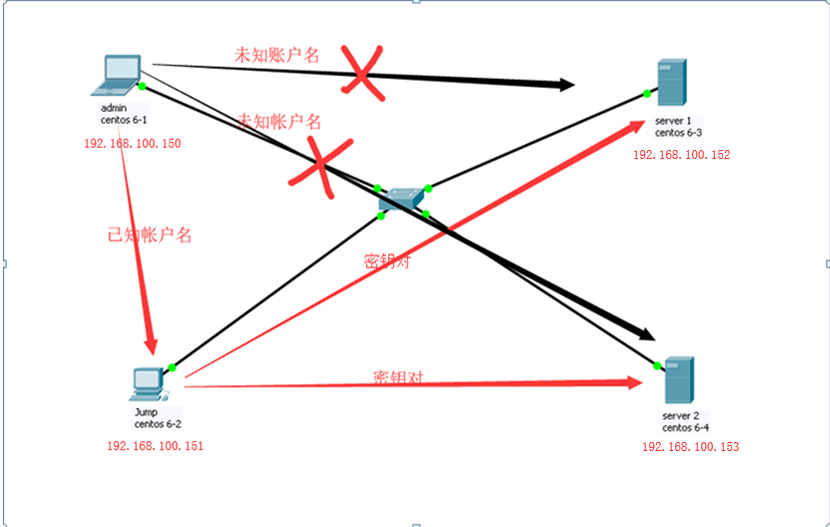
[root@lwh ~]# trap "" HUP INT QUIT

[root@lwh ~]# trap "" {1..64}

**4.更改ctrl c的控制；**



**5.企业级案例，跳板机的应用；**



**解析：**admin主机作为管理员主机，未知server1和server2的帐户名和密码，若实现登录必须通过jump跳板机实现，跳板机与server1和server2之间验证方式为密钥对方式。并且admin主机访问jump跳板机时只能执行特定的脚本，无任何权限执行跳板机的命令；

[root@jump ~]# useradd jump

[root@jump ~]# echo "123123" |passwd --stdin jump

[root@jump ~]# su - jump

[jump@jump ~]$ ssh-keygen -t rsa

[jump@jump ~]$ ssh-copy-id root@192.168.100.152

[jump@jump ~]$ ssh-copy-id root@192.168.100.153

**方法一：**

[jump@jump ~]$ vi begin.sh

#!/bin/bash

trap "" HUP INT QUIT

while true;do

read -p "MENU:

1).Input 1 is ssh 192.168.100.152 server1;

2).Input 2 is ssh 192.168.100.153 server2;

3).Input admin passwd is ssh 192.168.100.151 jump;

please choose a number:" NUM

if [[ $NUM -eq 1 ]];then

ssh root@192.168.100.152

elif [[ $NUM -eq 2 ]];then

ssh root@192.168.100.153

elif [[ $NUM == 9404 ]];then

exit &>/dev/null

else

echo "please you again!"

sleep 2

fi

done

[jump@jump ~]$ echo "/home/jump/begin.sh" >>/home/jump/.bash\_profile

**方法二：**

[jump@lwh ~]$ cat admin.sh

#!/bin/bash

trap "" 1 2 3 20 15

while true ;do

read -p "MENU:

1).Input 1 is ssh 192.168.100.152 server1;

2).Input 2 is ssh 192.168.100.153 server2;

3).Input admin passwd is ssh 192.168.100.151 jump;

please choose a number:" NUM

case $NUM in

1)

ssh root@192.168.100.152

;;

2)

ssh root@192.168.100.153

;;

9404)

exit 0

;;

\*)

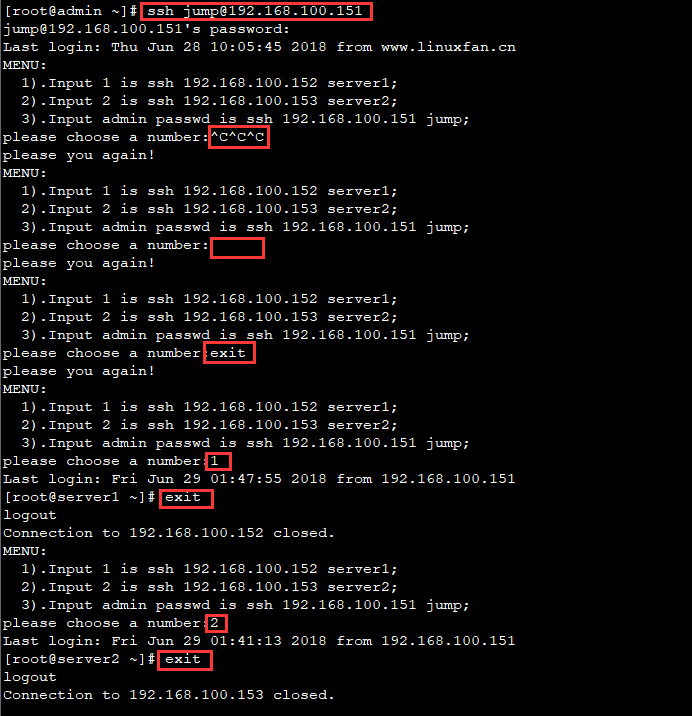
echo "please you again!"

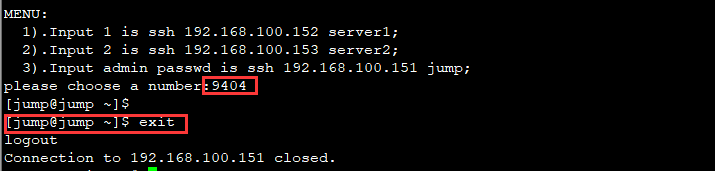
;;

esac

done

**admin管理主机测试：**

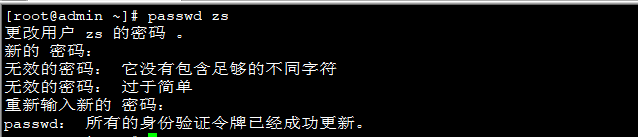


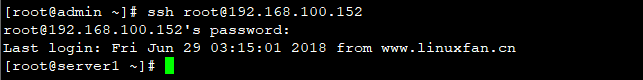


**三、Expect自动化交互式程序；**

**概述：**Expect是一个用来实现自动交互功能的软件套件，功能强大；

**使用场景：**在管理系统的多种情况下，会出现管理员手动输入字符，称为交互式，如下：





**工作流程：**spawn启动指定进程-->expect获取期待的关键字-->send向指定进程发送指定字符-->进程执行完毕，退出结束（严格遵循以上流程）

**命令：**



$argv 相当于$\* $argc 相当于$#

**安装：**





**案例一：**两台虚拟机，实现expect自动交互功能ssh远程登录（无yes验证场景）；

|  |  |
| --- | --- |
| IP地址 | 主机名 |
| 192.168.100.150 | ssh1 |
| 192.168.100.151 | ssh2 |

[root@ssh1 ~]# vi ssh.sh

#!/usr/bin/expect

set ip [lindex $argv 0]

##将第一个位置变量交给变量ip；

set passwd [lindex $argv 1]

##指定命令

spawn ssh root@$ip

##指定匹配字段

expect "\*password"

##指定交互式内容\n代表换行

send "$passwd\n"

##开启交互式模式，不然导致远程后无法敲命令

interact

[root@ssh1 ~]# ./ssh.sh 192.168.100.151 pwd@123

spawn ssh root@192.168.100.151

root@192.168.100.151's password:

Last login: Thu Jun 28 18:04:45 2018 from www.linuxfan.cn

[root@ssh2 ~]#

**案例二：**两台虚拟机，实现expect自动交互功能ssh远程登录（有无yes验证场景均可）；

[root@ssh1 ~]# vi ssh2.sh

#!/usr/bin/expect

set ip [lindex $argv 0]

set password [lindex $argv 1]

set timeout 10

spawn ssh root@$ip

expect {

"\*yes/no" { send "yes\r"; exp\_continue}

##exp\_continue表示继续匹配字段

"\*password:" { send "$password\r" }

}

interact

[root@ssh1 ~]# ./ssh2.sh 192.168.100.151 pwd@123

spawn ssh root@192.168.100.151

root@192.168.100.151's password:

Last login: Thu Jun 28 16:54:00 2018 from www.linuxfan.cn

[root@ssh2 ~]#

**案例三：**交互式采取输入信息

[root@ssh1 ~]# cat name.sh

#!/bin/bash

read -p "please input your name:" NAME

read -p "please input your age:" AGE

read -p "please input your email:" MAIL

echo -e "Your INFO:\nNAME: $NAME\nAGE: $AGE\nMAIL: $MAIL"

[root@ssh1 ~]# ./name.sh

please input your name:liwenhu

please input your age:25

please input your email:li\_\_wenhu@163.com

Your INFO:

NAME: liwenhu

AGE: 25

MAIL: li\_\_wenhu@163.com

**案例四：**通过expect改写案例三

[root@ssh1 ~]# cat name1.sh

#!/usr/bin/expect

spawn /bin/bash /name.sh

send\_user "此脚本用于采集个人身份信息\n请如实填写：\n"

##相当于echo e

expect {

"\*name\*" {exp\_send "liwenhu\r";exp\_continue}

"\*age\*" {exp\_send "25\r";exp\_continue}

"\*mail\*" {exp\_send "li\_\_wenhu@163.com\r"}

}

expect eof

##代表expect正常结束

[root@ssh1 ~]# ./name1.sh

spawn /bin/bash /root/name.sh

此脚本用于采集个人身份信息

请如实填写：

please input your name:liwenhu

please input your age:25

please input your email:li\_\_wenhu@163.com

Your INFO:

NAME: liwenhu

AGE: 25

MAIL: li\_\_wenhu@163.com

**案例五：**变量的练习；

[root@ssh1 ~]# cat bl.sh

#!/usr/bin/expect

set A [lindex $argv 0]

set B [lindex $argv 1]

set C [lindex $argv 2]

set D haha

send\_user "\$A=$A\n"

send\_user "\$B=$B\n"

send\_user "\$C=$C\n"

send\_user "\$D=$D\n"

send\_user "\$argv=$argv\n"

##相当$\*

send\_user "\$argc=$argc\n"

##相当$#

[root@ssh1 ~]# ./bl.sh a b c d e f g

$A=a

$B=b

$C=c

$D=haha

$argv=a b c d e f g

$argc=7

**案例六：expect结合if的练习，注意：括号的空格，严格要求；**

[root@ssh1 ~]# cat if.sh

#!/usr/bin/expect

if {$argc != 3} {

send\_user "usage: expect $argv0 file host dir\n"

exit

} else {

set FILE [lindex $argv 0]././

set HOST [lindex $argv 1]

set DIR [lindex $argv 2]

puts "$FILE\t$HOST\t$DIR;" }

[root@ssh1 ~]# ./if.sh a b c d

usage: expect ./if.sh file host dir

[root@ssh1 ~]# ./if.sh a b c

a b c;

**案例七：**企业级案例--使用expect实现交互式管理执行命令；

环境如下；

|  |  |
| --- | --- |
| IP地址 | 主机名 |
| 192.168.100.150 | admin |
| 192.168.100.151 | server1 |

[root@admin ~]# cat admin.sh

#!/usr/bin/expect

set ip [lindex $argv 0]

set user [lindex $argv 1]

set password [lindex $argv 2]

set com [lindex $argv 3]

set timeout 10

spawn ssh $user@$ip $com

expect {

"\*yes/no" { send "yes\r"; exp\_continue}

"\*password:" { send "$password\r" }

}

interact

[root@admin ~]# ./admin.sh 192.168.100.151 root pwd@123 "free -m"

spawn ssh root@192.168.100.151 free -m

root@192.168.100.151's password:

total used free shared buffers cached

Mem: 482 229 253 0 32 126

-/+ buffers/cache: 70 412

Swap: 1983 0 1983

[root@admin ~]# ./admin.sh 192.168.100.152 root pwd@123 "df -hT"

spawn ssh root@192.168.100.152 df -hT

root@192.168.100.152's password:

Filesystem Type Size Used Avail Use% Mounted on

/dev/mapper/vg\_lwh-lv\_root ext4 18G 1.5G 15G 10% /

tmpfs tmpfs 242M 0 242M 0% /dev/shm

/dev/sda1 ext4 485M 34M 427M 8% /boot

**案例八：结合expect实现批量化自动下发脚本，执行脚本，完成自动化运维管理（结合案例七）**

[root@admin ~]# cat 1.sh ##准备测试脚本文件

#!/bin/bash

ping 127.0.0.1

[root@admin ~]# cat adminfile.sh ##准备远程下发脚本文件的脚本

#!/usr/bin/expect

set ip [lindex $argv 0]

set user [lindex $argv 1]

set password [lindex $argv 2]

set sfile [lindex $argv 3]

set ddir [lindex $argv 4]

set timeout 10

spawn scp $sfile $user@$ip:$ddir

expect {

"\*yes/no" { send "yes\r"; exp\_continue}

"\*password:" { send "$password\r" }

}

interact

[root@admin ~]# cat admin.sh ##准备远程执行脚本文件的脚本

#!/usr/bin/expect

set ip [lindex $argv 0]

set user [lindex $argv 1]

set password [lindex $argv 2]

set com [lindex $argv 3]

set timeout 10

spawn ssh $user@$ip $com

expect {

"\*yes/no" { send "yes\r"; exp\_continue}

"\*password:" { send "$password\r" }

}

interact

[root@admin ~]# ./adminfile.sh 192.168.100.151 root pwd@123 /root/1.sh /tmp/ ##下发脚本文件

spawn scp /root/1.sh root@192.168.100.151:/tmp/

root@192.168.100.151's password:

1.sh

[root@admin ~]# ./admin.sh 192.168.100.151 root pwd@123 "/tmp/1.sh" ##执行脚本

spawn ssh root@192.168.100.151 /tmp/1.sh

root@192.168.100.151's password:

PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.

64 bytes from 127.0.0.1: icmp\_seq=1 ttl=64 time=0.010 ms

64 bytes from 127.0.0.1: icmp\_seq=2 ttl=64 time=0.030 ms

64 bytes from 127.0.0.1: icmp\_seq=3 ttl=64 time=0.021 ms

64 bytes from 127.0.0.1: icmp\_seq=4 ttl=64 time=0.018 ms

64 bytes from 127.0.0.1: icmp\_seq=5 ttl=64 time=0.018 ms

64 bytes from 127.0.0.1: icmp\_seq=6 ttl=64 time=0.013 ms