Automatically enter input in command line

Asked 7 years, 3 months ago Active 1 year, 5 months ago Viewed 440k times

183

I'm running a script that it requests entering 'y' on each operation, I am looking for a solution like `$ ./script < echo 'yyyyyyyyyyyyy'` to pass all my inputs in one time.

command-line input

74

asked Aug 29 '13 at 12:05

NewMrd
2,107 4 12 9

Sometimes `-f` option works well with certain commands. – iammilind Jun 24 '16 at 10:00

7 Answers

Active | Oldest | Votes

315

There is a command created specifically for that case: `yes`

```
$ yes | ./script
```

followed by a newline. So basically as if the user is entering `y` for every question of `./script`.

If you want to say no (`n`) instead of yes (`y`) you can do it like this:

```
$ yes n | ./script
```

Note that some tools have an option to always asume `yes` as answer. See here for example: Bypass the yes/no prompt in 'apt-get upgrade'

Other methods to enter input:

If you know exactly how many `y` your script is expecting you can do it like this:

```
$ printf 'y\ny\ny\n' | ./script
```

The newlines (`\n`) are the enter keys.

Using `printf` instead of `yes` you have more fine grained control of input:

```
$ printf 'yes\nno\nmaybe\n' | ./script
```

Note that in some rare cases the command does not require the user to press enter after the character. in that case leave the newlines out:

```
$ printf 'yyy' | ./script
```

For sake of completeness you can also use a here document:

```
$ ./script << EOF
y
y
y
EOF
```

Or if your shell supports it a here string:

```
$ ./script <<< "y
y
y
"
```

Or you can create a file with one input per line:

If the command is sufficiently complex and the methods above no longer suffice then you can use expect.

Here is an example of a super simple expect script:

```
spawn ./script
expect "are you sure?"
send "yes\r"
expect "are you really sure?"
send "YES!\r"
expect eof
```

Technical nitpick:

The hypothetical command invocation you gave in your question does not work:

```
$ ./script < echo 'yyyyyyyyyyyyyy'
bash: echo: No such file or directory
```

This is because the shell grammar allows a redirect operator anywhere in the command line. As far as the shell is concerned your hypothetical command line is the same as this line:

```
$ ./script 'yyyyyyyyyyyyyy' < echo
bash: echo: No such file or directory
```

That means `./script` will be called with the argument `'yyyyyyyyyyyyyy'` and the stdin will get input from a file named `echo`. And bash complains since the file does not exists.

edited Jul 22 '19 at 5:40

answered Aug 29 '13 at 12:08

lesmana

15.9k 8 49 49

2 I get a `cannot enable tty mode on non tty input`. Would you know a workaround for that? – bmpasini Apr 22 '16 at 19:33

When I try the `printf` trick with a `run` file that I need to automate the installation process of, all that happens is I get an error message saying `Warning: Tried to connect to session manager, None of the authentication protocols specified are supported`, and the script opens in a new terminal and asks me to enter my input manually as usual. This is happening on Debian by the way. Any suggestions? – DaveTheMinion Jul 11 '17 at 18:45

need more details. too big for comments. please ask a new question with all the details. you can put a link here. – lesmana Jul 11 '17 at 19:55

nothing works with 'vagrant destroy' on centos – Drew Sep 4 '17 at 18:06

12

Use the command `yes` :

```
yes | script
```

Excerpt from the man page:

```
NAME
       yes - output a string repeatedly until killed

SYNOPSIS
       yes [STRING]...
       yes OPTION

DESCRIPTION
       Repeatedly output a line with all specified STRING(s), or 'y'.
```

edited Apr 11 '14 at 15:47
Andrea Corbellini
14k 2 58 79

answered Apr 11 '14 at 14:40
Peter W. Osel
121 1 2

9

Some things (`apt-get` for example) accept special flags to run in silent mode (and accept defaults). In `apt-get` 's case, you just pass it a `-y` flag. This does completely depend on your script though.

If you need more complicated things, you can wrap your script in an expect script. expect allows you to read output and send input so you can do pretty complicated things that other scripting wouldn't allow. Here's one of the examples from its Wikipedia page:

```
# Assume $remote_server, $my_user_id, $my_password, and $my_command were read in
earlier
# in the script.
# Open a telnet session to a remote server, and wait for a username prompt.
spawn telnet $remote_server
expect "username:"
# Send the username, and then wait for a password prompt.
send "$my_user_id\r"
expect "password:"
# Send the password, and then wait for a shell prompt.
send "$my_password\r"
expect "%"
# Send the prebuilt command, and then wait for another shell prompt.
send "$my_command\r"
expect "%"
# Capture the results of the command into a variable. This can be displayed, or
written to disk.
set results $expect_out(buffer)
# Exit the telnet session, and wait for a special end-of-file character.
send "exit\r"
```

answered Aug 29 '13 at 12:12

Oli ♦
263k 104 627 802

This can't be used it with a `.sh` shell script, right? Or is there a way? – Shayan Aug 29 '19 at 15:50

7

In the shell script you can also use the following trick of spawn, expect and send

```
spawn script.sh
expect "Are you sure you want to continue connecting (yes/no)?"
send "yes"
```

However in the above scenerio you will have to give the phrase you are expecting to get while you execute the script for more examples go to the following link

Expect within Bash

edited May 23 '17 at 12:39

Community ♦
1

answered Aug 29 '13 at 12:35

Tarun
4,041 11 42 70

2

Okay, this may not be a very elegant solution but if you write your options in a separate file and then pass it as an input to the script, it would work as well. So if you create a new file with all your options (call this file as 'options.in'), then you can easily run your script with `./script.sh < options.in` and edit/create different options files as suitable.

edited Aug 29 '13 at 15:57

answered Aug 29 '13 at 15:50

Sidhha
121 3

How should I write the `options.in` file? Can you give an example? – Shayan Aug 29 '19 at 17:08

1

I was writing a bash script with Dialog and needed this to happen automatically also. I did this and it worked like a charm.

```
# -Wy force signaturewipe (if exists)
echo "y" | sudo lvcreate -W y -n $lvName -L "$lvSize"G /dev/"$svg" >> $nfsUtilLog
```

edited Apr 23 '18 at 12:38

answered Aug 3 '17 at 16:05

1

You can supply user input to your script with `cat` , from a text file, piped to your script with `bash` like this:

```
cat input.txt | bash your_script.sh
```

Just put your desired user input into your input.txt file, whatever answers you want - y, n, digits, strings, etc.

edited Jul 5 '18 at 15:37
SudoSURoot
2,579 1 11 16

answered Jul 5 '18 at 9:46
WebComer
187 1 6

Aka askubuntu.com/a/338955/158442 – muru Jul 5 '18 at 9:51

@muru For me, the solution you've mentioned didn't work, so i have to invent my own. – WebComer Jul 5 '18 at 9:53

Unlikely. Both of you describe the same thing. – muru Jul 5 '18 at 9:55

@muru Alike idea of supply answers from file (which can be done several ways) is not the same thing, and moreover it accomplished quite a different way. You should be able to see it. If the solution you point out would work for me, i won't spent my time reinventing the wheel today. You may use whatever work for you. – WebComer Jul 5 '18 at 10:35

@muru I believe there is a difference between the two, some of which described in answers say here askubuntu.com/questions/172982/…. For me, for example, my code if output is redirected to file seems more clear as a natural flow from left to right. After all, downvoting working well solutions like you doesn't stimulate people to share their solutions here. I believe that said is enough to remove downvoting. – WebComer Jul 5 '18 at 16:38