Expect
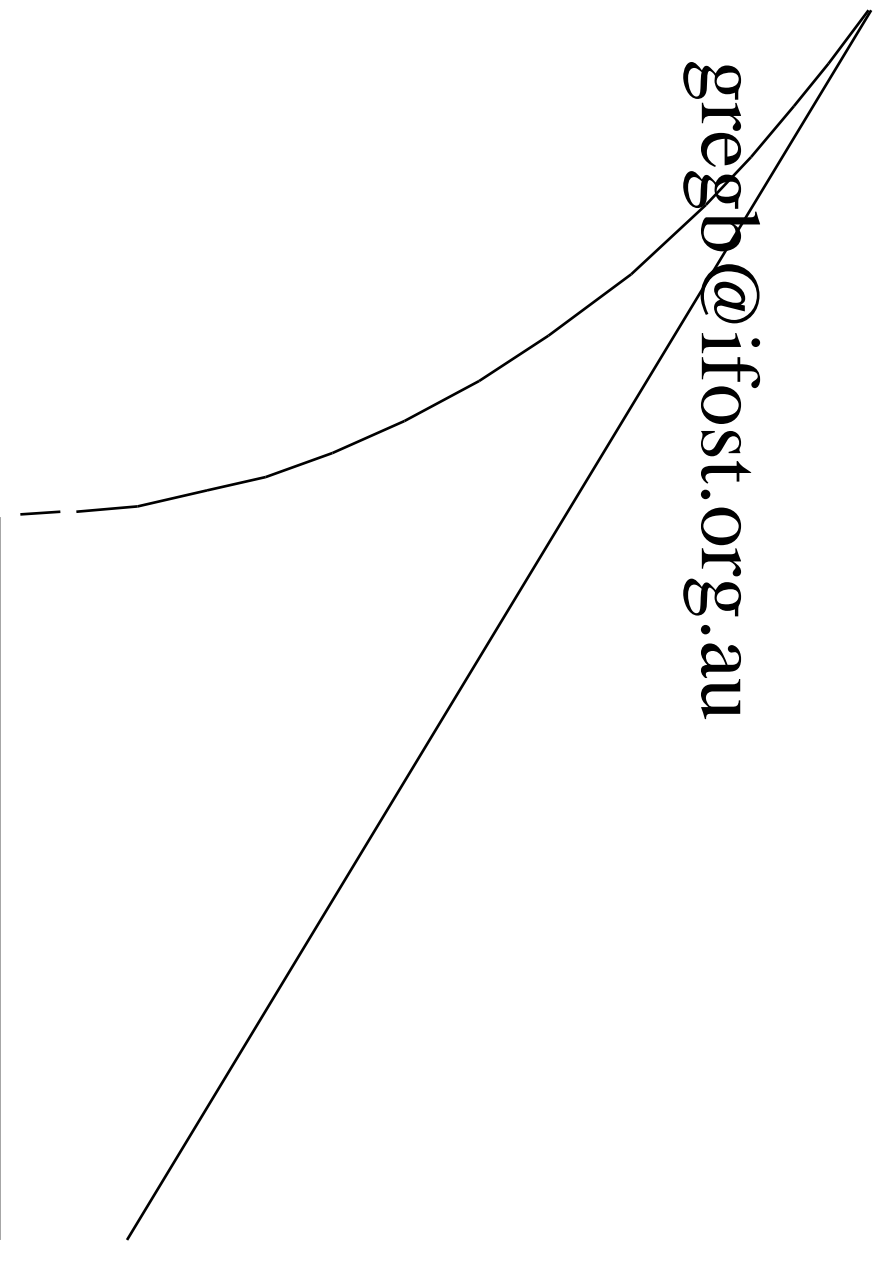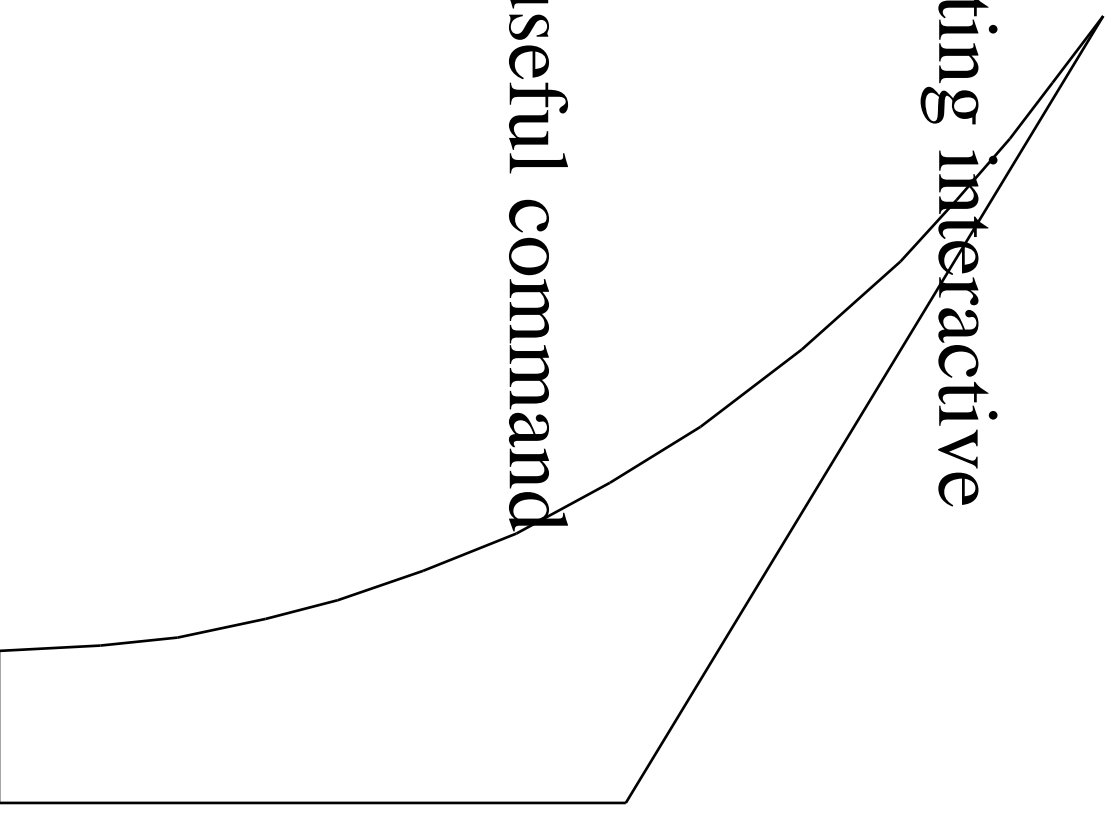
Greg Baker

gregb@ifost.org.au

What is it?
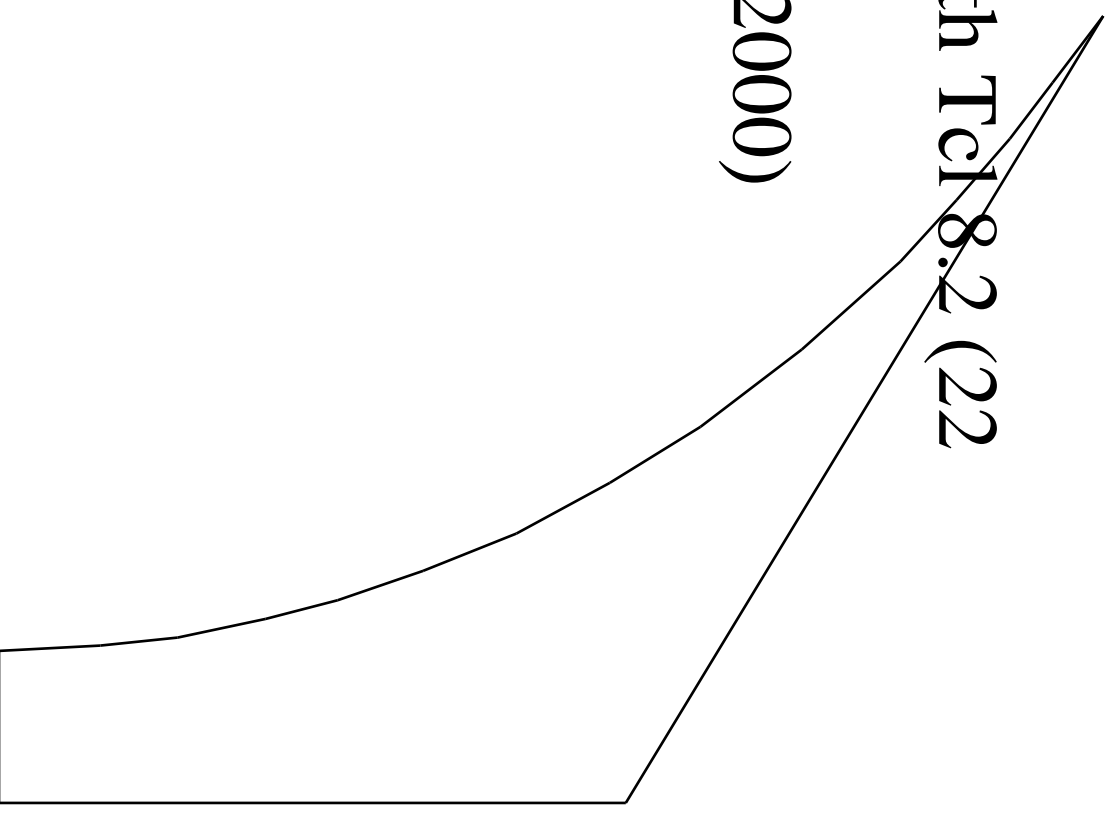
- A language for automating interactive programs
- A Tcl extension
- Public domain
- Named after its most useful command

Versions

1 - Version 5.31 works with Tcl 8.2 (22 October 1999)
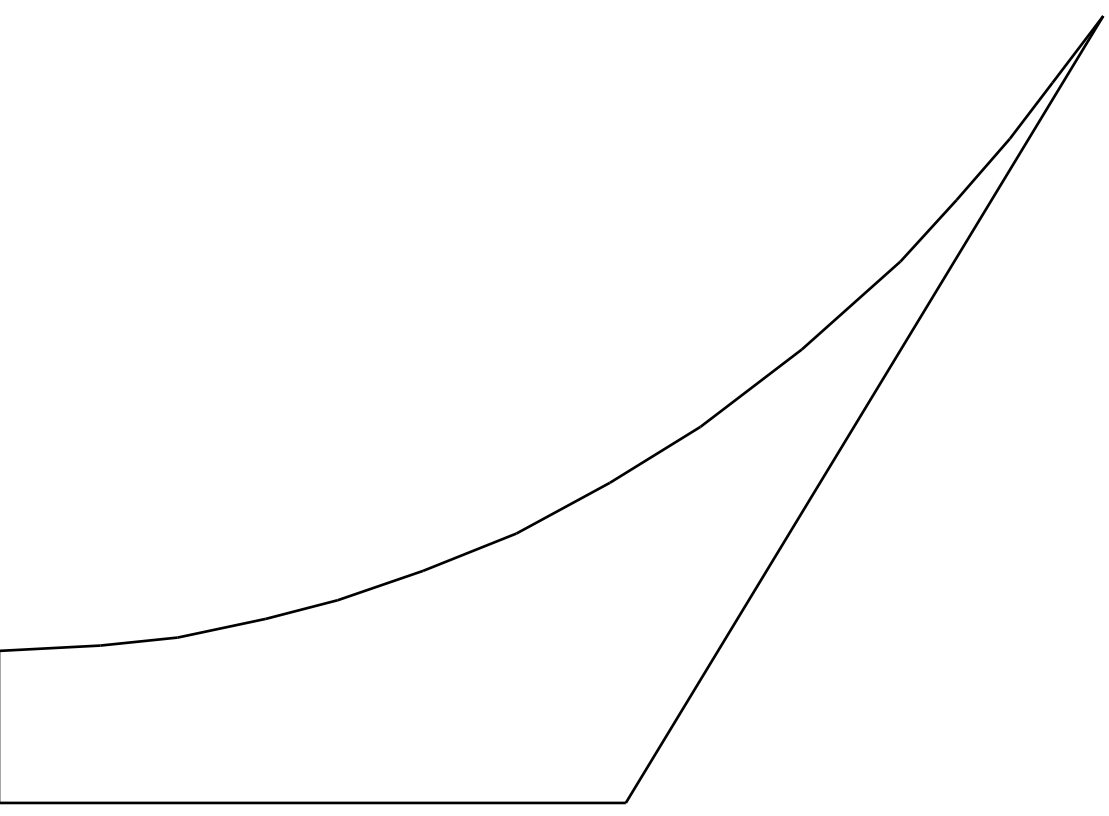
1 - Version 5.31.8 (5 Dec 2000)

Useful Expect examples

- **1** - passmass
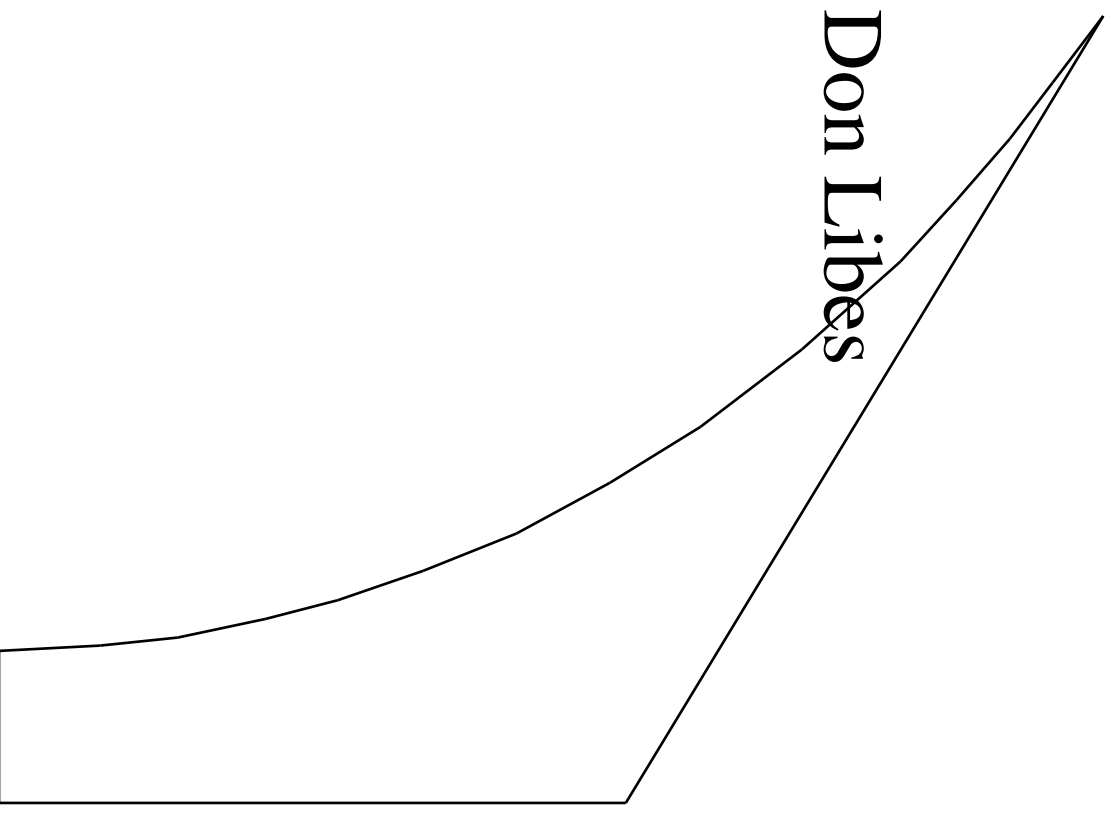- **1** - kibitz
- **1** - timed-read, timed-run
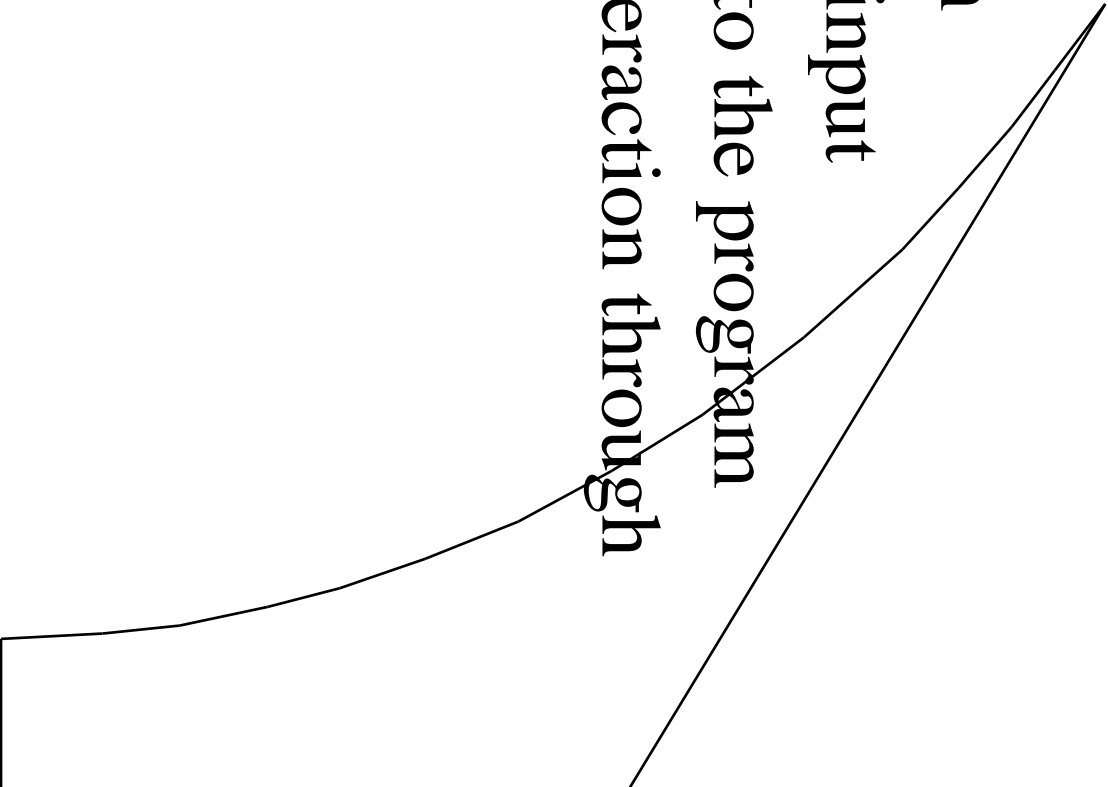- **1** - dislocate

Documentation

1 - expect(1) man page

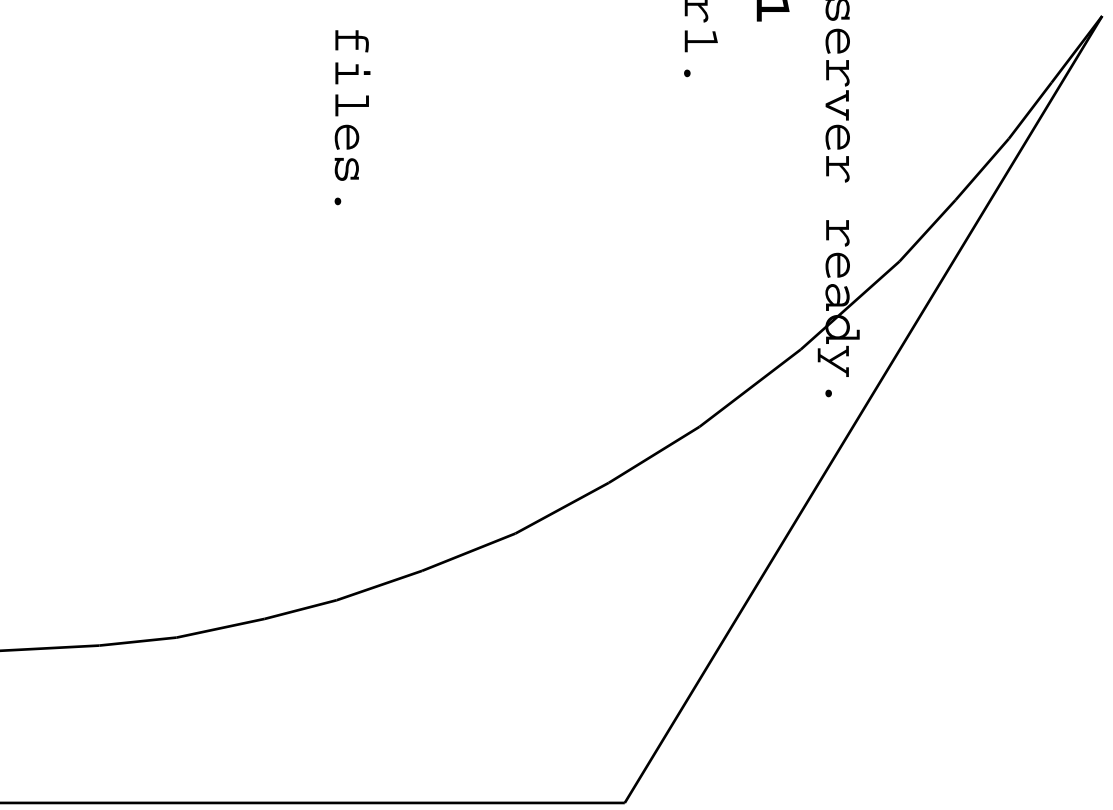1 - "Exploring Expect" by Don Libes
(O'Reilly and Associates,
ISBN 1-56592-090-2).

Four/fore most important commands

1 - [spawn] start a program
1 - [expect] wait for some input
1 - [send] send something to the program
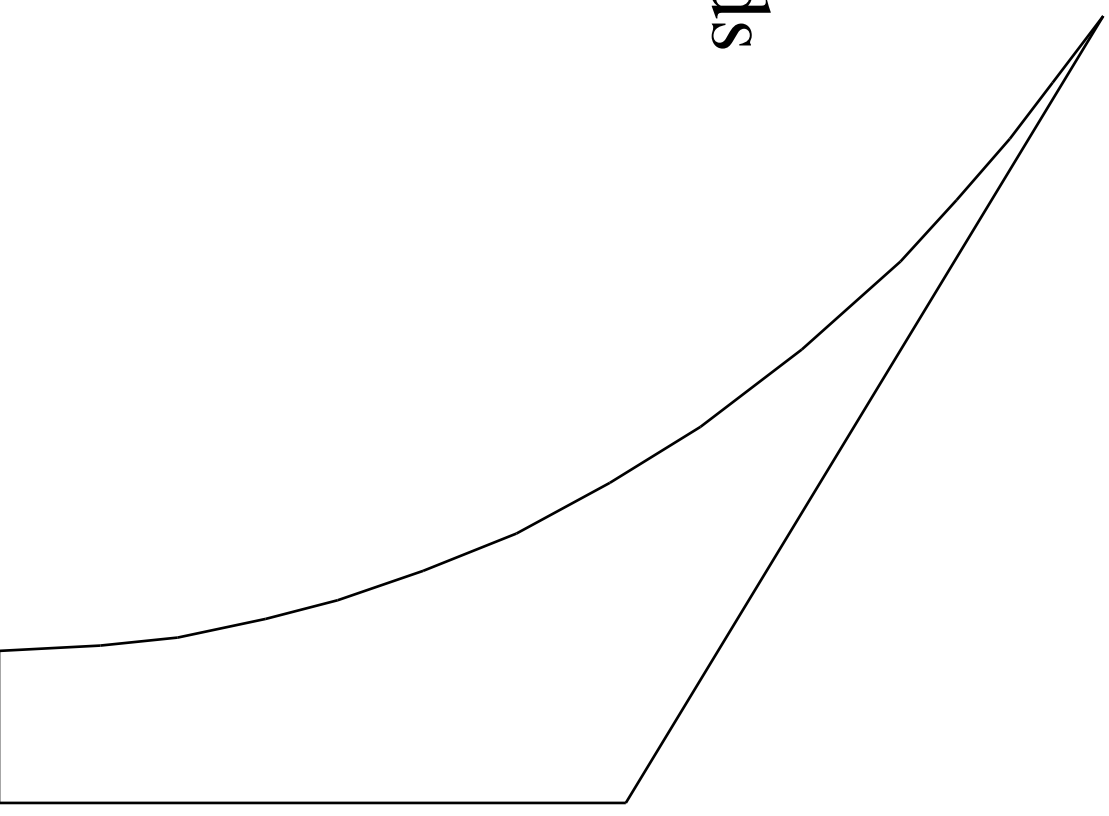1 - [interact] allow user interaction through

Automating ftp

```
ftp localhost
Connected to localhost.
220 jayanya.ifost.org.au FTP server ready.
Name (localhost:gregb): user1
331 Password required for user1.
Password:
230- jayanya ppc
230 User user1 logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> dir
```
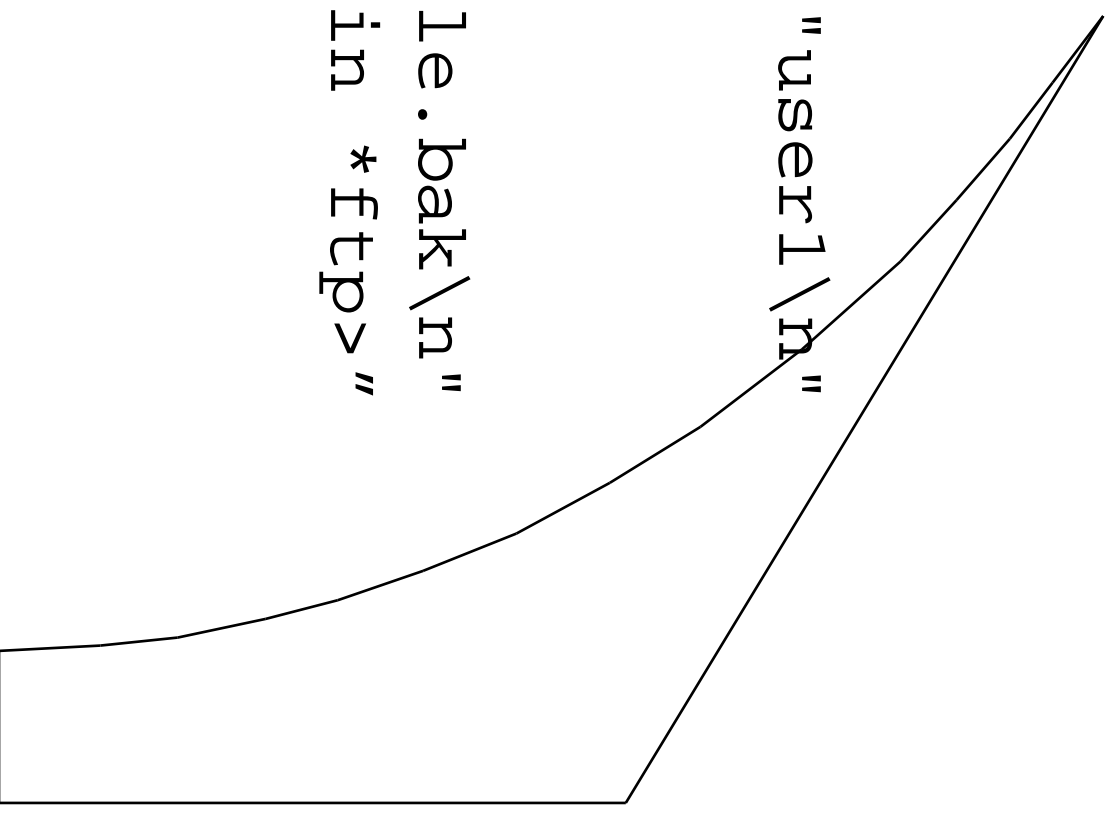
The interactions necessary are:

1 - To put in a username

1 - To put in a password
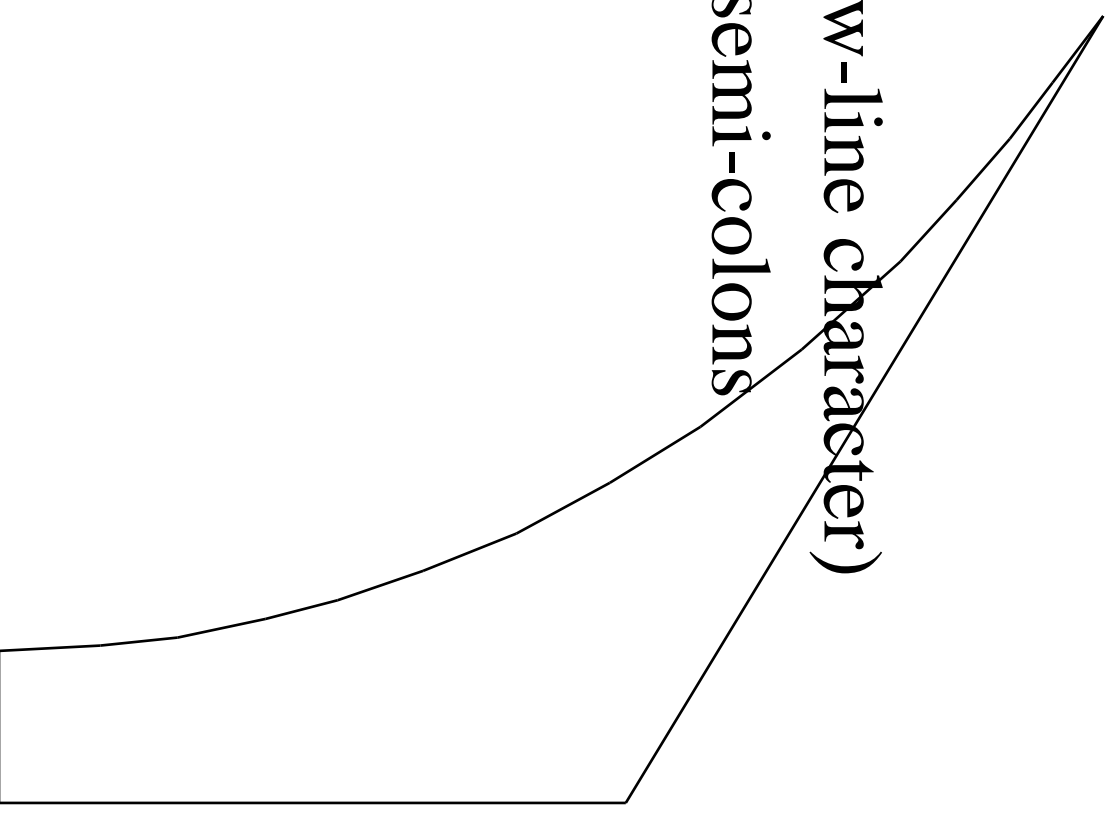
1 - To run some commands

Example Prog

```
#!/usr/bin/expect
spawn ftp localhost
expect "Name" ; send "user1\n"
expect "Password:"
send "class1\n"
expect "ftp>"
send "put myfile myfile.bak\n"
expect "* bytes sent in *ftp>"
send "bye\n"
```

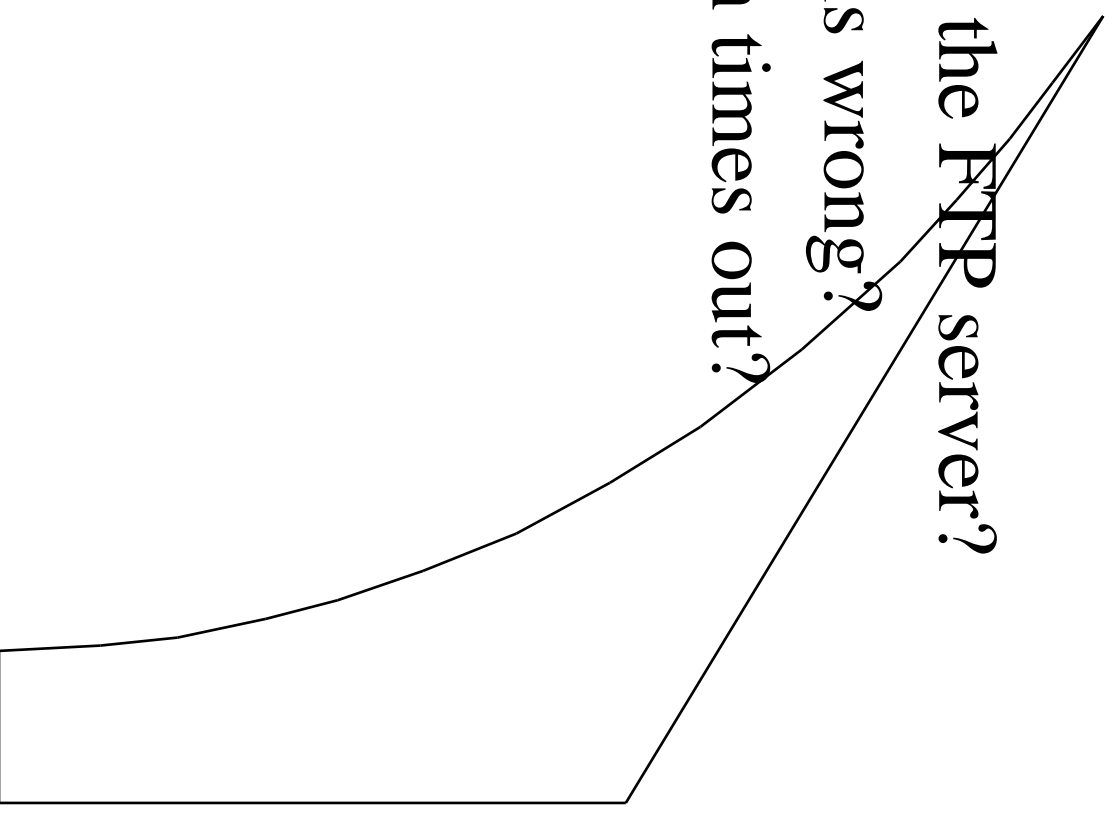# Tcl/Expect Syntax

- normally one-per-line
- (i.e. terminated by a new-line character)
- Can be separated with semi-colons

Problems

1 - What if we can't reach the FTP server?

1 - What if the password is wrong?

1 - What if the connection times out?

1 - It's a bit talkative

Multi-pattern expect

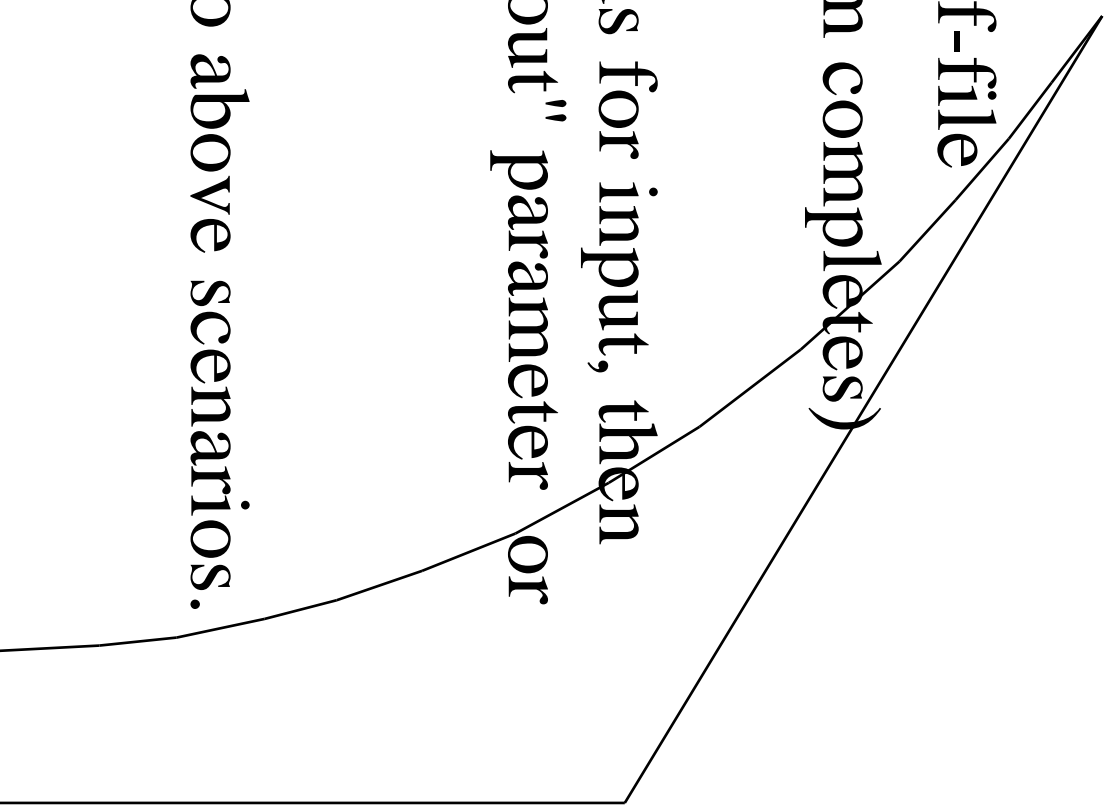expect pattern action pattern action pattern action ...

1 - Use { ... } or " ... " for actions or patterns with spaces

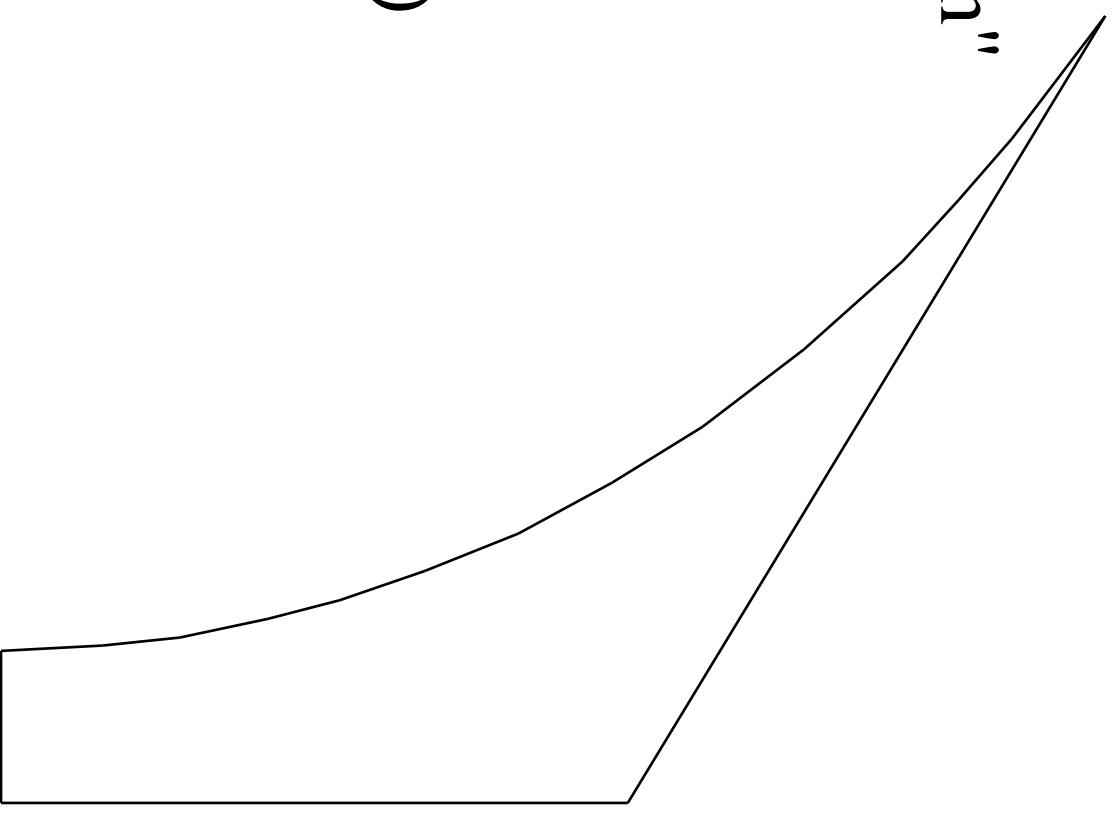**1** - If the input buffer matches a pattern, the paired action is run

The three keywords for patterns

- [eof] To be run on end-of-file (e.g. the spawned program completes)

- [timeout] Wait N seconds for input, then try this (N = the "-timeout" parameter or "timeout" variable)

- [default] Either of the two above scenarios.
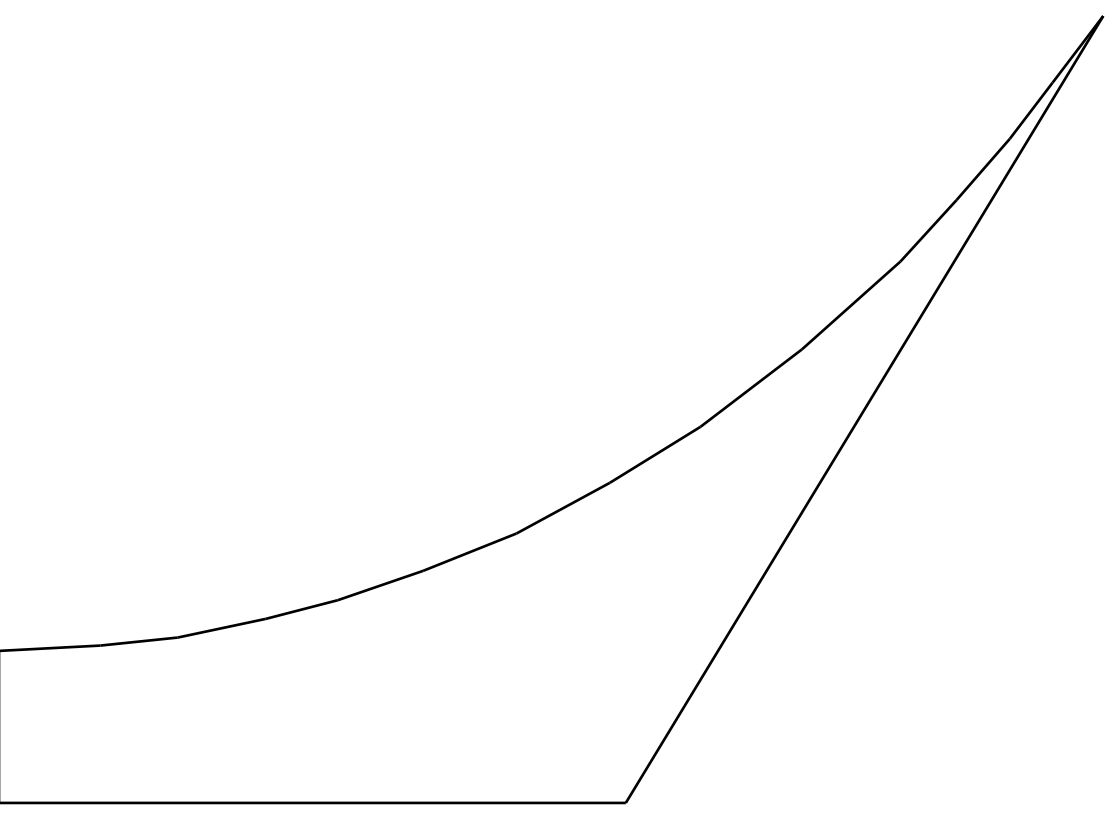
Informing the user

1 - send_user "Working!\n"
 (standard output)

1 - send_error "Failed.\n"
 (standard error)

1 - send_tty "Message\n"
 (controlling terminal)

Tcl variables

1 - set username "user1"

1 - send "$username\n"
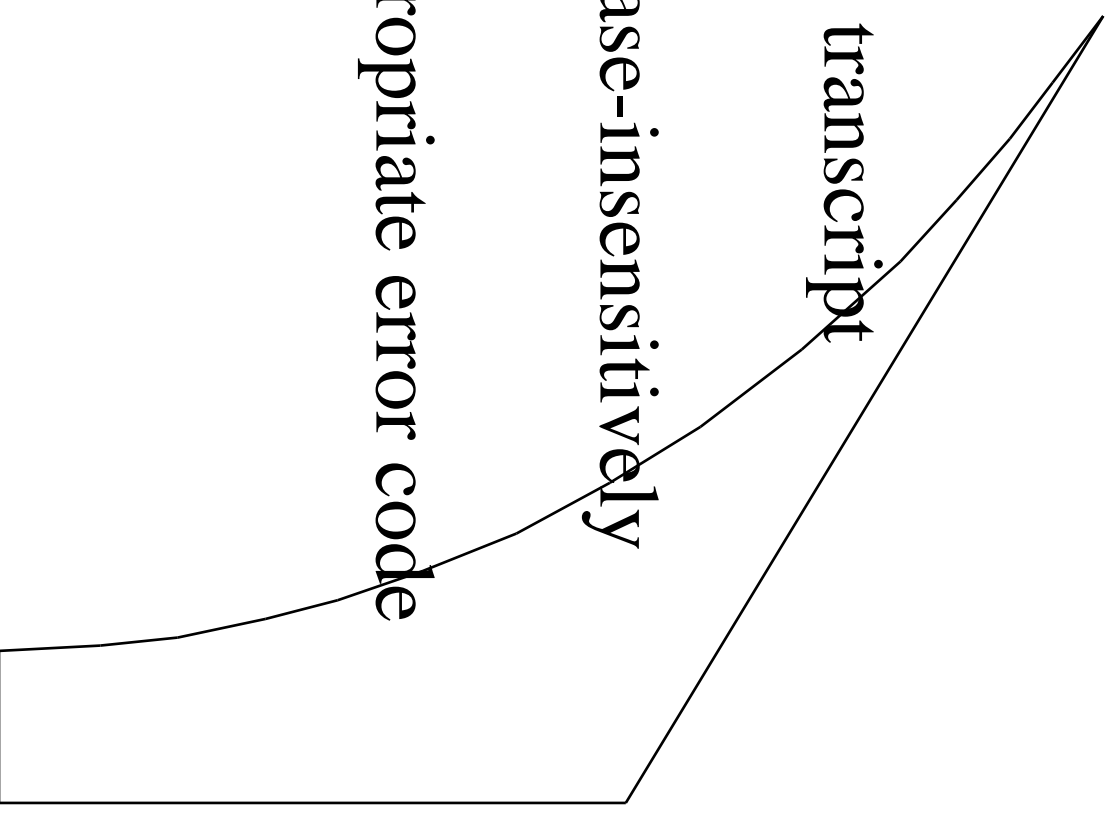
Other things

- [log_user 0]

  Don't print the session transcript

- [expect -i]
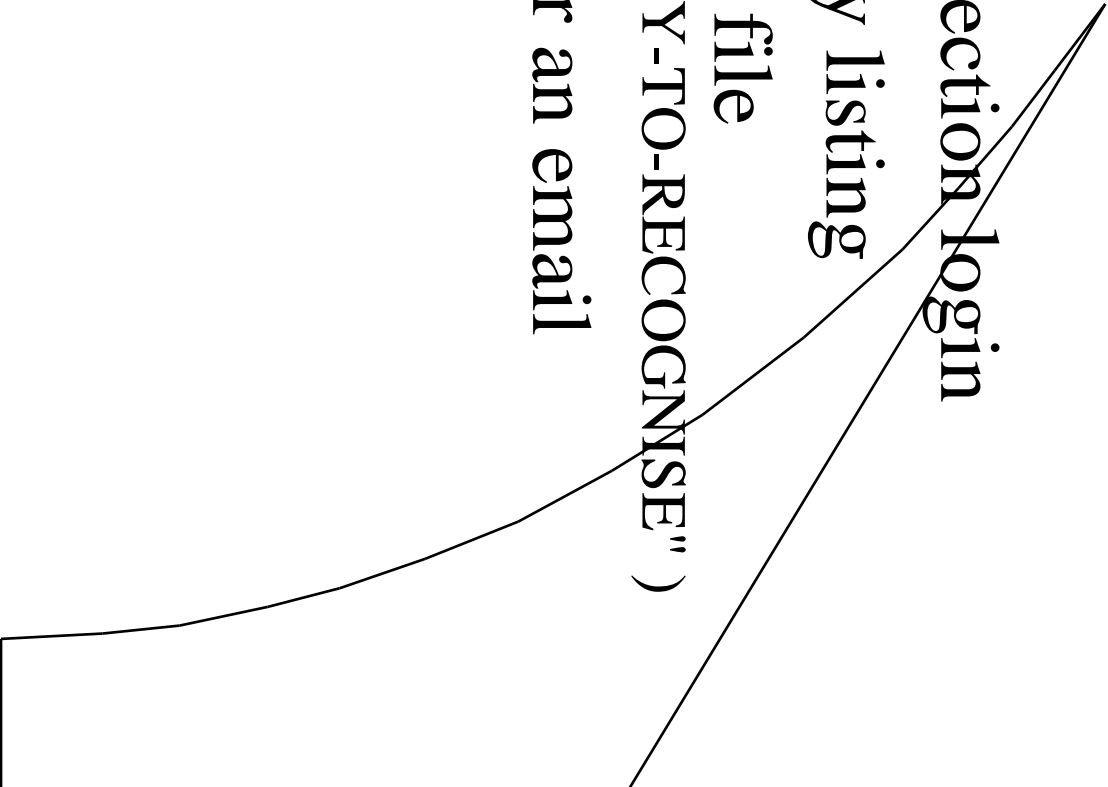
  Look for the pattern case-insensitively

- [exit]

  Exit program with appropriate error code

```
set username "user1" ; set password "class1" ; log_user 0
spawn ftp localhost
set timeout 45
expect "220*Name" { send_user "Connected!\n" } \
    timeout { send_error "Never got name\n" ; exit}
send "$username\n"
expect -i "Password"
send "$password\n"
expect "530 Login incorrect" {
    send_error "Bad password\n" ; exit 2 } \
"230*logged in*ftp>" { send_user "Password OK\n" } \
    timeout { send_error "Login timed out" ; exit 3 }
set timeout 200  ; # could take a while to send
send "put myfile myfile.bak\n"
expect "226 Transfer complete*ftp>" {
    send_user "Transfer OK\n" } \
    timeout { send_error "Transfer not done in time" }
send "bye\n"
```
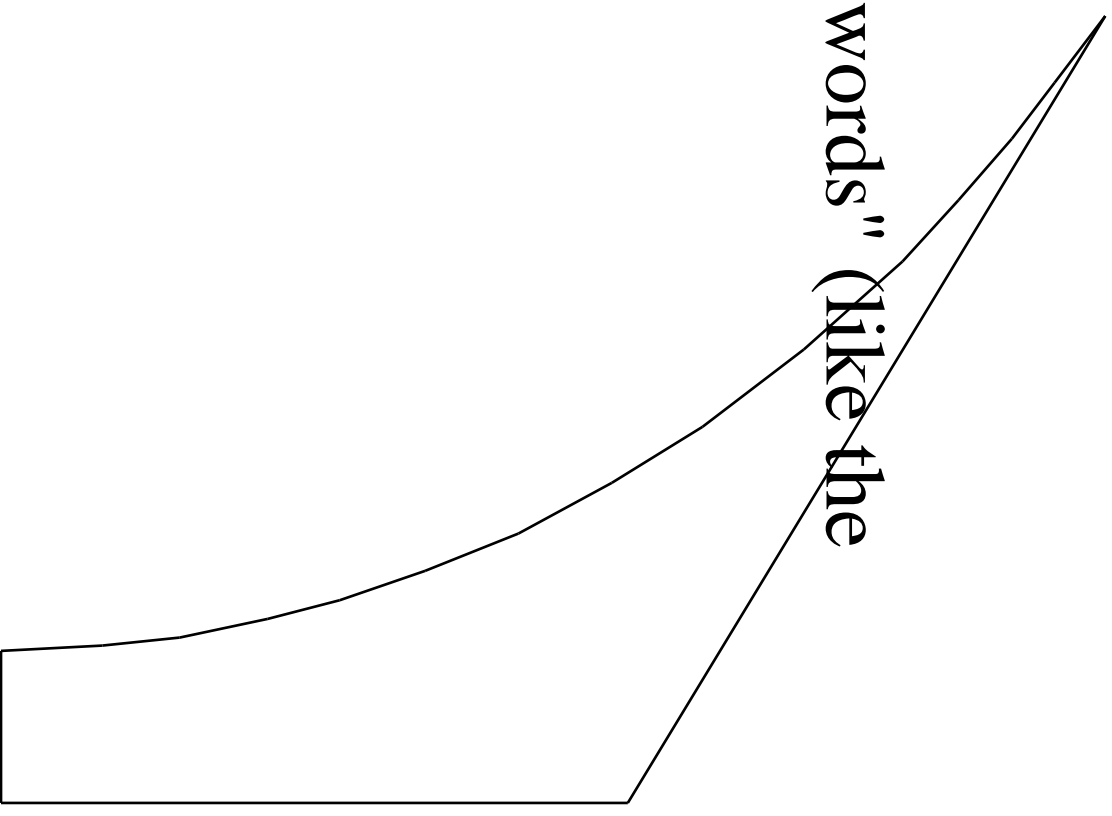
Exercise Break

1. Automate a telnet connection login

2. Use it to get a directory listing

3. Transfer it to transfer a file

   (hint: cat file ; echo "ZZZ-EASY-TO-RECOGNISE")

4. (Optional) Test whether an email address is valid

Tcl/Expect Syntax

- everything is a string
- commands break into "words" (like the Unix shell)
- five special characters:
    {...} "..." [...] $ \

{ ... }

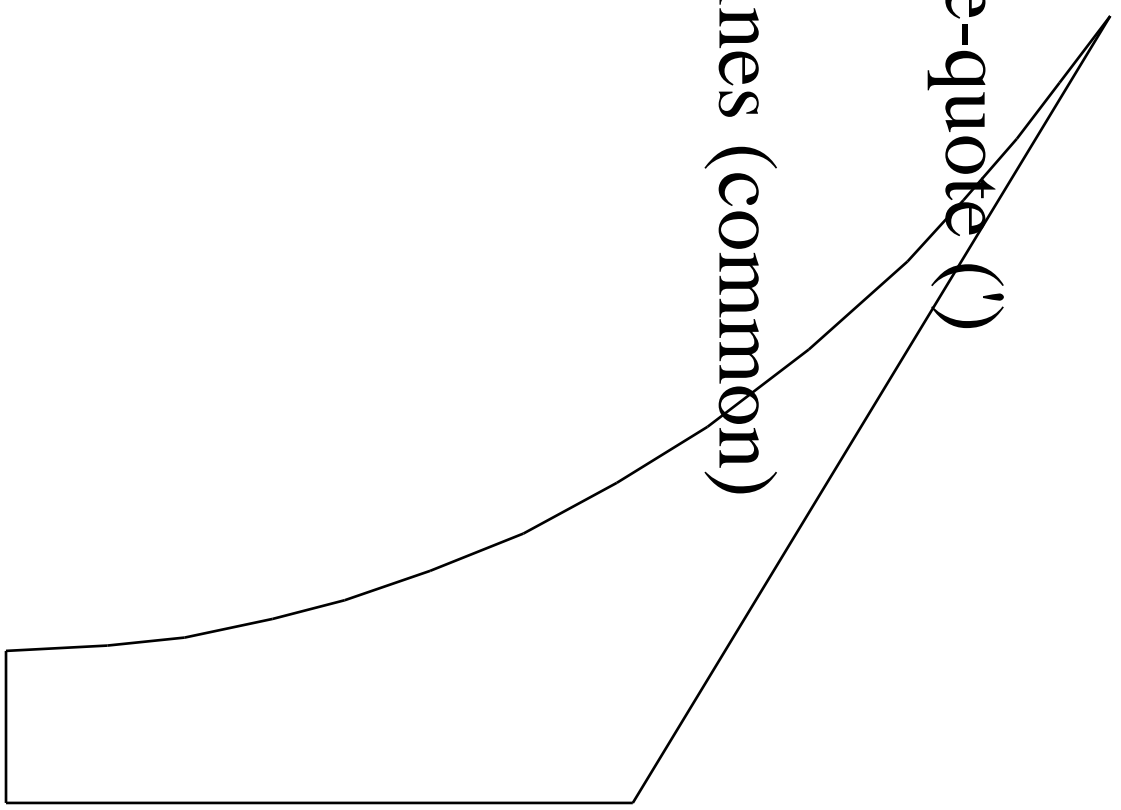' acts like Unix shell single-quote (')

' can be nested

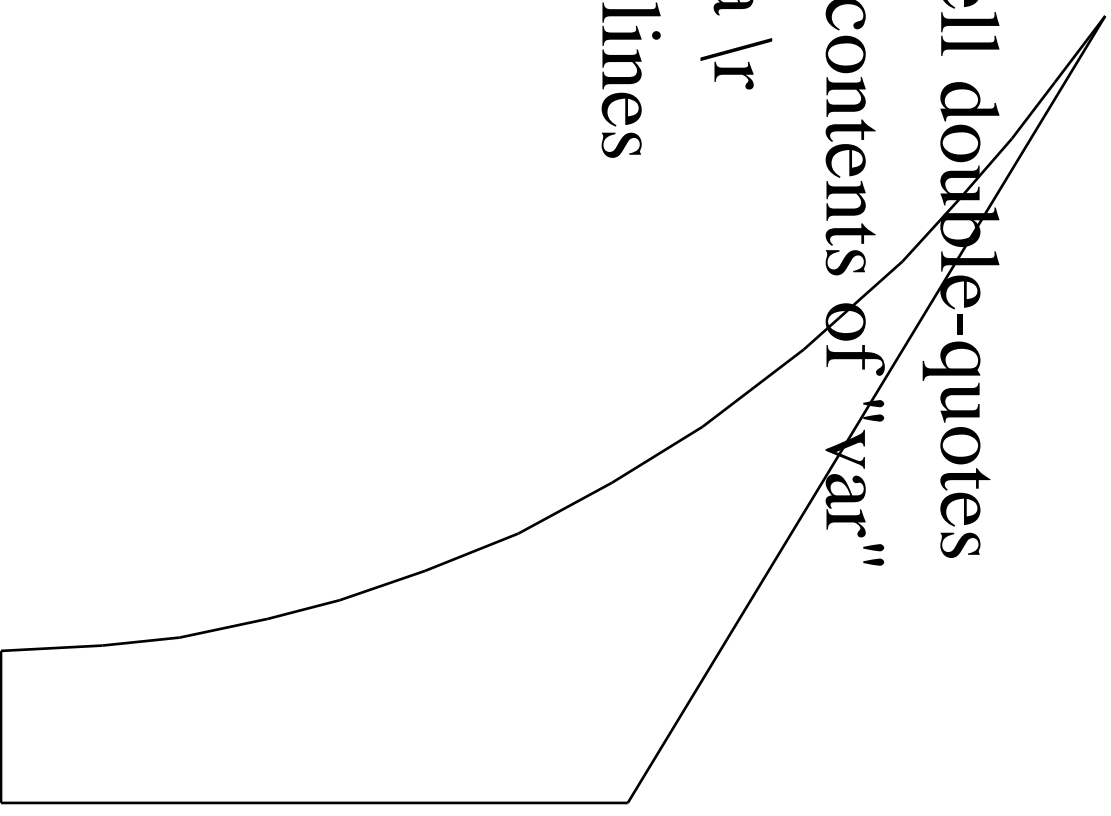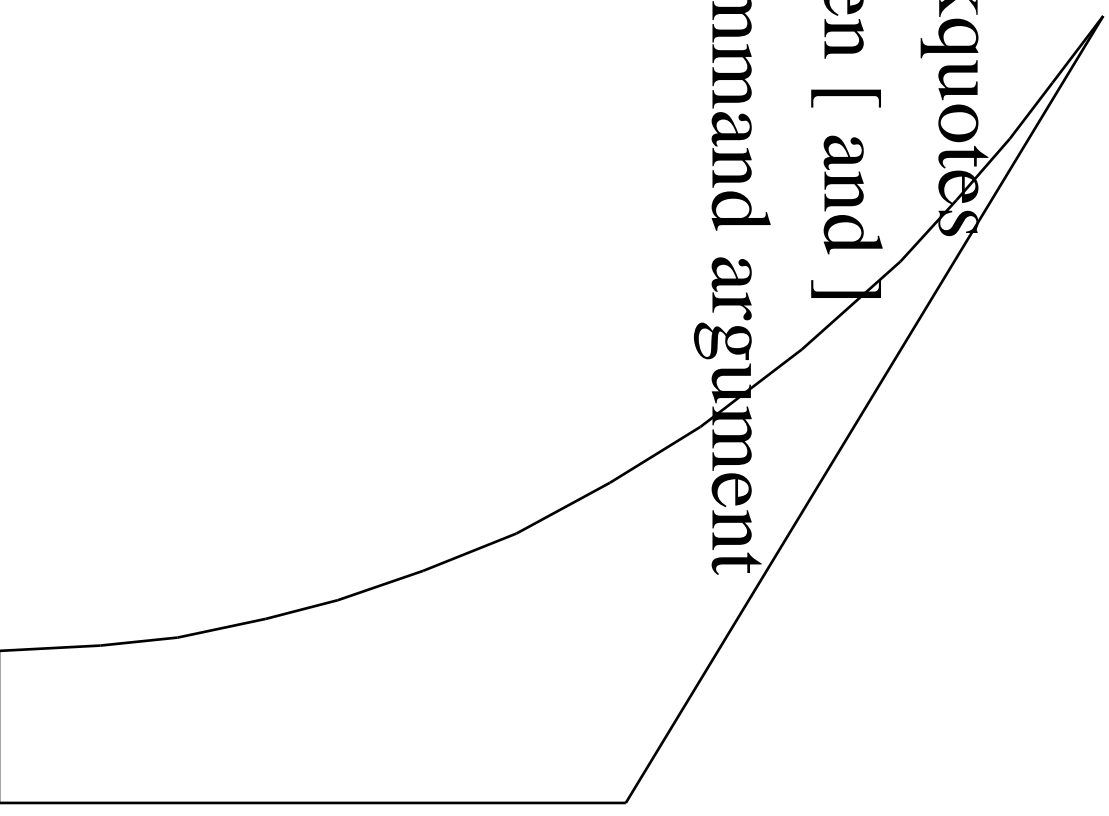' can spread over several lines (common)

" .... "

- **1** " .... " acts like Unix shell double-quotes
- **1** $var gets replaced with contents of "var"
- **1** usual string chars \n \t \a \r
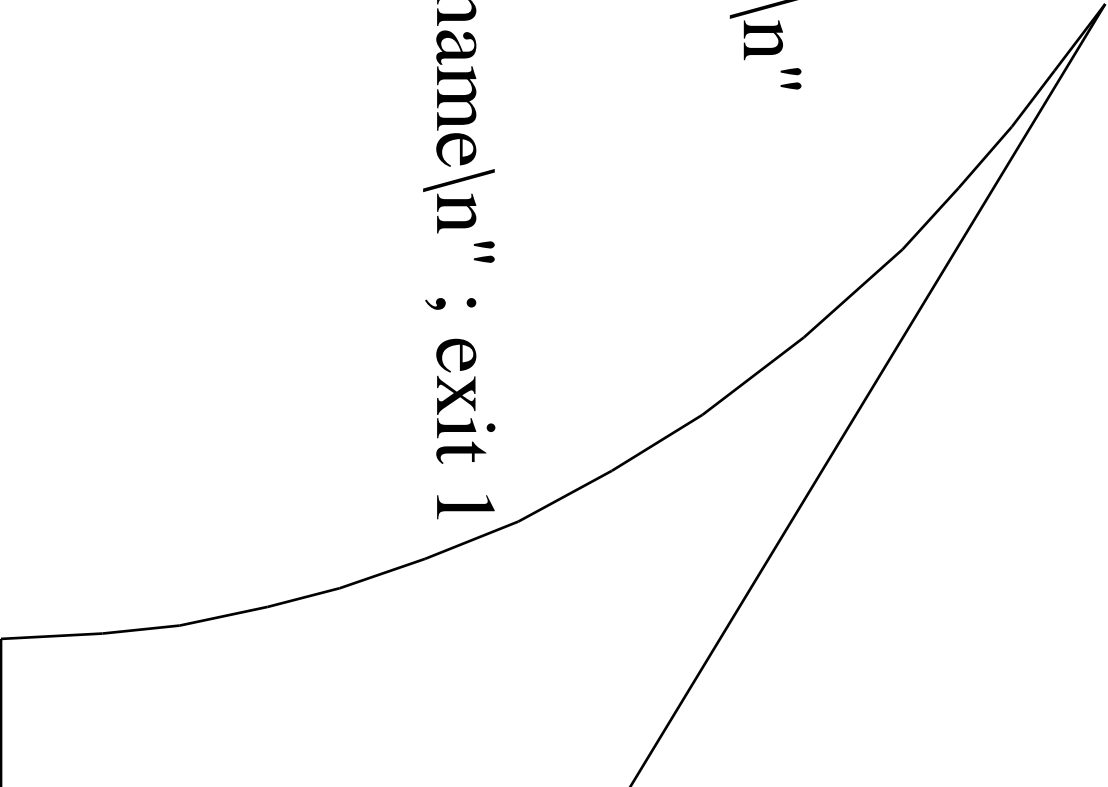- **1** can spread over several lines

[...]

1 a bit like Unix shell backquotes

1 run the command between [and]

1 command output is a command argument
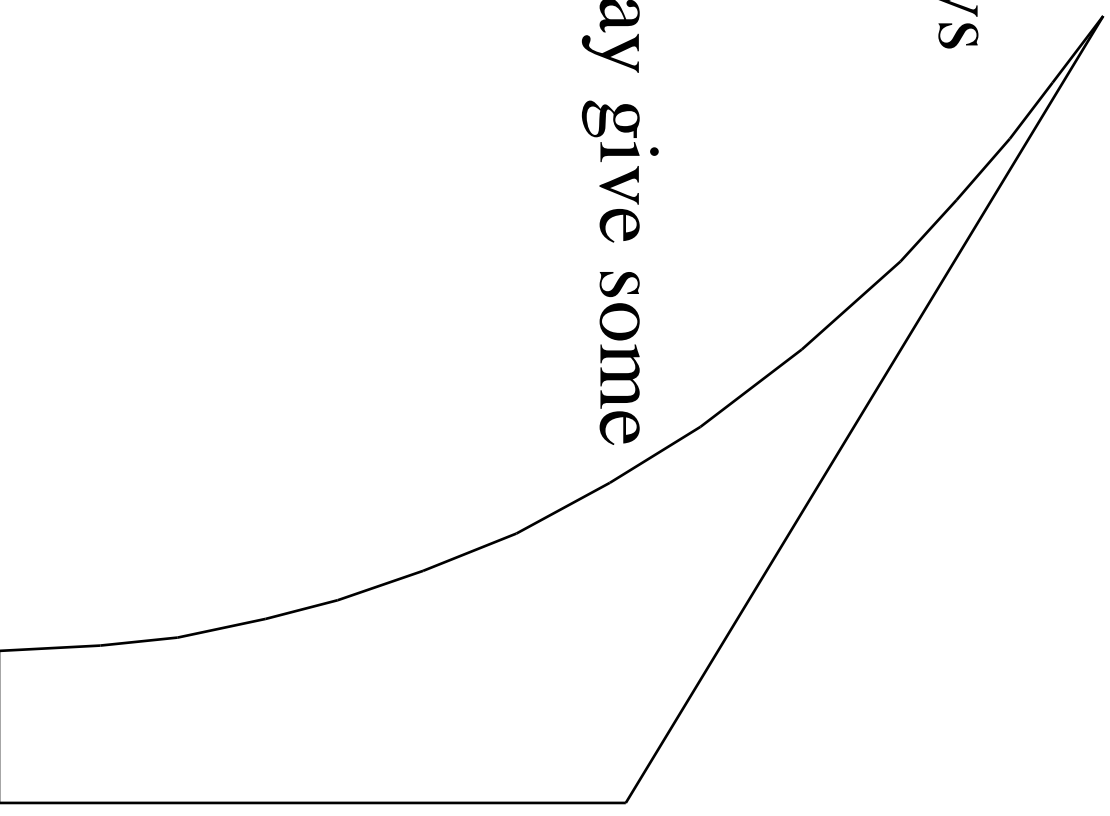
Stylistic improvements

```
expect {
    "220*Name" {
        send_user "Connected!\n"
    }
    timeout {
        send_error "Never got name\n" ; exit 1
    }
}
```

What commands are there?

◗ looping, lists, hash/arrays

◗ procedures, functions

◗ `info` commands

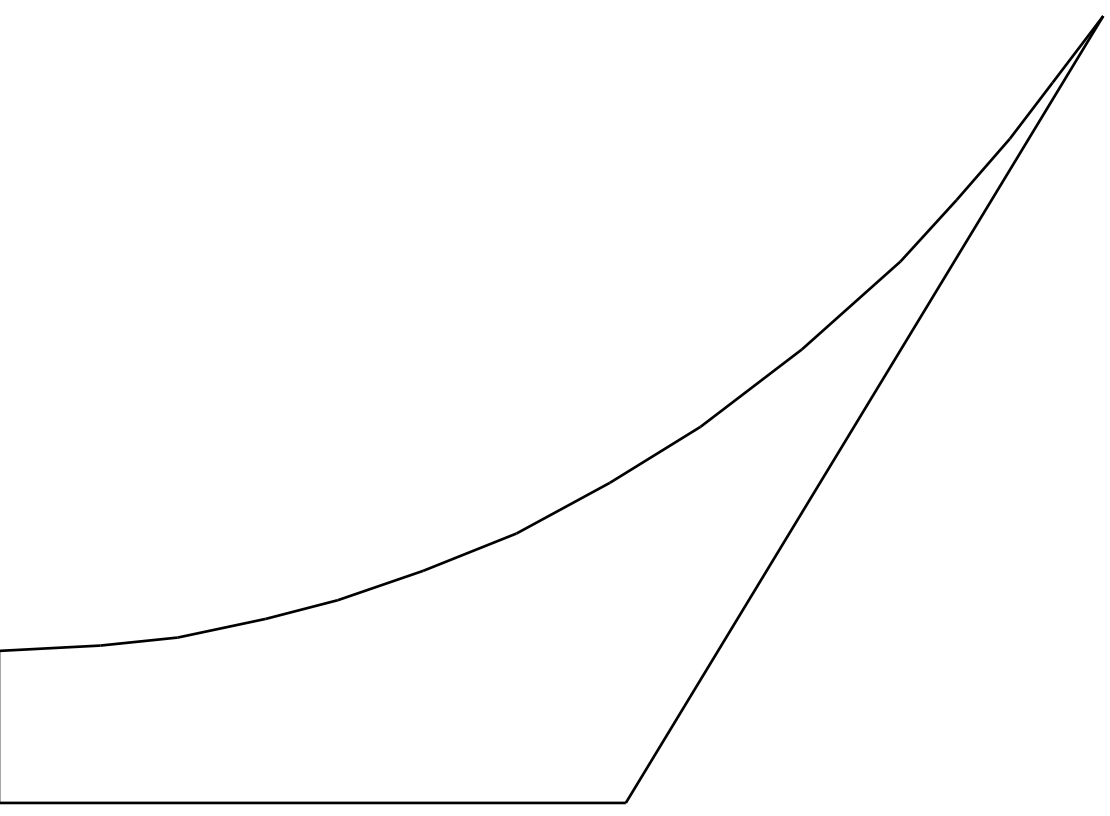◗ commands themselves may give some usage information
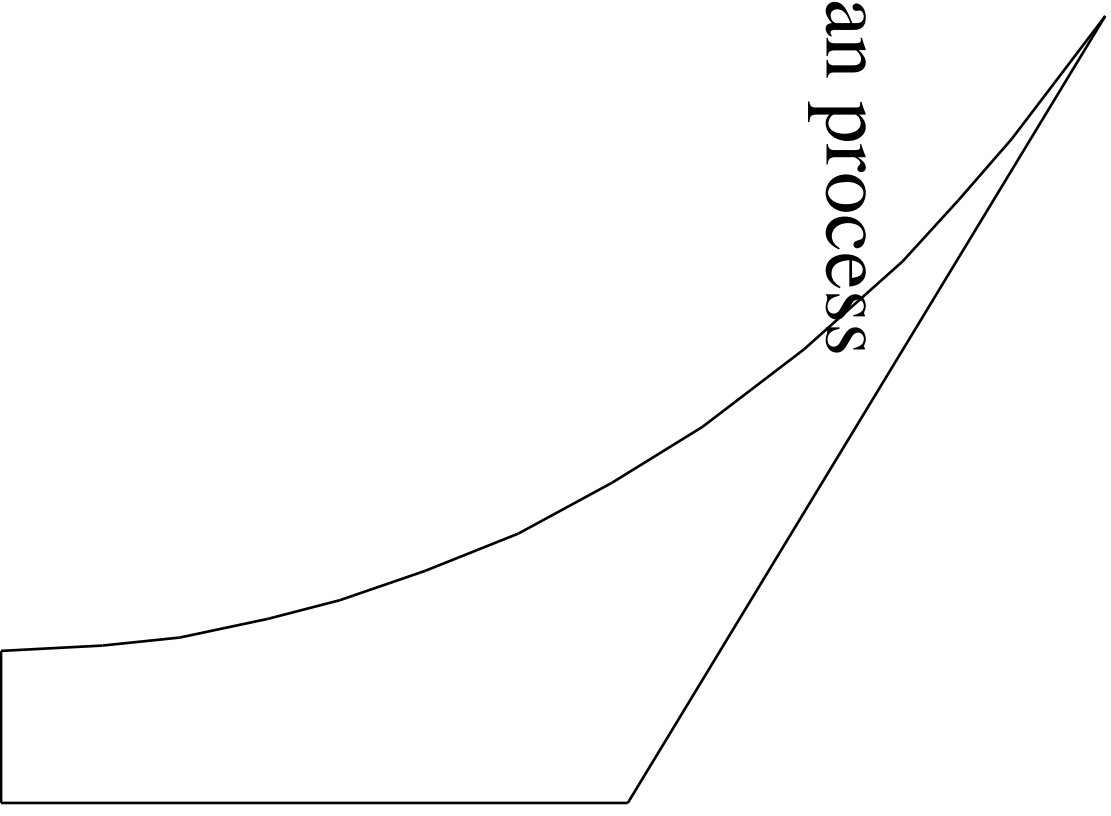
◗ tries `auto_load`

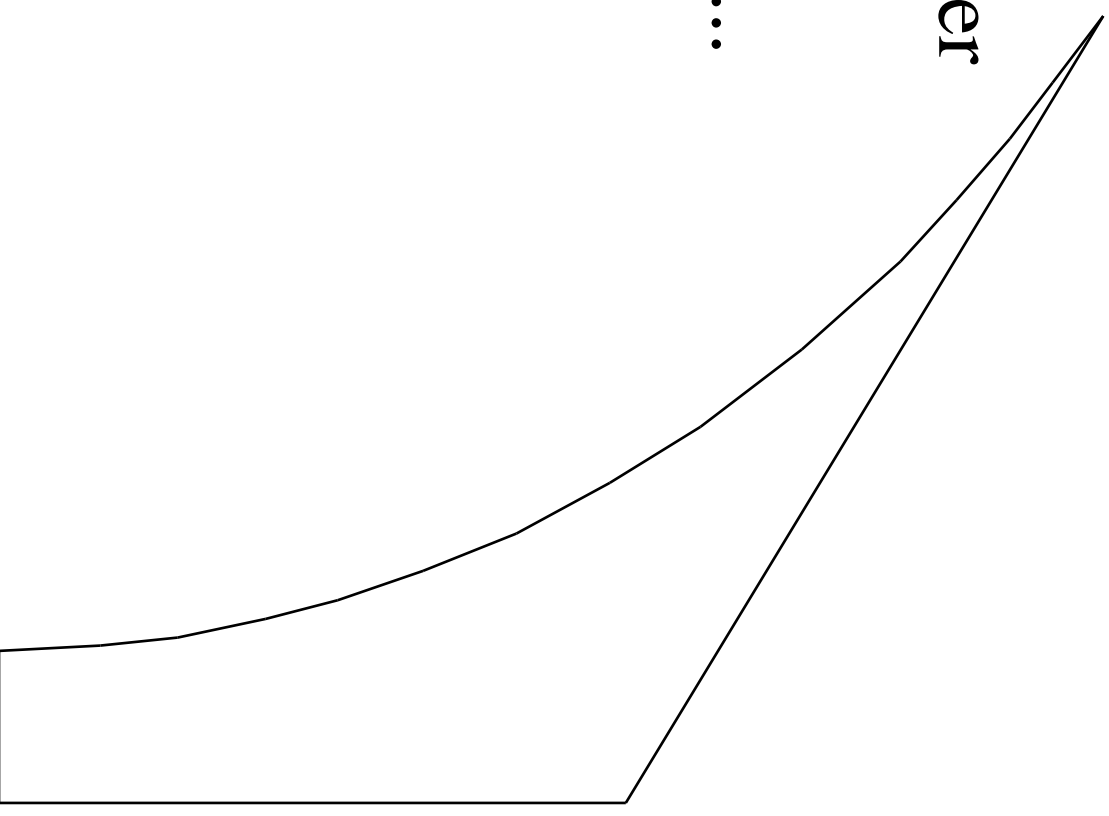# User interaction

1 expect_user

1 interpreter

1 interact

expect_user

1 same as normal expect

1 reads from user rather than process

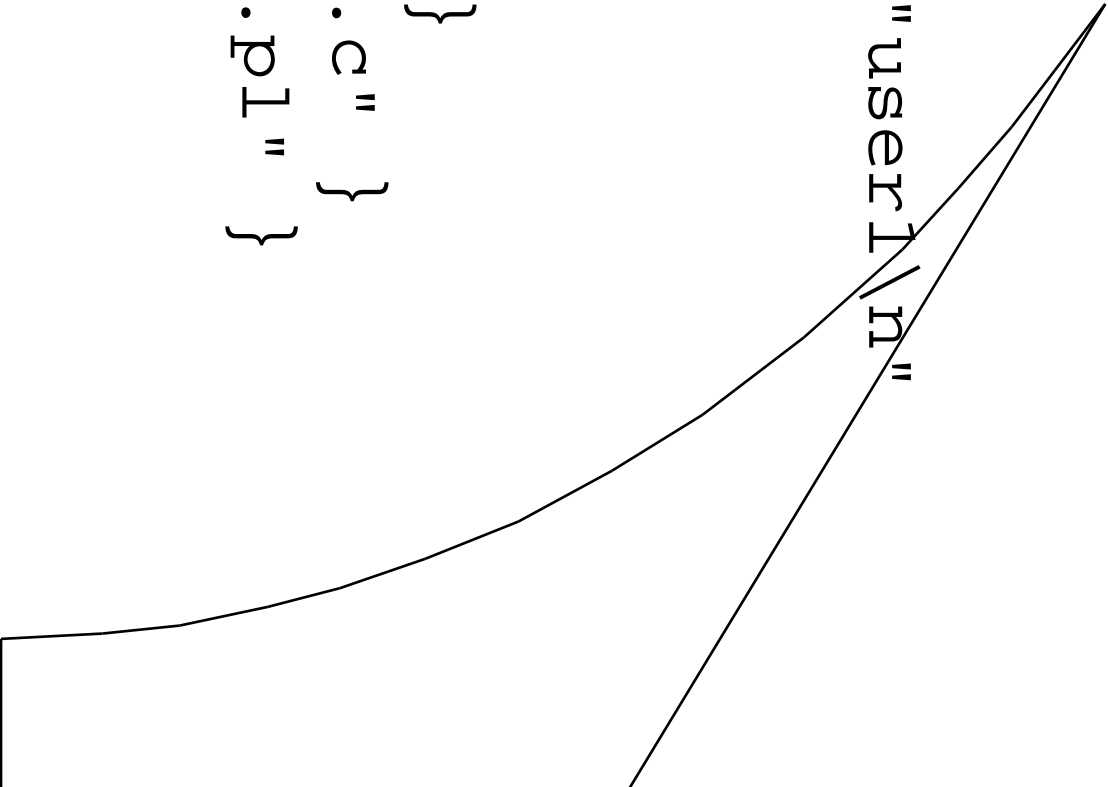# interpreter

- starts an expect interpreter
- useful for debugging
- very useful with interact...

interact

```
spawn ftp localhost
expect "Name" ; send "user1\n"
expect "Password:"
send "class1\n"
expect ftp>
interact {
    rm { send delete }
    ~c { send "mget *.c" }
    ~p { send "mget *.pl" }
}
```
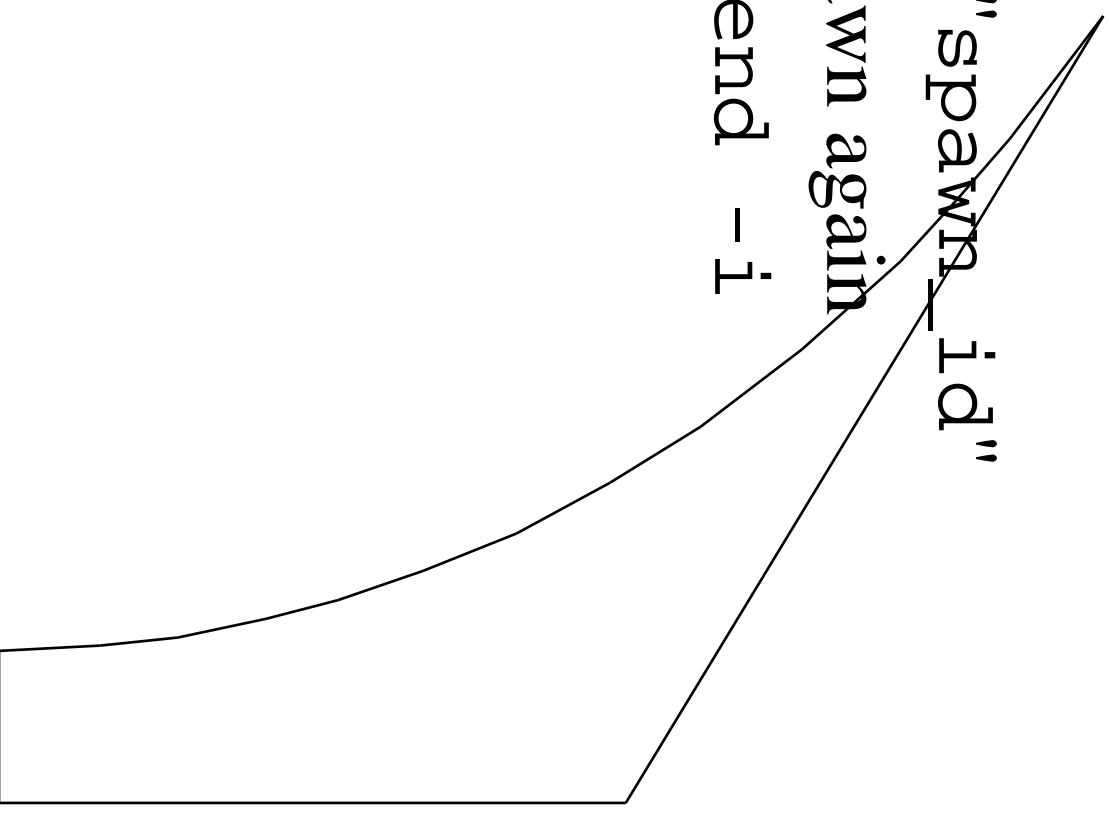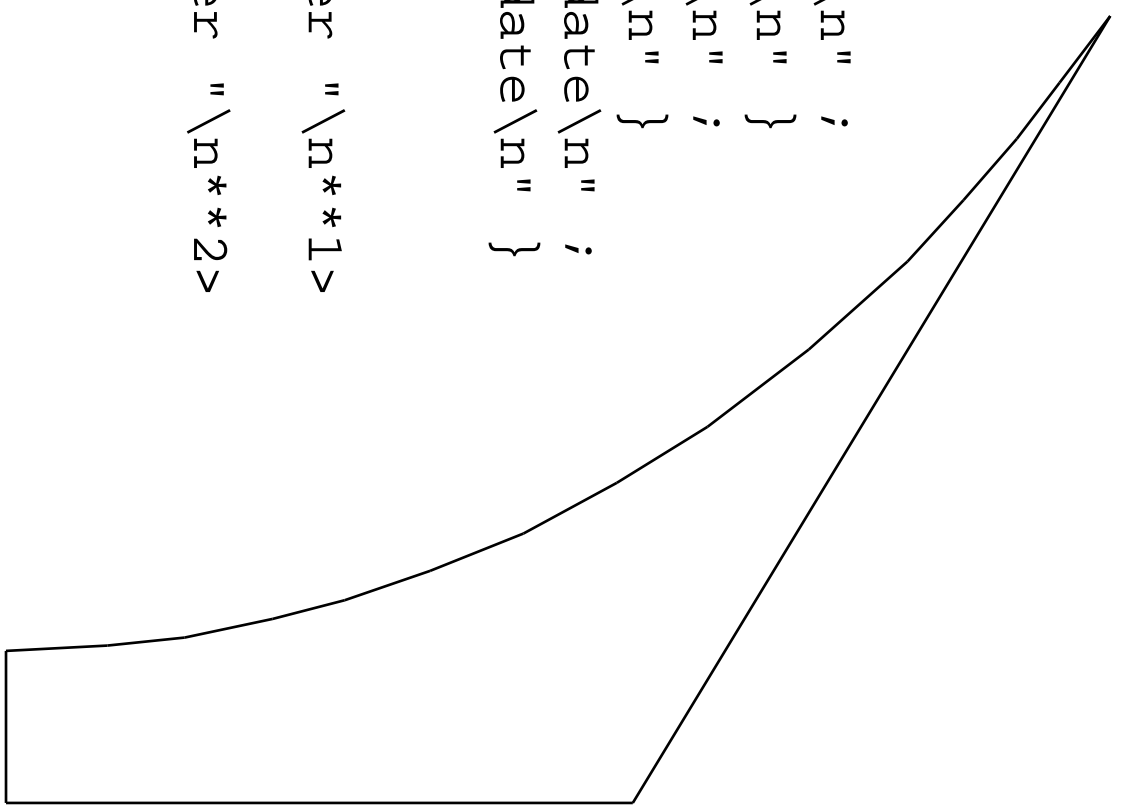
multiple sessions

1 spawn sets the variable "spawn_id"

1 save spawn_id and spawn again
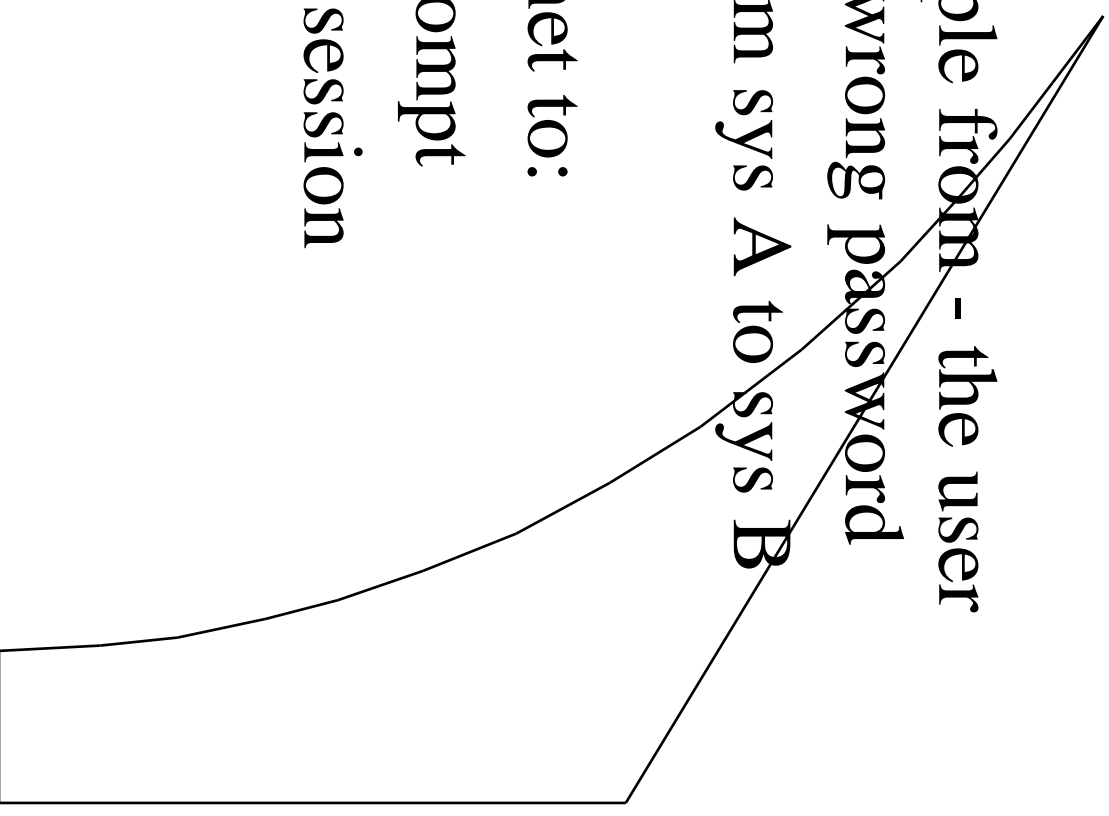
1 use expect -i and send -i

```
spawn telnet host1 ; set session1 $spawn_id ;
interact "^-" return ;# manual login
spawn telnet host2 ; set session2 $spawn_id
interact "^-" return ;# manual login
set timeout -1 ;# no timeout
while (1) {
    send_user "\nid, ls, date or q> "
    expect {
        -i $user_spawn_id
        "id\n" { send -i $session1 "id\n" ;
                send -i $session2 "id\n" }
        "ls\n" { send -i $session1 "ls\n" ;
                send -i $session2 "ls\n" }
        "date\n" { send -i $session1 "date\n" ;
                send -i $session2 "date\n" }
        "q" { break }
        "\n" { }
    }
    -i $session1 -re ".+" { send_user "\n**1>
    $expect_out(buffer)" }
    -i $session2 -re ".+" { send_user "\n**2>
    $expect_out(buffer)" }
}
}
```
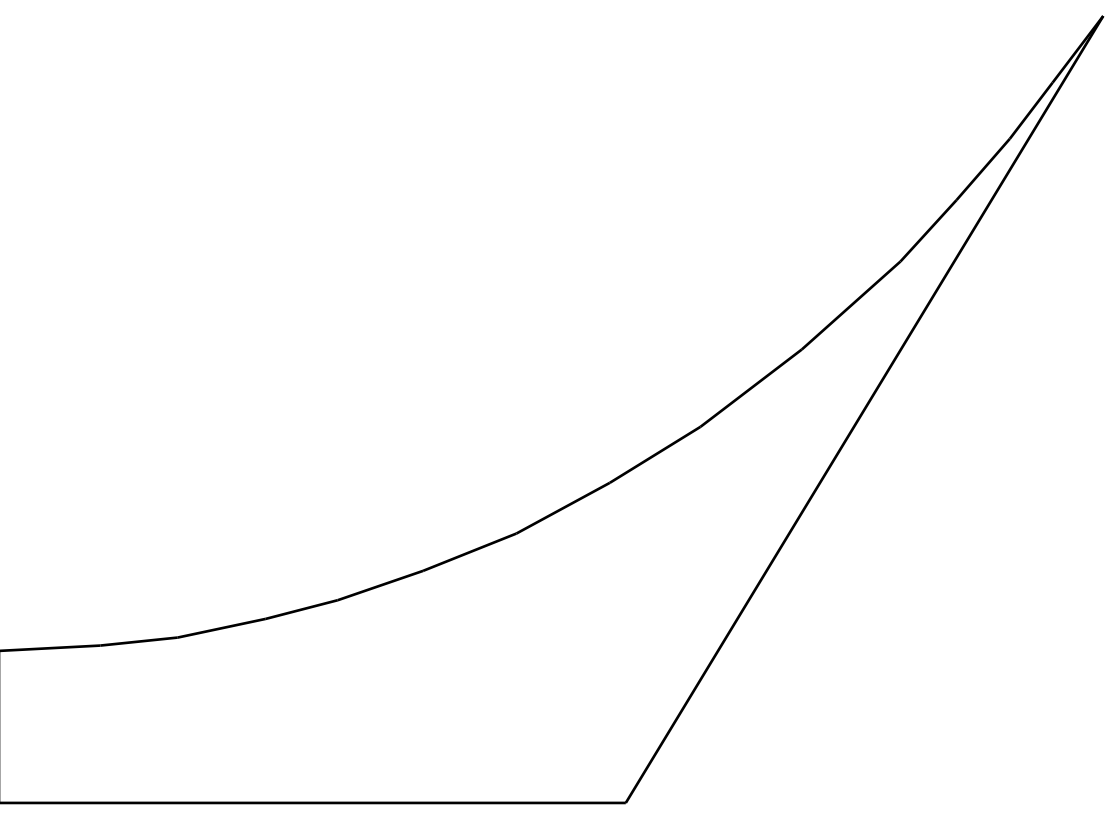
Exercise time

1. Improve the FTP example from - the user should interact to fix a wrong password

2. Extend FTP to copy from sys A to sys B via your machine

3. Fix double-headed-telnet to:

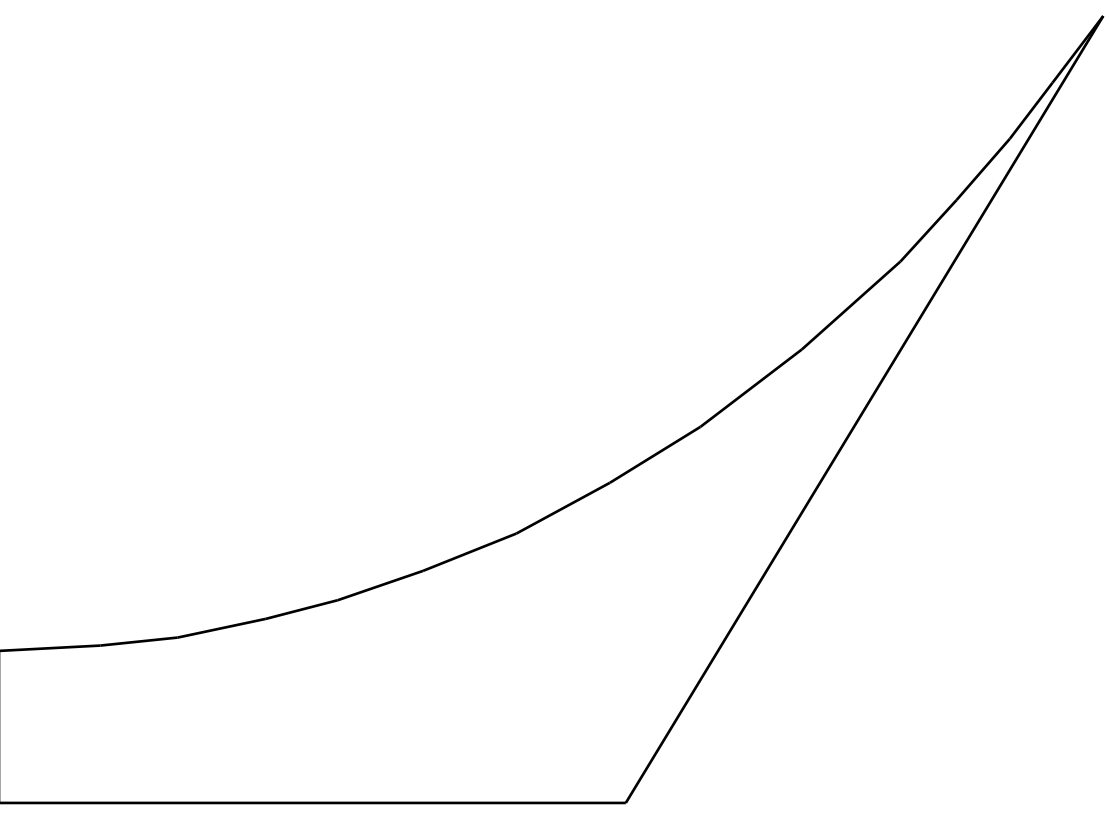   a) not show the shell prompt

   b) interact again with a session

Walk through of Tcl examples

1 sampletcl-basic.exp

1 sampletcl-advanced.exp

Modular programming

- **1** source filename
- **1** namespace

Exercise

1 Remove your most time-wasting task