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.

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

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

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

wait
Reap the process, get its exit status

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

EXPECT PATTERNS

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

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

expect eof
Wait for the process to end

EXPECT PATTERNS (cont.)

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(1,string)
First -re capture group

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

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

INTERACT

interact
Give the keyboard to the process

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

log_user 0
Hide the process output from you

stty -echo
Turn off echo (for passwords)

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