Sunday, March 05, 2017

LIKE GEEKS

LINUX

# Expect Command And How To Automate Shell Scripts Like Magic

*February 27, 2017* *admin* 0 Comments

In the previous post we've talked about writing **practical shell scripts** and we've seen how it is easy to write a shell script and we've used most of our knowledge we've discussed on the previous posts, today we are going to talk about a tool that does magic to our shell scripts,that tool is expect command or expect programming language.  Expect command or **expect programming language** is a language that talks with your interactive programs or scripts that require user interaction for input. Expect works by expecting input, and upon receiving the expected input, the Expect script will send the response without any user interaction, just like magic.

**Expect command** or expect programming language is a language that talks with your interactive programs or scripts that require user interaction for input. Expect works by expecting input, and upon receiving the expected input, the Expect script will send a response without any user interaction, just like magic.

You can say that this tool is your robot who will automate some of your daily tasks

If expect command if 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
```

**Our main points are:**

Interact command

## Expect command

Before we talk about expect command, we need to look at the basic commands that Expect uses to interact with a program or script

| | |
|---|---|
| spawn | Starts a process or program |
| send | Sends a reply to a process or program |
| expect | Waits for output from a process or program |
| interact | Allows you in interact with a process or program |

- The spawn command is used to start up a process or application, such as the shell, FTP, telnet, ssh, scp, and so on.
- The send command to send a reply to a process or program. This works like echo in our shell scripts. You must specify the proper cursor control when talking with an interactive application, using \r for carriage return and \n for each new line. Carriage returns and line feeds are not automatically implied.
- The expect command is used to wait for specific output from a process or program, just as read waits for input in a shell script
- The interact command, which enables us to program a predefined user interaction point in our Expect scripts. For example, we might want to have an Expect script that will log us into an FTP site and then turn control over to the end user.

Don't worry we are going to use those commands one by one.

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
echo "Can I ask you some questions?"
read $REPLY
echo "What is your favorite topic?"
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 executing shell, but in this case #!/usr /bin/expect we specifies the path to Expect command because those lines will be interpreted by expect command.

On the first line of code, we disable timeout by setting the Expect timeout variable to -1. Next, we start our interactive script

We can use spawn to run any program we want or any other interactive script

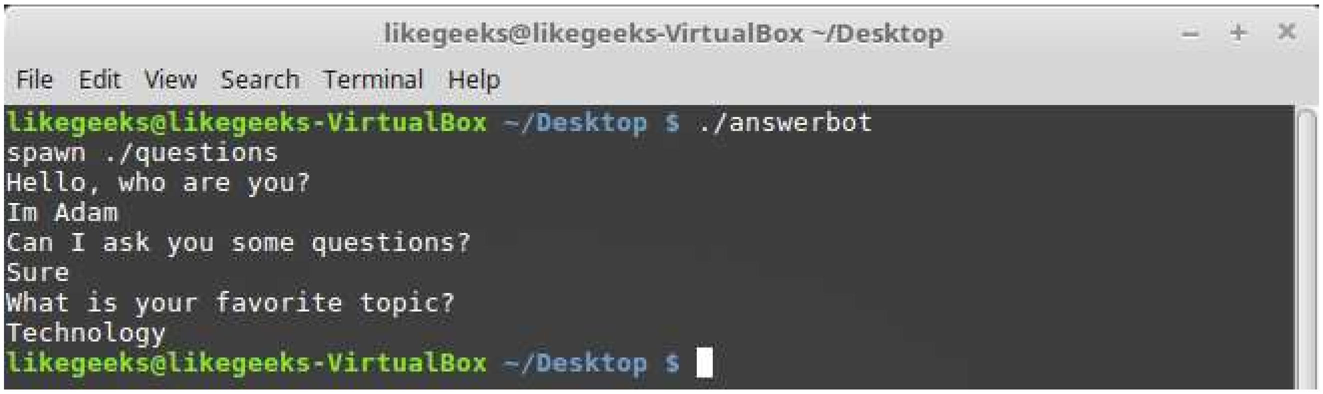
The remaining lines are the expect script that interacts with our shell script

The last Expect statement tells Expect the end of file is expected and the script ends

Now Showtime, let’s run our answerbot 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 $ 
```

Cool!! All questions are answered as we expect.

if you get errors about the location of expect command you can get the location using which command

```
$ which expect
```

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

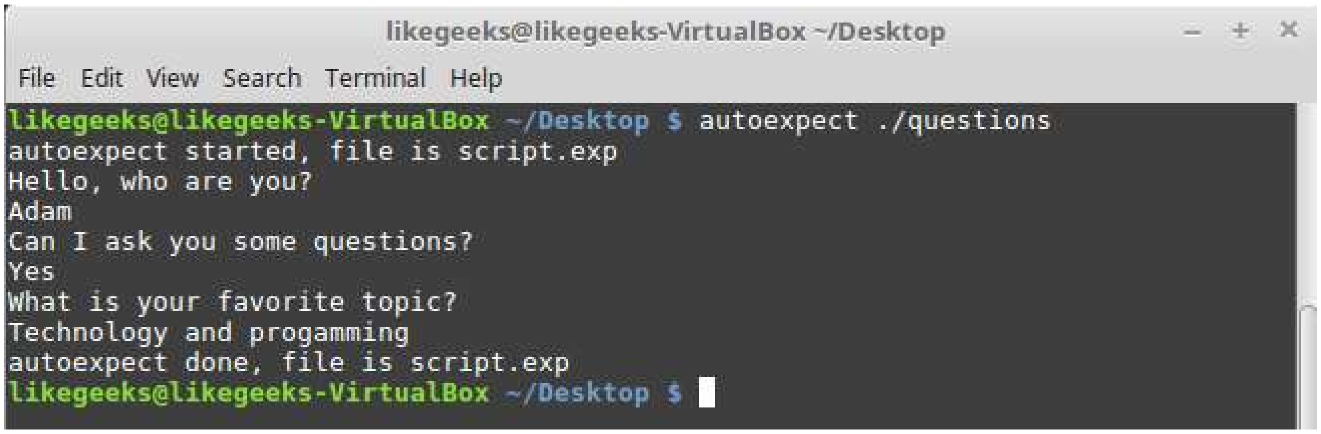
The above method can be applied to any interactive script or program.Although the above expect script is very easy to write but you maybe want to speed the automation, well you have it.

## Using autoexpect to Automatically Create an Expect Script

To create an Expect script quickly and automatically, we can use autoexpect command.

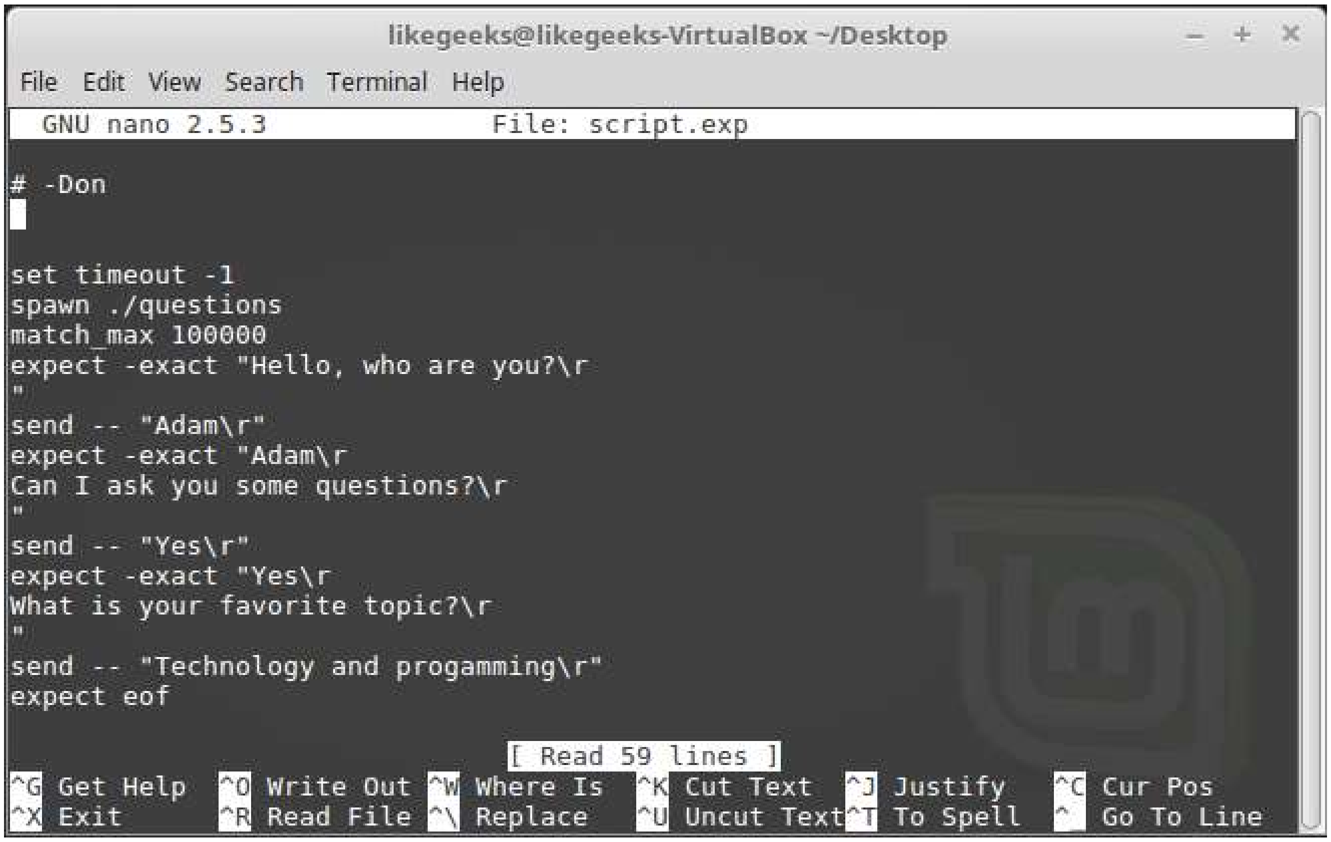
autoexpect works the same way the script command works. We enter autoexpect on the command line, and a new session is started that saves all the keystrokes in a file until the application ends.

```
$ autoexpect ./questions
```


likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help

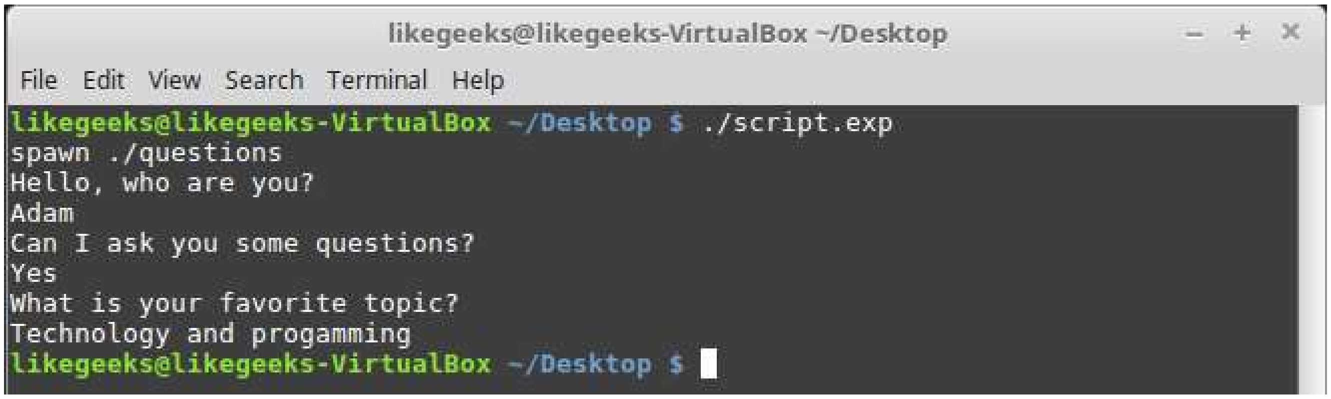
```
likegeeks@likegeeks-VirtualBox ~/Desktop $ autoexpect ./questions
autoexpect started, file is script.exp
Hello, who are you?
Adam
Can I ask you some questions?
Yes
What is your favorite topic?
Technology and progamming
autoexpect done, file is script.exp
likegeeks@likegeeks-VirtualBox ~/Desktop $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

A file generated called script.exp contains the same code 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
match_max 100000
expect -exact "Hello, who are you?\r
"
send -- "Adam\r"
expect -exact "Adam\r
Can I ask you some questions?\r
"
send -- "Yes\r"
expect -exact "Yes\r
What is your favorite topic?\r
"
send -- "Technology and progamming\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 result


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 progamming
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

Awesome!! That super easy.

There are many commands that use timestamps in output or show transfer statistics or any changeable output, as in the case with FTP programs, the Expect script can fail or stuck. If you get this type of output when using autoexpect, either delete the data that changes from the Expect script or use wildcards to represent the data.

## Working with Variables

The Expect command set is used to define variables and assign values. For example, to assign the value 5 to the variable VAR1, we use the following syntax

```
set VAR1 5
```

To access the variable, we use the same technique as we do in a normal shell script —add a $ in front of the variable name $VAR1

To define command-line arguments to Expect script variables, we use the following syntax:

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

The set command defines the variable VAR as a pointer to the first command-line argument $argv 0.

For our shell script, we want to define the first argument as the name, my_name. The second command-line argument is assigned to the favorite, my_favorite. So the variables definition will look like this

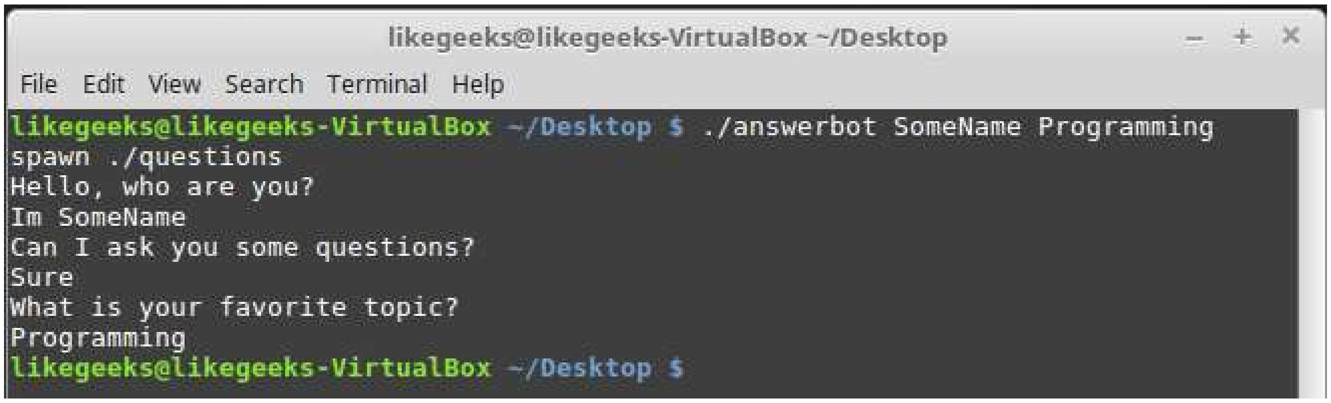
```
set my_name [lindex $argv 0]

set my_favorite [lindex $argv 1]
```

Let's modify our script with the new variables.

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

set my_name [lindex $argv 0]

set my_favorite [lindex $argv 1]

set timeout -1

spawn ./questions

expect "Hello, who are you?\r"

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
```


likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help

```
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./answerbot SomeName Programming
spawn ./questions
Hello, who are you?
Im SomeName
Can I ask you some questions?
Sure
What is your favorite topic?
Programming
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

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

## Conditional Tests

Sometimes what we are expecting varies. Based on the output so your expect script much be ready for those conditions. Expect handle this type of conditional test by using braces like this.

```
expect {
    "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

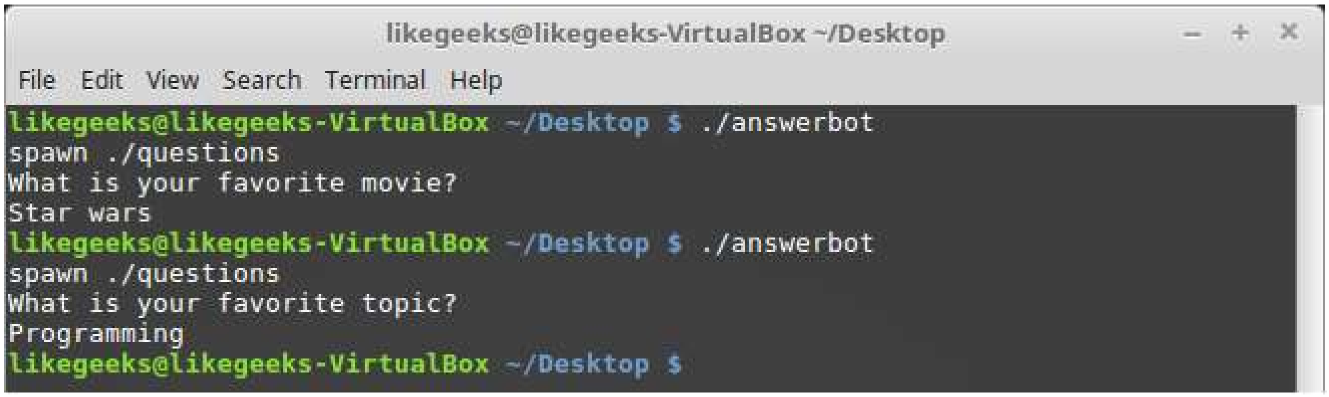
```
#!/bin/bash
let number=$RANDOM
if [ $number -gt 25000 ]
then
echo "What is your favorite topic?"
else
echo "What is your favorite movie?"
fi
read $REPLY
```

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

Let’s make out expect script that will deal with that.

```
#!/usr/bin/expect -f
set timeout -1
```

```
spawn ./questions
expect {
    "*topic?" { send -- "Programming\r" }
    "*movie?" { send -- "Star wars\r" }
}
expect eof
```


likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help

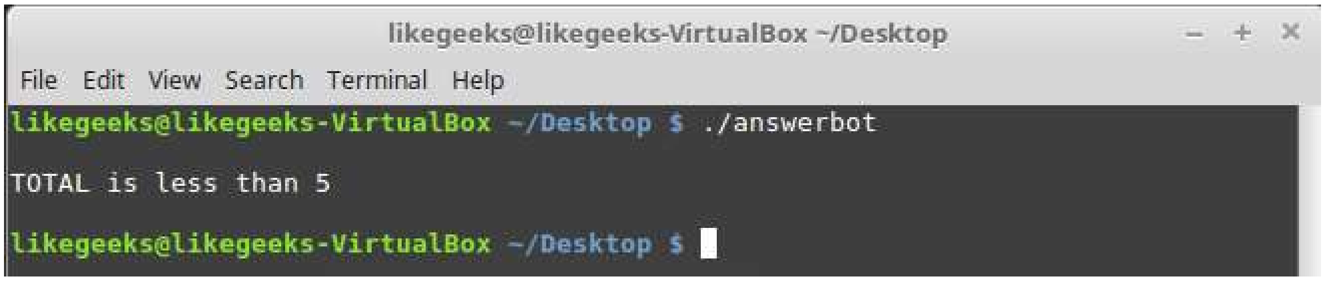
```
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./answerbot
spawn ./questions
What is your favorite movie?
Star wars
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./answerbot
spawn ./questions
What is your favorite topic?
Programming
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

very clear to you that 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

Expect also provides if else and other basic control structures you can write it like this

```
#!/usr/bin/expect -f
set TOTAL 1
if { $TOTAL < 5 } {
puts "\nTOTAL is less than 5\n"
} elseif { $TOTAL > 5 } {
puts "\nTOTAL greater than 5\n"
} else {
puts "\nTOTAL is equal to 5\n"
}
```


likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help

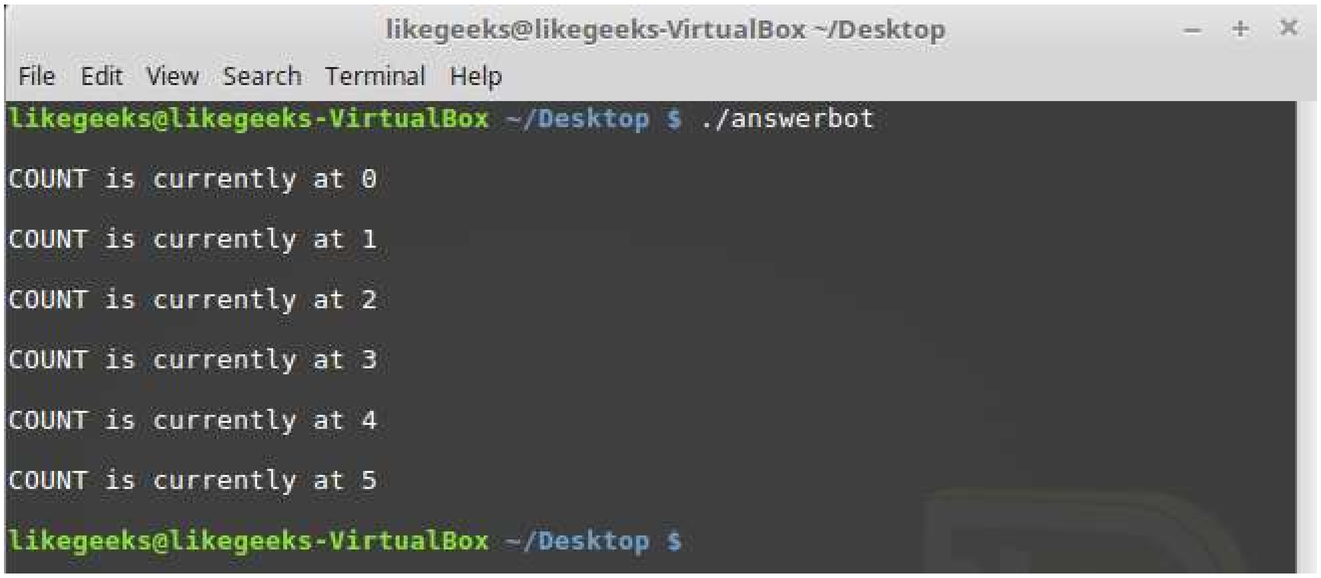
```
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./answerbot

TOTAL is less than 5

likegeeks@likegeeks-VirtualBox ~/Desktop $
```

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

## While Loops

Expect uses similar syntax for a while loop as our shell scripts use. But again, you must use braces to contain the expression

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

set COUNT 0

while { $COUNT <= 5 } {

puts "\nCOUNT is currently at $COUNT"

set COUNT [ expr $COUNT + 1 ]

}

puts ""
```


likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help

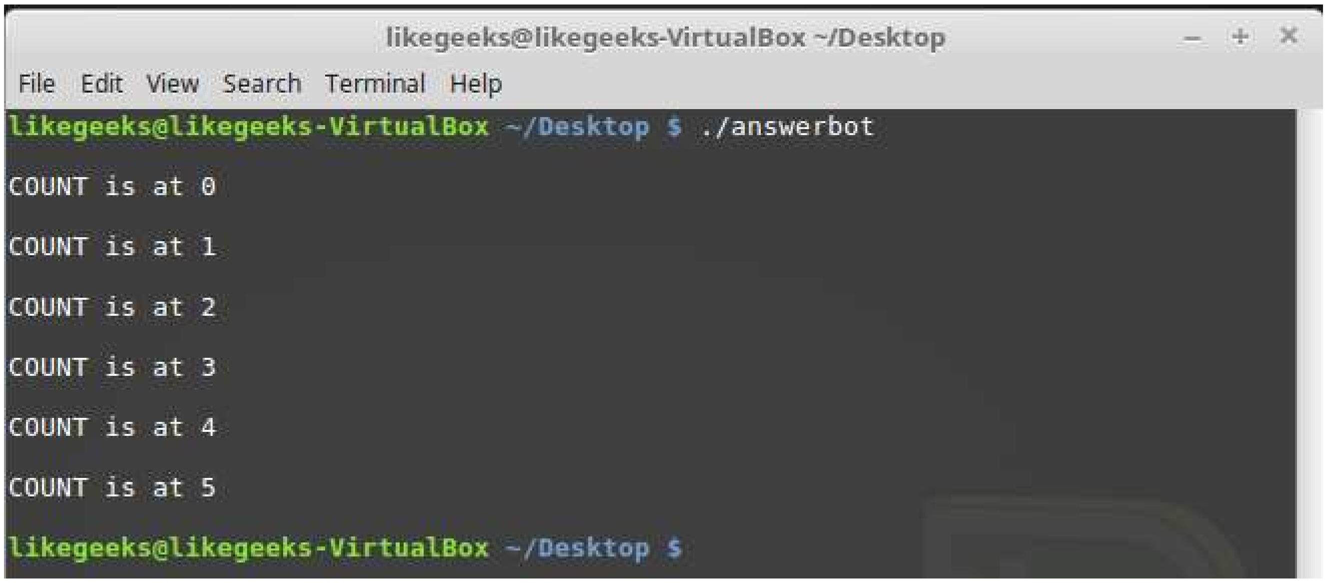
```
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./answerbot

COUNT is currently at 0

COUNT is currently at 1

COUNT is currently at 2

COUNT is currently at 3

COUNT is currently at 4

COUNT is currently at 5

likegeeks@likegeeks-VirtualBox ~/Desktop $
```

## For Loops

The for loop in Expect works a little differently from a for loop in a shell script.

Three fields must be specified like the following format

```
#!/usr/bin/expect -f
for {set COUNT 0} {$COUNT <= 5} {incr COUNT} {
puts "\nCOUNT is at $COUNT"
}
puts ""
```



```
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./answerbot

COUNT is at 0

COUNT is at 1

COUNT is at 2

COUNT is at 3

COUNT is at 4

COUNT is at 5
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

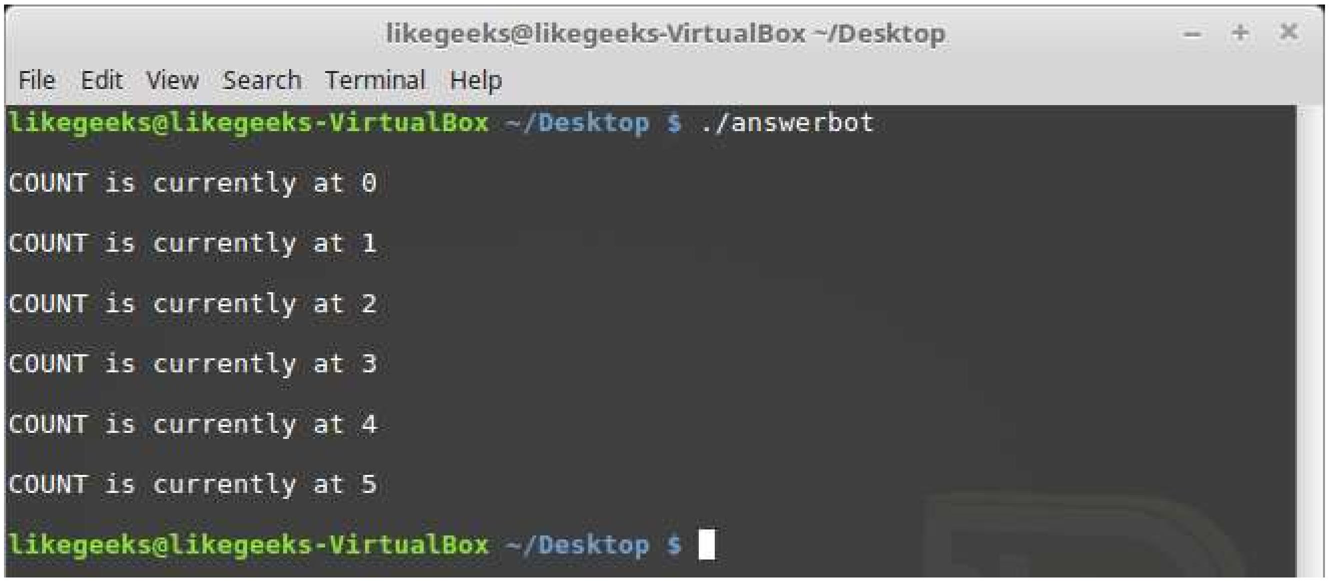
## User-defined functions

Expect supports functions using the proc Expect operator. You can define function like this

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

And you can use them after that.

```
#!/usr/bin/expect -f
proc myfunc { MY_COUNT } {
set MY_COUNT [expr $MY_COUNT + 1]
return "$MY_COUNT"

}

set COUNT 0

while {$COUNT <= 5} {

puts "\nCOUNT is currently at $COUNT"

set COUNT [myfunc $COUNT]

}

puts ""
```



```
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./answerbot

COUNT is currently at 0

COUNT is currently at 1

COUNT is currently at 2

COUNT is currently at 3

COUNT is currently at 4

COUNT is currently at 5

likegeeks@likegeeks-VirtualBox ~/Desktop $
```

## Interact command

Sometimes your expect script contains some sensitive information that you don;t want to share with other users who uses your expect scripts like passwords or any other data,so you want your script to take this password from you and continuing automation normally.

expect provides a command that turns control from the spawned script or program to you this command is interact.

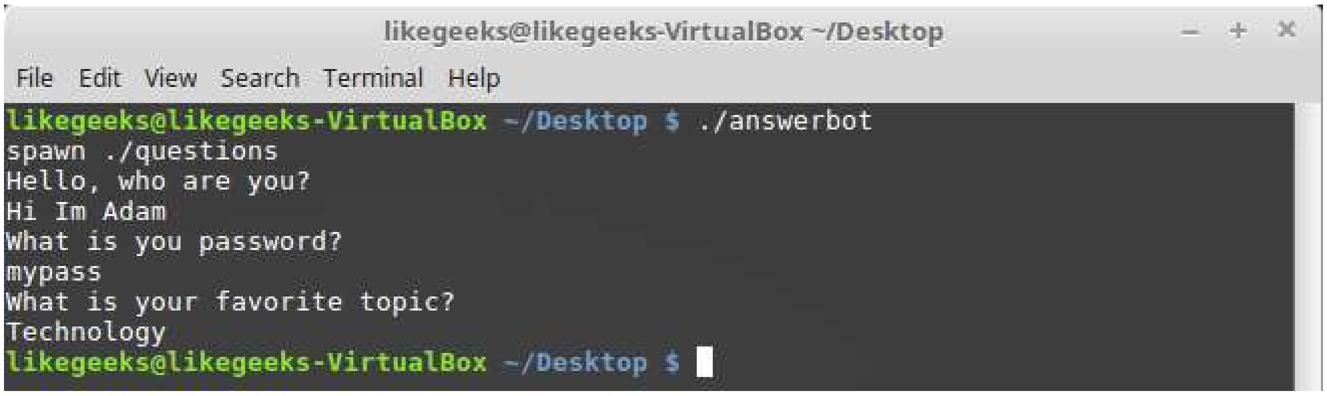
When this command is executed, Expect script stops reading commands from the script and instead begins reading from the keyboard and this is very useful.

This shell script will ask about the password as shown

```
#!/bin/bash

echo "Hello, who are you?"

read $REPLY

echo "What is you 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"

send -- "Technology\r"

expect eof
```


likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help

```
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./answerbot
spawn ./questions
Hello, who are you?
Hi Im Adam
What is you password?
mypass
What is your favorite topic?
Technology
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

we write ++ return with interact to return back the control to the expect script

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 programming language is used in quality assurance, network measurements such as echo response time, automate file transfers and updates and many other productions use.

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

Thank you.

**7**

**Admin**

https://likegeeks.com

## RELATED ARTICLES

LINUX

**Regex tutorial for Linux**

February 23, 2017 admin

LINUX

**How to write practical shell script**

February 25, 2017 admin

LINUX

**Main Linux Commands Easy Guide**

January 31, 2017 admin

In order to successfully working with the Linux sed editor and the awk command in your shell scripts you has to understand regular expressions or in short regex and to be accurate in our case it is bash regex, since there are many engines for regex you can use and we here in this regex […]

**16**

In the last post, we've talked about regex and we see how to use them in sed and awk for text processing and we discussed before Linux sed command and awk command. During the series, we write small shell scripts but we didn't mix things up, I think we should take a small step and write […]

**41**

Main Linux Commands That You Should Know As A Beginner On previous post we discussed how to install Linux now we are going to talk about the most powerful thing in Linux which is Linux commands or shell commands for the whole documentation of Linux Commands, you can check Linux Documentation if you will use Linux […]

**0**

How to write practical shell script

Linux process management simplified

**0 Comments** **likegeeks** 1 **Login**

**Recommend** **Share** **Sort by Best**

Start the discussion…

Be the first to comment.

**Subscribe** **Add Disqus to your site Add Disqus Add** **Privacy**

## SEARCH

Search … Search