

[| Disclaimer](#) | [Privacy Policy](#) | [About](#) | [Contact](#)

Linux

Expect Command And How To Automate Shell Scripts Like Magic

 2017-02-27  Comments(25)

In the previous post, we talked about writing **practical shell scripts** and we saw how it is easy to write a shell script. Today we are going to talk about a tool that does magic to our shell scripts, that tool is the **Expect command** or **Expect scripting language**.

Expect command or expect scripting language is a language that talks with your interactive programs or scripts that require user interaction.





Expect scripting language works by expecting input, then the Expect script will send the response without any user interaction.

You can say that this tool is your robot which will automate your scripts.

If Expect command is not installed on your system, you can install it using the following command:

```
$ apt-get install expect
```





Or on Red Hat based systems like CentOS:

```
$ yum install expect
```

Table of Contents [\[hide\]](#)

1 Expect Command

2 Using autoexpect

3 Working with Variables

4 Conditional Tests

5 If else Conditions

6 While Loops

7 For Loops

8 User-defined Functions

9 Interact Command





Expect Command

Before we talk about expect command, Let's see some of the expect command which used for interaction:

spawn	Starting a script or a program.
expect	Waiting for program output.
send	Sending a reply to your program.
interact	Allowing you in interact with your program.

- The spawn command is used to start a script or a program like the shell, **FTP**, Telnet, SSH, SCP, and so on.
- The send command is used to send a reply to a script or a program.
- The Expect command waits for input.
- The interact command allows you to define a predefined user interaction.

We are going to type a shell script that asks some questions and we will make an Expect script that will answer those questions.

First, the shell script will look like this:



```
#!/bin/bash

echo "Hello, who are you?"

read $REPLY

f      "Can I ask you some questions?"

t      $REPLY

in     "What is your favorite topic?"

p

read $REPLY
```

Now we will write the Expect scripts that will answer this automatically:

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

set timeout -1

spawn ./questions

expect "Hello, who are you?\r"

send -- "Im Adam\r"

expect "Can I ask you some questions?\r"

send -- "Sure\r"

expect "What is your favorite topic?\r"

send -- "Technology\r"

expect eof
```

The first line defines the expect command path which is



```
#!/usr/bin/expect
```

.

On the second line of code, we disable the timeout. Then start our script using spawn and.



I use spawn to run any program we want or any other interactive script.



... ..naining lines are the Expect script that interacts with our shell script.



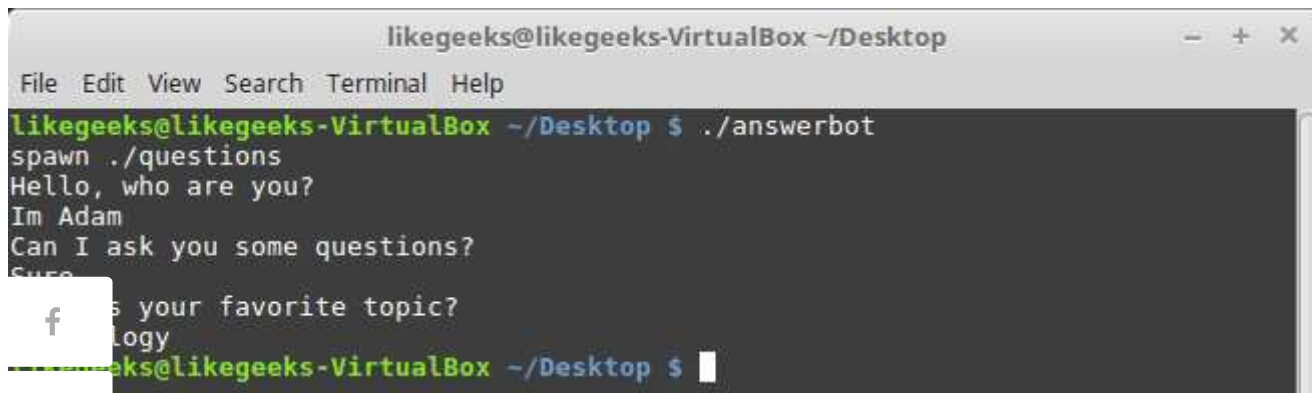
I ne last line if the end of file which means the end of the interaction.

Now Showtime, let's run our answer bot and make sure you make it executable.

```
$ chmod +x ./answerbot
```

```
$ ./answerbot
```





```
likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./answerbot
spawn ./questions
Hello, who are you?
Im Adam
Can I ask you some questions?
Sure
What is your favorite topic?
Technology
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

• Now All questions are answered as we expect.

• If you get errors about the location of Expect command you can get the location using the `which` command:

```
$ which expect
```

We did not interact with our script at all, the Expect program do the job for us.

The above method can be applied to any interactive script or program. Although the above Expect script is very easy to write, maybe the Expect script little tricky for some people, well you have it.

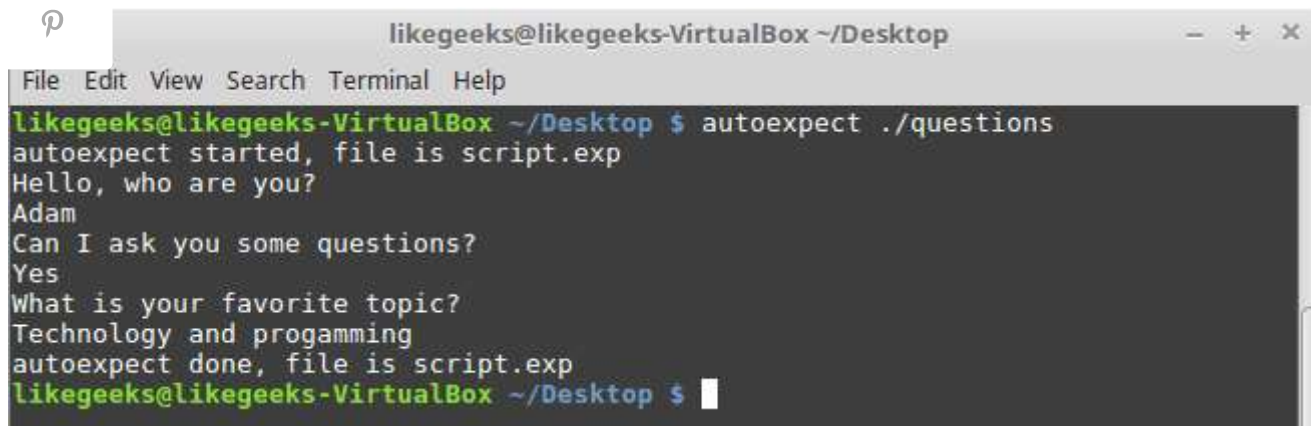
Using autoexpect

To build an expect script automatically, you can use the autoexpect command.

autoexpect works like expect, but it builds the automation script for you. The script you want automate is passed to autoexpect as a parameter and you answer the questions and your answers are saved in a file.



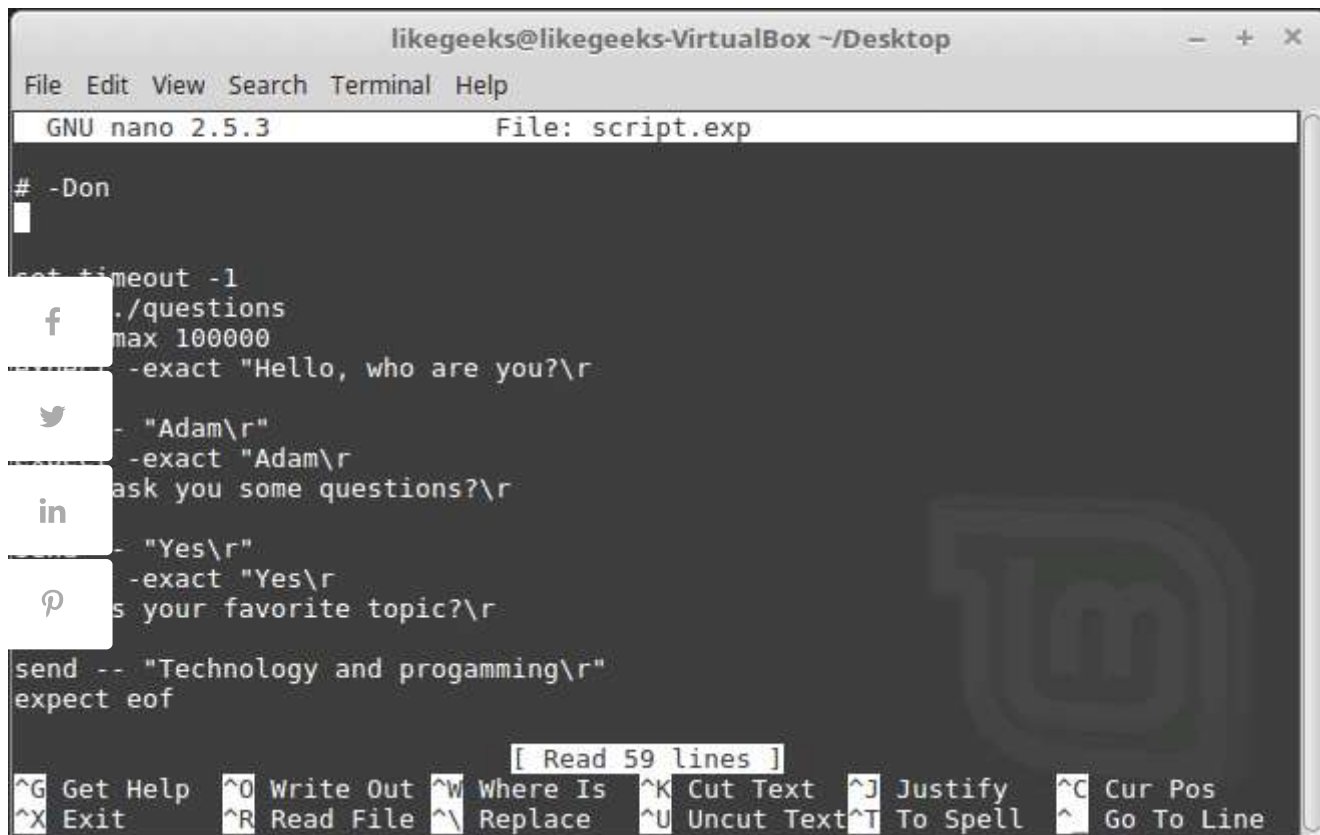
autoexpect ./questions



```
likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
likegeeks@likegeeks-VirtualBox ~/Desktop $ autoexpect ./questions
autoexpect started, file is script.exp
Hello, who are you?
Adam
Can I ask you some questions?
Yes
What is your favorite topic?
Technology and programming
autoexpect done, file is script.exp
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

A file is generated called script.exp contains the same code as we did above with some additions that we will leave it for now.





```

likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
GNU nano 2.5.3 File: script.exp

# -Don

set timeout -1
spawn ./questions
max 100000
expect -exact "Hello, who are you?\r"

- "Adam\r"
-exact "Adam\r"
ask you some questions?\r

- "Yes\r"
-exact "Yes\r"
is your favorite topic?\r

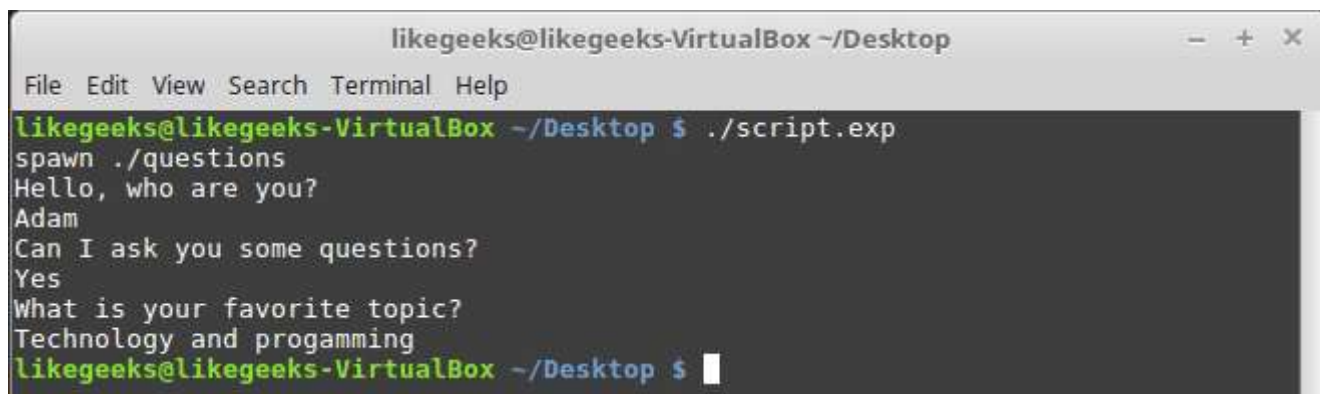
send -- "Technology and programming\r"
expect eof

[ Read 59 lines ]

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line

```

If you run the auto generated file script.exp, you will see the same answers as expected:



```

likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./script.exp
spawn ./questions
Hello, who are you?
Adam
Can I ask you some questions?
Yes
What is your favorite topic?
Technology and programming
likegeeks@likegeeks-VirtualBox ~/Desktop $

```

Awesome!! That super easy.

There are many commands that produce changeable output, like the case of FTP programs, the expect script may fail or stuck. To solve this problem, you can use wildcards for the changeable data to make your script more flexible.



Working with Variables

The set command is used to define variables in Expect scripts like this:

 `set MYVAR 5`



— To pass the variable, precede it with \$ like this \$VAR1



In the command line arguments in Expect scripts, we use the following syntax:

```
set MYVAR [lindex $argv 0]
```

Here we define a variable MYVAR which equals the first passed argument.

You can get the first and the second arguments and store them in variables like this:

```
set my_name [lindex $argv 0]
```

```
set my_favorite [lindex $argv 1]
```

Let's add variables to our script:



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

set my_name [lindex $argv 0]

set my_favorite [lindex $argv 1]

f
    timeout -1

t
    spawn ./questions

in
    expect "Hello, who are you?\r"

p
    send -- "Im $my_name\r"

expect "Can I ask you some questions?\r"

send -- "Sure\r"

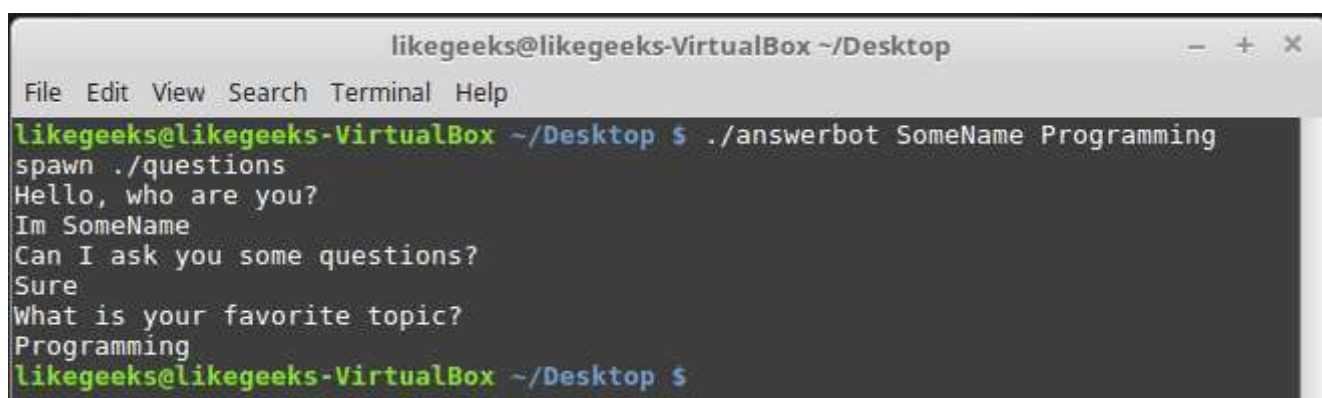
expect "What is your favorite topic?\r"

send -- "$my_favorite\r"

expect eof
```

Now try to run the Expect script with some parameters to see the output:

```
$ ./answerbot SomeName Programming
```

A screenshot of a terminal window titled 'likegeeks@likegeeks-VirtualBox ~/Desktop'. The terminal shows the execution of the script './answerbot SomeName Programming'. The output of the script is: 'spawn ./questions', 'Hello, who are you?', 'Im SomeName', 'Can I ask you some questions?', 'Sure', 'What is your favorite topic?', 'Programming'. The prompt returns to the user: 'likegeeks@likegeeks-VirtualBox ~/Desktop \$'.

Awesome!! Now our automated Expect script is more dynamic.

Conditional Tests

... can write conditional tests using braces like this:

```
expect {  
    in  
        "something" { send -- "send this\r" }  
        "*another" { send -- "send another\r" }  
}
```

We are going to change our script to return different conditions, and we will change our Expect script to handle those conditions.

We are going to emulate different expects with the following script:

```
#!/bin/bash  
  
let number=$RANDOM  
  
if [ $number -gt 25000 ]; then  
    echo "What is your favorite topic?"  
  
else  
    echo "What is your favorite movie?"  
  
fi  
  
read $REPLY
```



A random number is generated every time you run the script and based on that number, we put a condition to return different expects.

Let's make out Expect script that will deal with that.

```
f  sr/bin/expect -f
t  :imeout -1
in  1 ./questions
p  st {

    "*topic?" { send -- "Programming\r" }

    "*movie?" { send -- "Star wars\r" }

}

expect eof
```

Very clear. If the script hits the topic output, the Expect script will send programming and if the script hits movie output the expect script will send star wars. Isn't cool?



If else Conditions

You can use if/else clauses in expect scripts like this:

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

if
    set NUM 1
    if { $NUM < 5 } {
        puts "\Smaller than 5\n"
    } elseif { $NUM > 5 } {
        puts "\Bigger than 5\n"
    } else {
        puts "\Equals 5\n"
    }
}
```

Note: **The opening brace must be on the same line.**

While Loops

While loops in expect language must use braces to contain the expression like this:



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

set NUM 0

while { $NUM <= 5 } {

    f      puts "\nNumber is $NUM"
    t      set NUM [ expr $NUM + 1 ]
    in
    ,
    p
    puts ""
}
```

For Loops

To make a for loop in expect, three fields must be specified, like the following format:

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

for {set NUM 0} {$NUM <= 5} {incr NUM} {
```



```
puts "\nNUM = $NUM"
```

```
}
```

```
puts ""
```

f

🐦

in

p

User-defined Functions

You can define a function using proc like this:

```
proc myfunc { TOTAL } {  
  
    set TOTAL [expr $TOTAL + 1]  
  
    return "$TOTAL"  
  
}
```

And you can use them after that.

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

```
proc myfunc { TOTAL } {
```




```
set TOTAL [expr $TOTAL + 1]
```

```
return "$TOTAL"
```

```
,
```

f

```
NUM 0
```

🐦

```
> { $NUM <= 5 } {
```

in

```
puts "\nNumber $NUM"
```

p

```
set NUM [myfunc $NUM]
```

```
}
```

```
puts ""
```

Interact Command

Sometimes your Expect script contains some sensitive information that you don't want to share with other users who use your Expect scripts, like passwords or any other data, so

want your script to take this password from you and continuing automation normally.

The interact command reverts the control back to the keyboard.

When this command is executed, Expect will start reading from the keyboard.

f

ell script will ask about the password as shown:

tw

```
#!/usr/bin/bash
```

in

```
echo "Hello, who are you?"
```

p

```
read $REPLY
```

```
echo "What is your password?"
```

```
read $REPLY
```

```
echo "What is your favorite topic?"
```

```
read $REPLY
```

Now we will write the Expect script that will prompt for the password:

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

```
set timeout -1
```

```
spawn ./questions
```

```
expect "Hello, who are you?\r"
```

```
send -- "Hi Im Adam\r"
```

```
expect "*password?\r"
```

```
interact ++ return
```



```
send "\r"

expect "*topic?\r"

-- "Technology\r"
f
t eof
t
in
p
```

After you type your password type ++ and the control will return back from the keyboard to the script.

Expect language is ported to many languages like C#, Java, Perl, **Python**, Ruby and Shell with almost the same concepts and syntax due to its simplicity and importance.

Expect scripting language is used in quality assurance, network measurements such as echo response time, automate file transfers, updates, and many other uses.

I hope you now supercharged with some of the most important aspects of Expect command, autoexpect command and how to use it to automate your tasks in a smarter way.

Thank you.



Share on Facebook



Tweet on Twitter





Mokhtar Ebrahim

I'm working as a Linux system administrator since 2010. I'm responsible for maintaining, securing, and troubleshooting Linux servers for multiple clients around the world. I love writing shell and Python scripts to automate my work.






Related Articles



31+ Examples for sed Linux Command in Text Manipulation

📅 2017-02-19

In the previous post, we

 about bash functions
 w to use them from
 the command line directly
 e saw some other cool
 oday we will talk about
 a very useful tool for string
 manipulation called sed or
 sed Linux command. Sed is
 used to work with text files
 like log files, [...]

How to install Linux step-by-step

📅 2017-01-30

How to install Linux? After you've chosen the Best Linux Distro, now it's the time to know how to install Linux. If you want to install Linux, there are 2 ways to do that: The first way is to download the Linux distribution you want and burn it into a DVD or USB stick and boot [...]

Bash Scripting Part6 – Create and Use Bash Functions

📅 2017-02-17

Before we talk about bash functions, let's discuss this situation. When writing bash scripts, you'll find yourself that you are using the same code in multiple places. If you get tired of writing the same lines of code again and again in your bash script, it would be nice to write the block of code [...]

◀ How to write practic...

Performance Tuning U...

25 thoughts on “Expect command and how to automate shell scripts like magic”

Joydeep Das

2018-08-27 at 11:07 am

That a very well written article 😊

Reply

Mokhtar Ebrahim

2018-08-27 at 5:47 pm

Thank you very much!



Reply

Balaji

2018-10-29 at 7:22 am



I need some suggestion



How can i automate a password entry for a command and that command is not to login to another server its like a permission within the server.



I need to copy a file from specific path but when i try to copy it will ask me to enter password for security reason , whether can i automate this one..

Here's the script i have used

```
#!/bin/ksh
copy_command()
{
cp /xxx/xxxxx/logs/SAMPLE.2016-11-29 /xxx/xxxxx/sample/folder2/ >
/xxx/xxxxx/sample/log.txt 2>&1
}
error_check()
{
if [ -f /xxx/xxxxx/sample/log.txt ]
then
if [ -s /xxx/xxxxx/sample/log.txt ]
then
echo "File exists and not empty"
if grep -q 'Cannot find the requested security attribute' /xxx/xxxxx/sample/log.txt;
then
echo "found"
efskeymgr -o ksh
expect "*EFS password*\r"
```



```
send — "abcd@123\r"
```

```
fi
```

```
fi
```

```
fi
```

```
}
```

```
main()
```

```
{
```

```
copy_command
```

```
retval=$?
```

```
error_check
```

```
fi
```

```
}
```

```
#Calling Main Function
```

```
main
```



Reply

Mokhtar Ebrahim

2018-10-30 at 7:48 am

Did you test this code?

If so, what is the error?

Reply

Nitin Shelke

2018-11-16 at 12:24 pm

Thanks for the article; Would i be able to spawn a function within the same script?

If so could you give me an example.

Spawning a script abcd.ksh within an expect script (xyz.ksh) is working but i would like to make abcd.ksh as a function inside xyz.ksh and call the same. ^

Could you give me an example please

Reply



Mokhtar Ebrahim

2018-11-18 at 6:58 am



Yes, you can.



To spawn a function from inside the same file, you can use the `-c` of the bash like this:



```
spawn bash -c "myfunc"
```

Reply

Aaron

2018-12-12 at 9:20 am

Is it possible to use a variable stored by bash? Or is there a way to get input without showing the input like displaying a password to the screen? I know you can do it in bash with `read -s` I am trying to learn expect because I would like to remotely change the passwords on windows 10, linux, and a couple different firewalls every 60 days. So having something like this would be beneficial to get a password one time and then send it to change the password on each OS.

Reply

Mokhtar Ebrahim

2018-12-12 at 10:49 am

If your calling process exported an environmental variable, you can use it like this:

```
$ : env(myvar)
```



But since you are using expect, you don't need to use Bash, you can access variables like this:

```
set myvar [lindex $argv 0]
```

```
if {$myvar == "myval"}
```

```
#Your code goes here
```



Hope that helps.



Reply



JJ

2019-02-04 at 5:47 pm

Your articles are very well written and a great help! Thanks for all your hard work and thank you for sharing!

Reply

Mokhtar Ebrahim

2019-02-04 at 6:55 pm

Thank you very much for the kind words!

That drives me to do my best.

Reply

Guru

2019-03-25 at 2:38 pm

Below is the code I used but didn't get output with autoexpect command..it is struck after asking first question..

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



```
set timeout -1
spawn ./questions
expect "Hello, How are you?\r"
send — " Hi Im Adam\r"
expect "*password?\r"
interact ++ return
send "\r"
expect "*topic?\r"
send — "Technology\r"
expect eof
```



Reply

Mokhtar Ebrahim

2019-03-25 at 4:41 pm

It depends on what is in your questions file.

Reply

Ryan

2019-04-03 at 5:47 am

awesome!

Reply

Mokhtar Ebrahim

2019-04-03 at 7:37 am

Thanks!



Reply

Girish

2019-04-08 at 2:51 pm



Here I'm trying to execute a simple command, I'm not getting any error however, its coming out without issuing the last command (df -gt)



kindly let me know what is the error or mistake?



```
#!/usr/bin/expect
```



```
set username [lindex $argv 0]
```

```
set hostname [lindex $argv 1]
```

```
set password [lindex $argv 2]
```

```
set username "inxxxxxx"
```

```
set hostname "xxxxxx111"
```

```
set password "xxxxxxxxxx2299"
```

```
spawn ssh $hostname
```

```
expect "$username@$hostname\'s password: " {send "$password\r"}
```

```
#expect "password: " {send "$password\r"}
```

```
expect "\] " {send "sudo su -\r"}
```

```
expect "root-\] " {send "su - db2v105\r"}
```

```
expect "db2v105\> " {send "df -gt\r"}
```

Reply

Mokhtar Ebrahim

2019-04-08 at 4:52 pm



Since there is no error as you said, then you should debug your expect script or run it step by step to check the output of every line.

To debug your script, you can use -D option like this:

```
$ expect -D 1 yourscript.file
```

Also, you can run a line each time using -b option like this:

```
$ expect -b
```

Hope that helps!



Reply



Girish

2019-04-12 at 1:38 pm

Ok, that didn't work, however can you give simple program or explain program where we can issue command, I may learn from there.

Thank you!

Mokhtar

Reply

Basavaraj

2019-06-13 at 11:32 am

i want to run some test programs one after another.

by using while loop i ran it.

first test program got failed and stuck i mean i am not getting response from CLI to run second test program.

shell script to run Test programs

```
n=1
```

```
while (($n <= 500))
```



```
do
./DOWNLOAD0350 -debug
sleep 5
./DOWNLOAD0340 -debug
sleep 5
n=$(( n+1 ))
done
```



CLI Output:

```
-* Invalid compile option : customize_code=74
fail : <> (line#108) [FW Rev:40F82840]
```

Reply

Mokhtar Ebrahim

2019-06-13 at 1:55 pm

That's a wired message.

It's a script, not a code to compile. How do you run your script?

Reply

Ludo Game

2019-06-28 at 5:18 am

I think this is one of the best blogs for me because this is really helpful for me.

Thanks for sharing this valuable information for free

Reply

Mokhtar Ebrahim

2019-06-28 at 11:58 am



You're welcome! Thank you very much!

I'll do my best always.

Reply



Luis Mtz Aguilar



2019-07-04 at 3:54 pm



Hi all



Do you know why I cant run "Expected" when I use a cron task ?

the script works when I run it manually but does not works using a cron task

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

```
set client client
```

```
spawn scp [lindex $argv 0] plclient@x.x.x.x:/dfcxact/workarea/Global
```

```
send "\r"
```

```
expect "password:"
```

```
send "$client"
```

```
send "\r"
```

```
interact
```

Thanks

Reply

Luis Mtz Aguilar

2019-07-04 at 4:19 pm

Hi all

I got the fix...



I just set

"expect eof" instead of "interact"

Regards



Prasenjeet Sahare

2019-11-18 at 8:18 am

hello,

how to give control to keyboard entirely.

Only after pressing lets say cntrl+c or any key control should come back to expect shell script.

Thanks in advance!

Reply

Mokhtar Ebrahim

2019-11-18 at 8:51 am

As mentioned in the tutorial, the interact command gives control to the keyword.

There is an example explains that.

Regards,

Reply

Leave a Reply



Your email address will not be published. Required fields are marked *

Comment

f

.

🐦

.

in

...ne *

p

Email *

Replies to my comments ▾

Notify me of followup comments via e-mail. You can also [subscribe](#) without commenting.

Post Comment

Email

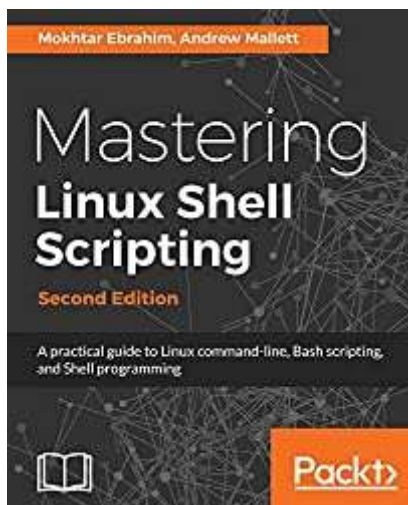
Subscribe

Advertisements



[report this ad](#)

My Published Book



Latest Posts



Exiting/Terminating Python scripts (Simple Examples)

📅 2020-05-06

20+ examples for NumPy matrix multiplication





📅 2020-05-05



Five Things You Must Consider Before 'Developing an App'

📅 2020-04-29



Caesar Cipher in Python (Text encryption tutorial)

📅 2020-04-28



NumPy loadtxt tutorial (Load data from files)

📅 2020-04-21

Advertisements




[report this ad](#)


Related



15+ examples for yum update command

 2020-02-12

Linux find command tutorial (with examples)

 2019-12-17





f

t

in

p

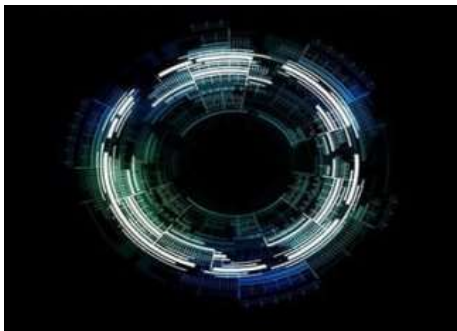
15+ examples for Linux cURL command

📅 2019-12-09



Grep command in Linux (With Examples)

📅 2019-11-27



Understanding Linux runlevels the right way

📅 2019-11-21

Latest Comments

💬 Mokhtar Ebrahim on 30 Examples for Awk Command in Text Processing

💬 Anu on 30 Examples for Awk Command in Text Processing

💬 swetha on 30 Examples for Awk Command in Text Processing



💬 Mokhtar Ebrahim on 30 Examples for Awk Command in Text Processing

💬 swetha on 30 Examples for Awk Command in Text Processing

Advertisements



Picked For You



Your Guide to Becoming a Better Android App Developer

📅 2018-12-13



30 Examples for Awk Command in Text Processing

📅 2017-02-21

Create and Use Dynamic Laravel Subdomain Routing

📅 2017-05-20





f



t

in

p

20+ examples for NumPy matrix multiplication

📅 2020-05-05



NLP Tutorial Using Python NLTK (Simple Examples)

📅 2017-09-21





[report this ad](#)

