Expect脚本学习

2015/3/9

renyl

# 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。

# 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。

## 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。

## 常用语法

### if语句

实例如下：

|  |
| --- |
| [root@localhost renyl]# cat test.exp  #!/usr/bin/expect  必须有空格  必须有空格  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]# |

### for语句

实例如下：

|  |
| --- |
| [root@localhost renyl]# cat test.exp  #!/usr/bin/expect    foreach i { a b c } {  puts $i  两种不同格式的for循环语法  }    for {set i 0} {$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]# |

### 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 $para  expect里的加法    #incr loop  set loop [expr {$loop + 1}]  }  [root@localhost renyl]# ./test.exp 1 2 3  1  2  3  [root@localhost renyl]# |

### switch语句

实例如下：

|  |
| --- |
| [root@localhost renyl]# cat test.exp  #!/usr/bin/expect  switch $argc {  1 {  puts "one parameter"  不需要break  }  2 {  puts "two parameters"  }  3 {  puts "three parameters"  }  default {  puts "Usage: $argv0 para1 para2 para3"  exit  }  }  [root@localhost renyl]# ./my.exp 1  one parameter  [root@localhost renyl]# |

### 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]# |

### 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]# |

### 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。 |

1. 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。 |

### 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>