# 1.Overview

|  |
| --- |
| **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 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 |

# 2. Table of Contents

* [**1 Expect Command**](https://likegeeks.com/expect-command/#Expect-Command)
* [**2 Using autoexpect**](https://likegeeks.com/expect-command/#Using-autoexpect)
* [**3 Working with Variables**](https://likegeeks.com/expect-command/#Working-with-Variables)
* [**4 Conditional Tests**](https://likegeeks.com/expect-command/#Conditional-Tests)
* [**5 If else Conditions**](https://likegeeks.com/expect-command/#If-else-Conditions)
* [**6 While Loops**](https://likegeeks.com/expect-command/#While-Loops)
* [**7 For Loops**](https://likegeeks.com/expect-command/#For-Loops)
* [**8 User-defined Functions**](https://likegeeks.com/expect-command/#User-defined-Functions)
* [**9 Interact Command**](https://likegeeks.com/expect-command/#Interact-Command)

## 2.1. 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.  set                Allowing you in interact with your program.   * The spawn command is used to *start a script* or a program like the shell, [**FTP**](https://likegeeks.com/ftp-server-linux/), 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*. |

### 2.1.1.Example

|  |  |  |  |
| --- | --- | --- | --- |
| 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: test.sh   |  | | --- | | #!/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: expect.exp   |  | | --- | | #!/usr/bin/expect -f    set timeout -1  spawn /home/dath/test/test.sh    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 command. * We can use spawn to run any program we want or any other interactive script. * The remaining send lines are the Expect script that interacts with our shell script. * The last line if the end of file which means the end of the interaction.   Run the expect script:   |  | | --- | | [root@TESTBED-VOD-CMS test]# ./expect.exp  spawn /home/dath/test/test.sh  Hello, who are you?  I'm Adam  Can I ask you some questions?  Sure  What is your favorite topic?  Technology | |

### 2.1.2 Set Comand \_ Working with variables

|  |  |  |
| --- | --- | --- |
| Working with Variables The set command is used to define variables in Expect scripts like this:  set MYVAR 5  To access the variable, precede it with $ like this $VAR1  To define 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 topic **[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 |   Run the expect script:   |  | | --- | | [root@TESTBED-VOD-CMS test]# ./expect.exp Dat film  spawn /home/dath/test/test.sh  Hello, who are you?  I'm Dat  Can I ask you some questions?  Sure  What is your favorite topic?  film | |

### 2.1.3. Expect command\_ Conditional Tests

|  |  |  |  |
| --- | --- | --- | --- |
| You can write conditional tests 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: test2.sh:  $RANDOM: A random number is generated ever time you run the script and based on that number, we put a condition to return different expects.   |  | | --- | | #!/bin/bash  let number=$RANDOM  if [ $number -gt 25000 ]  then  echo "What is your favorite topic?"  else  echo "What is your favorite movie?"  fi  read REPLY |   The expect2.exp expect script:   |  | | --- | | #!/usr/bin/expect –f  set timeout -1  spawn /home/dath/test/test2.sh  expect {  "\*topic?" {send -- "film\r"}  "\*movie?" {send -- "sassy\r"}  }  expect eof |   Chmod the bash script and expect script, then run the expect2.exp script:   |  | | --- | | [root@TESTBED-VOD-CMS test]# ./expect2.exp  spawn /home/dath/test/test2.sh  What is your favorite movie?  sassy  [root@TESTBED-VOD-CMS test]# ./expect2.exp  spawn /home/dath/test/test2.sh  What is your favorite topic?  film | |

### 2.1.4 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 you 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.  This shell script will ask about the password as shown: test3.sh   |  | | --- | | #!/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: expect3.exp   |  | | --- | | #!/usr/bin/expect -f  set timeout -1  spawn /home/dath/test/test3.sh  expect "Hello, who are you?\r"  send -- "Im Adam\r"  expect "\*password?\r"  interact ++ return  send "\r"  expect "What is your favorite topic?\r"  send -- "Technology\r"  expect eof |   Run the expect script:   |  | | --- | | [root@TESTBED-VOD-CMS test]# ./expect3.exp  spawn /home/dath/test/test3.sh  Hello, who are you?  Im Adam  What is your password?  abs =>\*\*\* After type abs must type ++ => will return back to the automatically script  What is your favorite topic?  Technology |   After you type your password and 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. |