NAME

runscript − script interpreter for minicom

SYNOPSIS

runscript scriptname [logfile [homedir]]

DESCRIPTION

runscript is a simple script interpreter that can be called from within the minicom communications program to automate tasks like logging in to a unix system or your favorite bbs.

INVOCATION

The program expects a script name and optionally a filename and the user’s home directory as arguments, and it expects that it’s input and output are connected to the "remote end", the system you are connecting to. All messages from **runscript** ment for the local screen are directed to the **stderr** output. All this is automatically taken care of if you run it from **minicom**. The logfile and home directory parameters are only used to tell the log command the name of the logfile and where to write it. If the homedir is omitted, runscript uses the directory found in the $HOME environment variable. If also the logfile name is omitted, the log commands are ignored.

KEYWORDS

Runscript recognizes the following commands:

```
expect   send     goto   gosub   return  !
exit     print    set    inc     dec     if
timeout  verbose  sleep  break   call    log
```

OVERVIEW OF KEYWORDS

send <string>

<string> is sent to the modem. It is followed by a ’\r’. <string> can be:
- regular text, eg ’send hello’
- text enclosed in quotes, eg ’send "hello world"’

Within <string> the following sequences are recognized:

\n - newline
\r - carriage return
\a - bell
\b - backspace
\c - don’t send the default ’\r’.
\f - formfeed
\^ - the ^ character
\o - send character **o** (**o** is an octal number)

Control characters can be used in the string with the ^ prefix (^A to ^Z, ^[, ^ ^], ^^ and ^_). If you need to send the ^ chracter, you must prefix it with the \ escape character.
Also $(environment_variable) can be used, for example $(TERM). Minicom passes three special environment variables: $(LOGIN), which is the username, $(PASS), which is the password, as defined in the proper entry of the dialing directory, and $(TERMLIN) which is the number of actual terminal lines on your screen (that is, the statusline excluded).

print <string>

Prints <string> to the local screen. Default followed by ’\r\n’. See the description of ’send’ above.

label: Declares a label (with the name ’label’) to use with goto or gosub.

goto <label>

Jump to another place in the program.

gosub <label>
Jumps to another place in the program. When the statement 'return' is encountered, control returns to the statement after the gosub. Gosub's can be nested.

return Return from a gosub.

! <command>
Runs a shell for you in which 'command' is executed. On return, the variable '$?' is set to the exit status of this command, so you can subsequently test it using 'if'.

exit [value]
Exit from "runscript" with an optional exit status. (default 1)

set <variable> <value>
Sets the value of <variable> (which is a single letter a-z) to the value <value>. If <variable> does not exist, it will be created. <value> can be a integer value or another variable.

inc <variable>
Increments the value of <variable> by one.

dec <variable>
Decrements the value of <variable> by one.

if <value> <operator> <value> <statement>
Conditional execution of <statement>. <operator> can be <, >, != or =. Eg, 'if a > 3 goto exitlabel'.

timeout <value>
Sets the global timeout. By default, 'runscript' will exit after 120 seconds. This can be changed with this command. Warning: this command acts differently within an 'expect' statement, but more about that later.

verbose <on|off>
By default, this is 'on'. That means that anything that is being read from the modem by 'runscript', gets echoed to the screen. This is so that you can see what 'runscript' is doing.

sleep <value>
Suspend execution for <value> seconds.

expect
```
 expect {
   pattern  [statement]
   pattern  [statement]
   [timeout <value> [statement] ]
   ....
 }
```
The most important command of all. Expect keeps reading from the input until it reads a pattern that matches one of the specified ones. If expect encounters an optional statement after that pattern, it will execute it. Otherwise the default is to just break out of the expect. 'pattern' is a string, just as in 'send' (see above). Normally, expect will timeout in 60 seconds and just exit, but this can be changed with the timeout command.

break Break out of an 'expect' statement. This is normally only useful as argument to 'timeout' within an expect, because the default action of timeout is to exit immediately.

call <scriptname>
Transfers control to another scriptfile. When that scriptfile finishes without errors, the original script will continue.

log <text>
Write text to the logfile.

NOTES

If you want to make your script to exit minicom (for example when you use minicom to dial up your ISP, and then start a ppp or slip session from a script), try the command "! killall -9 minicom" as the last script command. The -9 option should prevent minicom from hanging up the line and resetting the modem before exiting.

Well, I don't think this is enough information to make you an experienced 'programmer' in 'runscript', but together with the examples it shouldn't be too hard to write some useful script files. Things will be easier if you have experience with BASIC. The **minicom** source code comes together with two example scripts, **scriptdemo** and **unixlogin**. Especially the last one is a good base to build on for your own scripts.

BUGS

Runscript should be built in to minicom.

AUTHOR

Miquel van Smoorenburg, <miquels@drinkel.ow.org> Jukka Lahtinen, <walker@netsonic.fi>