Expect Login to Juniper Routers

These scripts login and retrieve “show chassis hardware” information from Juniper routers. The output of each login is written to a separate file:

```
report-<ip address>.txt
```

You can change the command “show chassis hardware” to any valid JUNOS command that you wish - or a valid Cisco IOS command if you are accessing Cisco routers.

Cut and paste the scripts below into files in the same directory and follow the instructions to execute.

Remember to change the execute mode of the “launch.sh” and “expect-script” files:

```
root@tony-laptop:~/Expect# chmod u+x launch.sh
```

File - hostip

Text file - Insert IP addresses, usernames and passwords of target hosts separated by tabs:

```
10.0.0.1            fred        bloggs
192.168.1.68        joe         bloggs
192.168.1.66        joe         bloggs
172.16.1.23         peter       pan
```

File - launch.sh

Bash file - Execute with “hostip” as argument:

```
./launch.sh hostip
```

Reads “hostip” line by line and executes “expect-script”. Passes the target host's IP address, username and password to “expect-script”. Processing terminates when EOF is reached in “hostip”.

```
#!/bin/bash
#
# Script that reads hostip file and executes an expect script to log in and get
# info from the routers. The variables $1, $2 and $3 are passed to the expect
# script
#

# Check that host ip information argument $1 has been passed
#
if [ -z "$1" ]; then echo "File not supplied"; exit; fi

# Read IP address, username & password in file "hostip" and execute expect
# script for each line
#
cat $1 | while read hostip
do
expect expect-script $hostip $username $password
done
```

File - expect-script

Expect file - Executed by “launch.sh” which passes the parameters $ipaddres, $username and $password to the script.

```
#!/usr/bin/expect -f

#####################################################
#                                                   #
# ipaddres  = passed here as $argv 0 from launch.sh #
# username  = passed here as $argv 1 from launch.sh #
# password  = passed here as $argv 2 from launch.sh #
#                                                   #
#####################################################

# Check that launch.sh supplied all the variables
if { $argc < 3 } {
   puts "Insufficient arguments were supplied\n"
   exit 1
}

# Set local variables
set ipaddres [lindex $argv 0]
set username [lindex $argv 1]
set password [lindex $argv 2]

# If one of the local variables is null, exit back to launch.sh
if { $ipaddres == "" || $username == "" || $password == "" }  {
  puts "Usage: <ip address> <username> <password>\n"
  exit 1
}

# Turn screen logging off
log_user 0

# SSH to target ip address with username
spawn ssh $username@$ipaddres

# Increase the expect buffer size - must be after the spawn command
match_max 50000

# Timeout is 8 seconds if no response from target host
# If "password" received execution passes to “send $password” below
# If "No route to host", write error to log file and exit
# If timeout, write error to log file and exit
# If "(yes/no)?" received, SSH key is required so send "yes" and proceed

set timeout 8

expect {
  "password:" {
  }
  "No route to host" {
    set outfile [open "report-$ipaddres.txt" w]
    puts $outfile "No route to host $ipaddres"
    close $outfile
    exit 1
  }
  timeout {
    set outfile [open "report-$ipaddres.txt" w]
    puts $outfile "Login timeout to $ipaddres"
    close $outfile
    exit 1
  }
  "(yes/no)?" {
    send "yes\r"
    expect "password:"
  }
}

# Configure a regular expression to match and capture the prompt
#
# The router prompt must terminate with what is specified between
# the “(....)” brackets e.g. prompt%, joe#, anything$, fred>, router~
#
# If not, add other characters between the brackets
#
# The “-re $prompt” tells expect that it is matching using a regular
# expression
#
# The “$” in “(...) $” tells expect that the character to look for is
# in the rightmost position of the prompt. Note the space after “)”
# and before “$”
#
# Note the “\\” characters in front of “$” to  "escape" it in “(...)”
# This is because “$” is a special character

set prompt "(%|#|\\$|>|~) $"
send "$password\r"
expect -re $prompt

# Put the device name in the variable “device”
send "\r"
expect -re $prompt
set device $expect_out(buffer)

# Get the output from command all the way down until prompt is received again
#
# Change this command to whatever you wish
send "show chassis hardware | no-more\r"
expect -re $prompt

# Write the result to an output file
set outfile [open "report-$ipaddres.txt" w]
puts $outfile "$device\n $expect_out(buffer)\n"
puts "Writing record $ipaddres\n"
close $outfile

send "exit\r"

log_user 1

exit
```