

```
#!/usr/bin/expect -f
#
#
# Copyright (C) 2017 Christopher Fedun
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program. If not, see <http://www.gnu.org/licenses/>.
#
# This Expect script was generated by autoexpect on Wed Mar 15 21:31:12 2017
# Expect and autoexpect were both written by Don Libes, NIST.
#
# Note that autoexpect does not guarantee a working script. It
# necessarily has to guess about certain things. Two reasons a script
# might fail are:
#
# 1) timing - A surprising number of programs (rn, ksh, zsh, telnet,
# etc.) and devices discard or ignore keystrokes that arrive "too
# quickly" after prompts. If you find your new script hanging up at
# one spot, try adding a short sleep just before the previous send.
# Setting "force_conservative" to 1 (see below) makes Expect do this
# automatically - pausing briefly before sending each character. This
# pacifies every program I know of. The -c flag makes the script do
# this in the first place. The -C flag allows you to define a
# character to toggle this mode off and on.

set force_conservative 0 ;# set to 1 to force conservative mode even if
                          ;# script wasn't run conservatively originally
if {$force_conservative} {
    set send_slow {1 .1}
    proc send {ignore arg} {
        sleep .1
        exp_send -s -- $arg
    }
}

set root_db_pass [lindex $argv 0]
#
# 2) differing output - Some programs produce different output each time
# they run. The "date" command is an obvious example. Another is
# ftp, if it produces throughput statistics at the end of a file
# transfer. If this causes a problem, delete these patterns or replace
# them with wildcards. An alternative is to use the -p flag (for
# "prompt") which makes Expect only look for the last line of output
# (i.e., the prompt). The -P flag allows you to define a character to
# toggle this mode off and on.
#
# Read the man page for more info.
#
# -Don

stty columns 177
stty rows 57
set timeout -1
spawn mysql_secure_installation
match_max 100000
expect -exact "Enter current password for root (enter for none): "
```

```
send -- "$root_db_pass\r"
expect -exact "Change the root password? \[Y/n\] "
send -- "n\r"
expect -exact "Remove anonymous users? \[Y/n\] "
send -- "y\r"
expect -exact "Disallow root login remotely? \[Y/n\] "
send -- "y\r"
expect -exact "Remove test database and access to it? \[Y/n\] "
send -- "y\r"
expect -exact "Reload privilege tables now? \[Y/n\] "
send -- "y\r"
expect eof
```