Expect 脚本学习

2015/3/9
renyl

1 介绍

1) shell 脚本可以使用 here 文档实现某些交互，但无法完成需要用户手动去输入的交互（如 passwd、scp）。expect 由此而生，可以轻松的完成此类交互。

2) expect 是一种可以提供“分支和嵌套结构”来引导程序流程的解释型脚本语言，主要运用于自动交互，能够按照脚本内容里面设定的方式与交互式程序进行交互。

3) expect 可以根据程序的不同提示或反馈内容来进行不同的应答，并且可随时在需要的时候把控制权交给用户，然后再还给脚本。

4) expect 脚本是基于 TCL（Tool Command Language）演变的，其语法与 TCL 语法类似。

2 expect 的使用

2.1 常用命令

命令	说明
spawn	调用需要执行的命令
expect	1) 等待命令提示信息的出现，也就是捕捉用户输入的提示。 2) 只有 spawn 执行的命令结果才会被 expect 捕捉到，因为 spawn 会启动一个进程，只有这个进程的相关信息才会被捕捉到。 3) 相关信息主要包括：标准输入的提示信息，eof 和 timeout。
send	1) 发送需要交互的值，替代了用户手动输入内容。 2) 交互的值是发送给 spawn 启动的那个进程
set	设置变量值
interact	执行完成后保持交互状态，把控制权交给控制台
system	相当于在 shell 执行命令

注：
命令“expect eof”: spawn 进程结束后会向expect发送eof；表示捕获终端输出信息终止，必须去匹配。如果不去匹配的话，进行一个下spawn的话可能会出问题。

2.2 命令使用

通过一个简单的 ssh 登录实例来说明上述的命令使用，如下所示：

```
[root@localhost renyl]# cat test.exp
#!/usr/bin/expect
system pwd
set timeout 10

set IP [lrange $argv 0 0]
send_user "script_name=$argv0\n"
send_user "argc=$argc\n"
send_user "IP=$IP\n"

set para "hello world"
send_user "para=$para\n"

spawn ssh $IP
expect "password"
send "rootroot\n"

expect "#"
send "exit\n"

expect eof
system "pwd"

[root@localhost renyl]# ./test.exp  193.168.220.17
/home/renyl
script_name=./test.exp
argc=1
IP=193.168.220.17
para=hello world
spawn ssh 193.168.220.17
root@193.168.220.17's password:
Last login: Thu Aug  7 17:36:13 2014 from 193.168.235.194
[root@localhost ~]# exit
logout
Connection to 193.168.220.17 closed.
/home/renyl
[root@localhost renyl]#
```

注：

1) 使用-d 选项，如 expect -d ./test.exp 193.168.220.17 可以详细的输出执行过程中的信息。

2) 通常 expect 会在执行脚本之前，把整个脚本都读入到内存中。使用-b 选项可以让 expect 一次只读取脚本中的一行。当没有写完整个脚本的时候，这个选项是十分有用的。

对上述的脚本进行分析，如下所示：

命令	说明
system pwd	类似于在 shell 脚本中执行 pwd 命令。
set timeout 10	1） 设置超时时间，单位为秒。 2） 如果 expect 在这个时间范围内没有匹配成功,则返回。 3） 设置为-1，表示永远不会超时。
set IP [lrange $argv 0 0]	1） 设置 IP 变量为命令的第一个参数。 2） [lrange $argv 0 3]表示取第一个到第四个参 3） argv 变量不包含脚本名字
send_user "name=$argv0\n" send_user "argc=$argc\n" send_user "IP=$IP\n"	1） send_user 用于向终端输出，类似于 shell 中的 echo 2） argv0 变量保存着脚本名字 3） argc 变量保存着参数的个数
set para "hello world" send_user "para=$para\n"	1） set 命令用于设置变量 2） 取变量同 shell 意义，使用$para 来取变量值
spawn ssh $IP expect "password" send "rootroot\n"	1） 使用 spawn 命令调用要执行的命令 2） 使用 expect 命令等待提示信息的出现 3） 使用 send 命令发送需要交互的值
expect "#" send "exit\n" expect eof	1） 等待“#”好的出现 2） 退出 ssh 连接的服务器 3） 终止捕获终端输出信息
system "pwd"	再执行一次 pwd 命令

注：

1） [$para]：方括号执行了一个嵌套命令。如果想传递一个命令的结果作为另外一个命令的参数，那么使用这个符号。

2） "$para"： 双引号把词组标记为命令的一个参数，$符号和方括号在双引号内仍被解释。

3） {$para}：大括号也把词组标记为命令的一个参数，但是其他符号在大括号内不被解释。

4） "\$para"：反斜线符号是用来引用特殊符号，如：\n 代表换。反斜线符号也被用来关闭"$"符号、引号、方括号和大括号的特殊含义。

2.3 常用语法

2.3.1 if 语句

实例如下：

必须有空格

必须有空格

```
[root@localhost renyl]# cat test.exp
#!/usr/bin/expect

set  para apple

if {"$para" == "apple"} {
        send_user "para=apple\n"
} else {
        send_user "para!=apple\n"
}

if { $argc < 2 } {
        send_user "usage: $argv0 para1 para2 para3... \n"
        exit
}

[root@localhost renyl]# ./test.exp
para=apple
usage: ./test.exp para1 para2 para3...
[root@localhost renyl]#
```

2.3.2 for 语句

实例如下：

两种不同格式的 for 循环语法

```
[root@localhost renyl]# cat test.exp
#!/usr/bin/expect

foreach i { a b c } {
        puts  $i
}

for {set i 0} {$i<$argc} {incr i} {
        set para [lrange $argv $i $i]
        puts $para
}
[root@localhost renyl]# ./test.exp  1 2 3
a
b
```

```
c
1
2
3
[root@localhost renyl]#
```

2.3.3 while 语句

实例如下：

```
[root@localhost renyl]# cat test.exp
#!/usr/bin/expect

set loop 0
while { $loop < $argc } {

        set para [lrange $argv $loop $loop]
        puts $para

        #incr loop
        set loop [expr {$loop + 1}]
}
[root@localhost renyl]# ./test.exp  1 2 3
1
2
3
[root@localhost renyl]#
```

expect 里的加法

2.3.4 switch 语句

实例如下：

```
[root@localhost renyl]# cat test.exp
#!/usr/bin/expect

switch $argc {
        1 {
                puts "one parameter"
        }
        2  {
                puts  "two parameters"
        }
        3 {
                puts  "three parameters"
        }
```

不需要 break

```
        default {
                puts "Usage: $argv0 para1 para2 para3"
                exit
        }
}
[root@localhost renyl]# ./my.exp  1
one parameter
[root@localhost renyl]#
```

2.3.5 continue 语句

实例如下：

```
[root@localhost renyl]# cat test.exp
#!/usr/bin/expect

for {set i 0} {$i<$argc} {incr i} {

        if { "$i" == 1  } {
                continue
        }

        set para [lrange $argv $i $i]
        puts $para
}
[root@localhost renyl]# ./test.exp  1 2 3
1
3
[root@localhost renyl]#
```

2.3.6 break 语句

实例如下：

```
[root@localhost renyl]# cat test.exp
#!/usr/bin/expect

for {set i 0} {$i<$argc} {incr i} {

        if { "$i" == 1  } {
                break
        }

        set para [lrange $argv $i $i]
        puts $para
```

```
}
[root@localhost renyl]# ./test.exp  1 2 3
1
[root@localhost renyl]#
```

2.3.7 exp_continue 语句

实例如下：

```
[root@localhost renyl]# cat test.exp
#!/usr/bin/expect

spawn ssh root@193.168.246.254

expect {
-re "passw*" { send "linux_pt!!\n"}
"yes/no" { send "yes\n";exp_continue}
}

interact
[root@localhost renyl]# ./test.exp
spawn ssh root@193.168.246.254
The authenticity of host '193.168.246.254 (193.168.246.254)' can't be establishe
d.
RSA key fingerprint is 04:aa:79:db:b0:60:3b:59:98:05:e3:fe:38:01:e9:82.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '193.168.246.254' (RSA) to the list of known hosts.
root@193.168.246.254's password:
Last login: Fri Aug  8 18:23:52 2014 from 193.168.235.194
[root@FileServer ~]# exit
logout
Connection to 193.168.246.254 closed.
[root@localhost renyl]# ./test.exp
spawn ssh root@193.168.246.254
root@193.168.246.254's password:
Last login: Fri Aug  8 18:24:49 2014 from 193.168.235.194
[root@FileServer ~]# exit
logout
Connection to 193.168.246.254 closed.
[root@localhost renyl]#
```

注：

-re 参数表示指定的字符串是一个正则表达式，而不是一个普通的字符串。

说明：

1) expect 有如下两种用法：

expect 用法	说明
expect pattern1 command1 expect pattern2 command2	1) 只有 pattern1 匹配成功后，才能执行 command1。然后才能继续匹配 pattern2。 2) 这是个串行的匹配流程，如果 pattern1 匹配不成功，将会被阻塞。

expect {
pattern1 { command1 }
pattern2 { command2 }
}
	1) expect 会先对 pattern1 进行匹配，如果匹配成功，执行命令 command1。整个 expect 工作完成。
	2) 如果 pattern1 匹配不成功，并不会被阻塞，而是会对 pattern2 进行匹配，如果匹配成功，执行命令 command2。
	3) 这种方式相当于并行的匹配工作。

2) exp_continue 语句的作用如下：

exp_continue	说明

expect {
pattern1 { command1 }
pattern2 { command2; exp_continue }
}
	1) expect 如果首次对 pattern2 匹配成功后，会执行命令 command2。
	2) 接着执行语句 exp_continue，表示将会继续从头重新开始进行匹配。

2.3.8 函数

实例如下：

```
[root@localhost renyl]# cat test.exp
#!/usr/bin/expect

proc my_print { para1 para2 } {
        puts $para1
        puts $para2
}
my_print  first_para  second_para
[root@localhost renyl]# ./test.exp
first_para
second_para
[root@localhost renyl]#
```

3 实例

一个完整的 ssh 无密码登录脚本，如下所示：

```
[root@localhost renyl]# cat ssh_no_password.exp
#!/usr/bin/expect

#Program:
#       auto ssh login
#History:
#       2014/8/11 renyl  0.1version


#---------confirm install expect package-------
spawn which expect
expect eof

catch wait  result
set  res [lindex  $result 3]

if { "$res" == "1" } {

        send_user "=================Warning================\n"
        send_user "It haven't install expect package.\n"
        send_user "Try it again after install expect package.\n"
        send_user "========================================\n"
        exit
}


#---------------my_help function---------------
proc  my_help { program_name } {

        send_user "Usage: $program_name <IP1 PASSWORD1> <IP2 PASSWORD2> ... \n"
        send_user  "Example:  $program_name  193.168.1.2  rootroot  193.168.1.3
linux_root \n"
        send_user "Try it again\n"
        exit
}

if { $argc < 2 } {

        my_help  $argv0
}
```

```
#---------------create key file ---------------
if { [file exists /root/.ssh/id_rsa] == 0 || [file exists /root/.ssh/id_rsa.pub]
== 0 } {
        system rm -rf /root/.ssh/id_rsa*

        spawn ssh-keygen -t rsa
        expect {
                "id_rsa" { send "\n"; exp_continue }
                "passphrase" { send "\n"; exp_continue }
                "same" { send "\n"; exp_continue }
               "eof" { send_user "It have created key file.\n" }
       }
}

sleep 3
set timeout 120

#----------execute ssh-copy-id command------------
set LOOP 0
while { $LOOP < $argc } {
       set IP  [lrange $argv  $LOOP $LOOP]
       incr  LOOP
       set PASSWORD [lrange $argv $LOOP $LOOP]
       incr  LOOP

       spawn ssh-copy-id  $IP
       expect {
               "yes/no" { send "yes \n"; exp_continue }
                "password" { send "$PASSWORD\n";exp_continue }
                 "eof"  {send_user "$IP have be logined without password.\n"}
       }
}

puts "==============completed================\n"
```

参考文献

1) http://www.thegeekstuff.com/2010/10/expect-examples/
2) http://www.linuxeye.com/Linux/expect.html