Expect Automate interactive programs

CONCEPTS

Expect scripts are Tcl: spawn a program, then loop on expect (wait for text) and send (type a reply).

spawn start program on a pty
↓ output
expect match pattern
↓ action
send type the reply
↓ repeat
interact / eof hand over or finish

Send \r (Enter), not \n, when answering prompts: the program sees a terminal, and Enter is a carriage return.

Patterns are globs by default, so * matches anything. Use -re for regexps and -ex for exact text.

RUNNING

expect *<script>* **[args]**
Run a script file

expect -d *<script>*
Run with diagnostic output

#!/usr/bin/env expect
Shebang for executable scripts

autoexpect [-f *<out>***]** *<cmd>*
Record a session as a script

SPAWN AND SEND

spawn *<cmd>* **[args]**
Start a program on a pty

spawn -noecho *<cmd>*
Spawn without echoing the command line

send "*<text>***\r"**
Type text and press Enter

send -- "*<text>***"**
Send text that starts with a dash

send -i $*<id>* **"***<text>***"**
Send to a specific spawn id

send_user "*<text>***"**
Print to your terminal

set *<id>* **$spawn_id**
Save the current spawn id

wait
Reap the process, get its exit status

SPAWN AND SEND (cont.)

exit [*<status>***]**
Exit the script

sleep *<sec>*
Pause (decimals allowed)

EXPECT PATTERNS

expect "*<glob>***"**
Wait for output matching a glob

expect -re "*<regexp>***"**
Wait for a regexp match

expect -ex "*<text>***"**
Wait for exact text, no wildcards

expect eof
Wait for the process to end

expect timeout { *<cmds>* **}**
Run commands when the wait times out

expect { "a" {*<cmds>***} "b" {***<cmds>***} }**
Several patterns, first match wins

exp_continue
Keep matching after an action

expect -timeout *<sec>* **"***<pat>***"**
Timeout for this one wait

VARIABLES

set timeout *<sec>*
Default wait; -1 waits forever

$expect_out(buffer)
Output consumed by the last match

$expect_out(0,string)
Text of the whole match

$expect_out(1,string)
First -re capture group

$spawn_id
Id of the current process

$argv / $argc / $argv0
Script arguments, count, name

lindex $argv *<n>*
Get argument number n

$env(*<NAME>***)**
Read an environment variable

INTERACT

interact
Give the keyboard to the process

INTERACT (cont.)

interact { "*<key>***" { *<cmds>* } }**
Run commands when you type a string

return
Leave interact, continue the script

log_user 0
Hide the process output from you

stty -echo
Turn off echo (for passwords)

expect_user -re "(.*)\n"
Read a line typed by the user

LOGGING, DEBUG AND SIGNALS

exp_internal 1
Print match diagnostics to stderr

log_file *<file>*
Record the session to a file

TCL ESSENTIALS

set *<var>* *<value>*
Assign a variable

foreach h $hosts { ... }
Loop over a list

proc *<name>* **{***<args>***} { ... }**
Define a procedure

RECIPES

Login over ssh

```
set timeout 20
spawn ssh <user>@<host>
expect {
 "yes/no" {
  send "yes\r"
  exp_continue }
 "assword:" {
  send "$env(PW)\r" }
}
interact
```

Run one command and capture output

```
spawn ssh <host> uptime
expect -re {average: ([0-9.]+)}
puts $expect_out(1,string)
expect eof
```

Exit code of the spawned program

```
spawn <cmd>
expect eof
lassign [wait] pid id os rc
exit $rc
```