Testing Interactive Programs

Don Libes

libes@cme.nist.gov

National Institute of Standards and Technology

"I'm from the government and I'm here to hack"

OVERVIEW

- Why testing is good
- Why humans are bad at it
- A brief overview of Expect
- Examples of Expect applied to testing
- POSIX.2a and other problems

Why testing is good

- assure that programs (still) conform
- assure that programs (still) work

Why we've been negligent

- **Humans suck at testing**
- **Simple programs don't cut it**

Ritchie: "*A program designed for inputs from people is usually stressed beyond the breaking point by computer-generated inputs.*"

- **Hey, let's take advantage of this!**

Expect – Very Brief Overview

- **Described: Summer 1990 USENIX**
- **Key ideas:**
  - **Process to process communication**
  - **Pseudo-terminals (ptys)**
  - **Standard language – Tcl (Ousterhout@Berkeley)**

    Described: Winter 1990 USENIX
  - **Pattern-action processing**

    send "string"

    expect "pattern" action "pattern" action

Automating passwd

```
set password [index $argv 2]
spawn passwd [index $argv 1]
expect "*password:"
send "$password\r"
expect "*password:"
send "$password\r"
expect eof
wait
```

Testing passwd

```
send "$password\r"
expect {
        "Password too long" reject-action
        "Retype password" retype-action
}
```

Data File for passwd Tester

```
passwd.exp really-long-passwd  pswd-bad
passwd.exp zrl4twy bogus        pswd-nomatch
passwd.exp zrl4twy zrl4twy      pswd-accept
```

Shortcuts

- expect_before, expect_after

```
expect_after "?" {unexpected_qmark}
send  "foo\r"
expect "?"         {expected_qmark}
```

- interact

  transfer control from script to user

  transfer control from user to script

- and others…

Testing sleep & tty driver

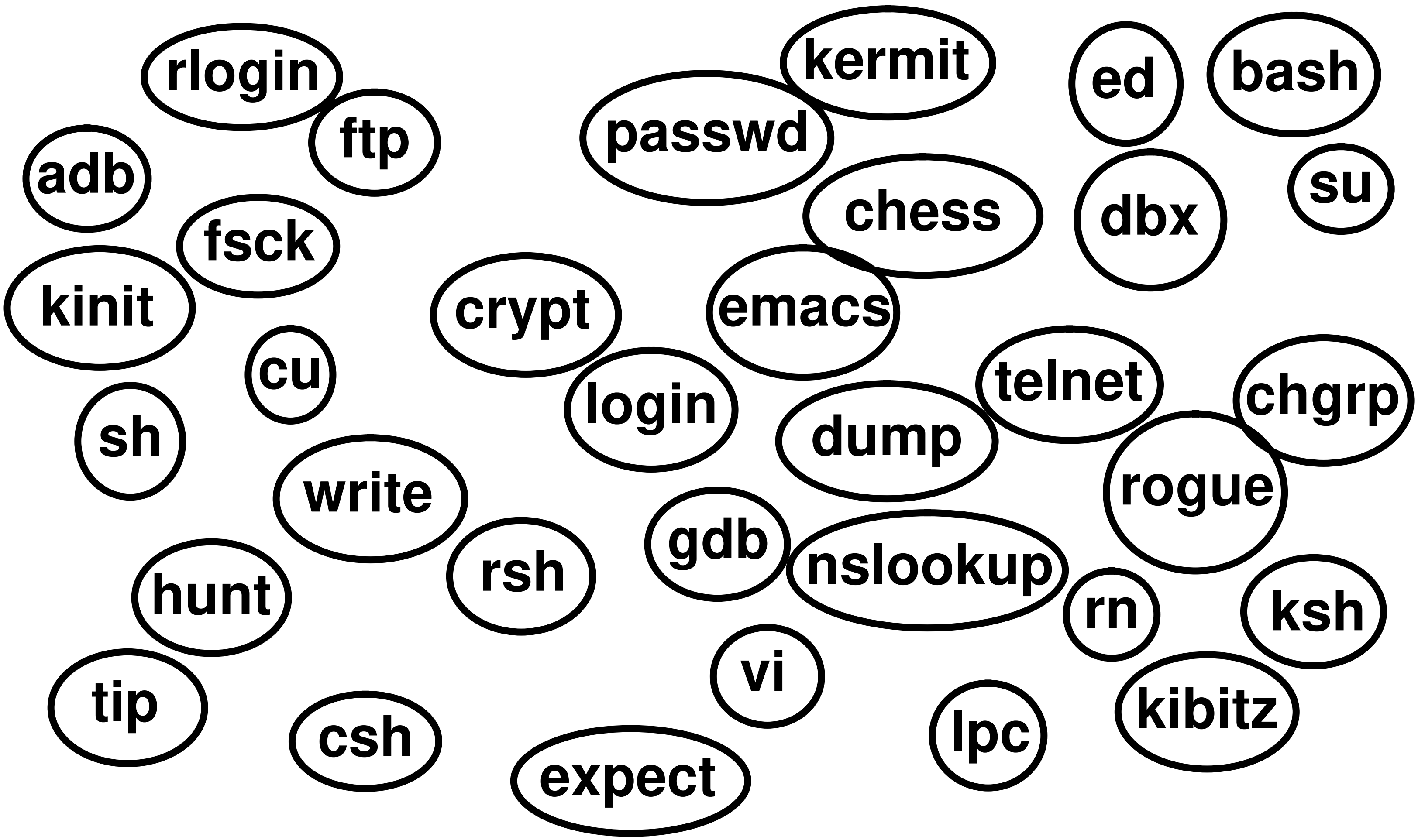
```
spawn csh                   ;# this is a comment
send "sleep 10\r"           ;# start a sleep command
exec sleep 1                ;# let sleep begin
send "\cZ"                  ;# suspend it
exec sleep 10               ;# wait for 10 seconds
send "fg\r"                 ;# let sleep resume
set timeout 5               ;# timeout after 5 secs
expect timeout  {print "control-Z stopped
                            sleep’s clock\n"}
        "$prompt" {print "control-Z didn’t stop
                            sleep’s clock\n"}
```

Applications

- POSIX.2a – interactive shells and tools
- Per-user regression scripts
- Modem testing
- One-shot testing
- Etc.

Expect

- **It's not just for testing anymore!**
- **Some of the programs Expect has automated:**



Conclusion

Expect aids testing and debugging of many interactive programs.

(But writing the specs is still hard.)

Testing Interactive Programs with *Expect*

Don Libes

libes@cme.nist.gov

National Institute of Standards and Technology

"Hard at work creating the standard core dump"