Lecture #22 – Expect

- Expect Background

  Tool for automating interactive applications (i.e. telnet, ftp, passwd, rlogin, etc)
  "Official Web Site" - expect.nist.gov

  Interpreter is either "expect" or "expecttk".
  "expect" contains Expect and TCL support
  "expecttk" contains Expect, Tk, and TCL support

- Syntax:

  Main keywords: spawn, send, expect

  "spawn" creates a new process running the specified command and attaches to expect
  "send" sends output to the new command
  "expect" waits for output from the command

  Example:

  ```
  send "Hello world"
  ```

  Output: Hello world

  Example:

  ```
  expect "hi"
  send "Hello, there"
  ```

  Output: <You type "hi">
  Hello, there

  Note: default timeout for input is 60 seconds, but can be adjusted with "timeout" variable

  Example:

  ```
  set timeout 120
  set timeout –1
  ```

  Example (pattern-action pairs):

  ```
  expect "hi" { send "You said hi\n" } \
      "hello" {send "hello, yourself\n" } \
      "bye" {send "Goodbye, cruel world\n"}
  ```

- Example: Changing root passwd on several machines

  Sample session might look like:

  localhost$ telnet remote0

  Welcome to remote0.
  login: myname
  Password: <password>

  Last Login: Yesterday.

  remote0$ su
  Password: <root's password>

  remote0# passwd root
  New password: <new password>
  Re-enter new password: <new password>
  Password changed.

  remote0# exit
  remote0$ exit

  Connection closed by foreign host.
  localhost$ telnet remote1
  [[etc…]]

  Yields an expect script like :

```
foreach host "remote0 remote1 … remoteN" {
        spawn telnet $host
        expect "login: ";               send "myname\r";
        expect "Password: ";            send "mypassword\r";
        expect "$ ";                    send "su\r"
        expect "password: ";            send "rootpassword\r"
        expect "# ";                    send "passwd root\r"

        expect {
                "password: " {
                        send "rootnewpassword\r";    exp_continue
                }
                "# " {
                        send "exit\r"
                }
        }
        expect "$ ";    send "exit\r"
```

- Example: Reprompt

```
# Name: reprompt
# Description: reprompt every so often until user enters something
# Usage: reprompt timeout prompt
# Author: Don Libes, NIST

foreach {timeout prompt} $argv {}

send_error $prompt
expect {
    timeout {
  send_error "\nwake up!!\a"
  send_error \n$prompt
  exp_continue
    }
    -re .+ {
  send_user $expect_out(buffer)
    }
}
```

- Example: Distributing files to remote machines

```
#!/usr/local/bin/expect –f

match_max 10000

set env(TERM) "dialup"
set user $env(LOGNAME)
stty –echo

send_user "Enter password for $user now: "
gets stdin password
send_user "\nEnter password for root on remotes now: "
gets stdin rootpw
stty echo

foreach machine $argv {
        spawn ftp $machine
        expect –re "Name .*: "
        send "$user\r"
        expect "word:"
        send "$password\r"
        expect "ftp> "; send "bin\r"
        expect "ftp> "; send "cd /tmp\r"
        expect "ftp> "; send "put localfile.tar\r"
        expect "ftp> "; send "quit\r"
        send_user "\r\nftp exited.\n"
        sleep 1; telnet $machine
        expect "ogin: "; send "$user\r"
```

```
        expect "word: "; send "$password\r"
        expect –re "(\\$|>) "; send "su\r"
        expect "word: "; send "$rootpw\r"
        expect "# "; send "cd /tmp\r"
        expect "# "; send "tar xvf localfile.tar\r"
        expect "# "; send "exit\r"
        expect –re "\\$|>"; send "exit\r"
}
```

- Example checking innd daemon with a procedure

```
#!/usr/local/bin/expect –f

set timeout 10
proc smart_expect { look send } {
        expect {
                -exact $look {
                        send $send
                }
                timeout {
                        send_user "Timeout occurred\n"
                        exit 1
                }
        }
}

spawn telnet newshost 119
match_max 10000
smart_expect "\r
200 " "group comp.risks\r"
smart_expect "\r
211 " "quit\r"
smart_expect "\r
205 " ""
smart_expect eof ""
```

- Example: Weather

```
exp_version -exit 5.0

if {$argc>0} {set code $argv} else {set code "WBC"}

proc timedout {} {
  send_user "Weather server timed out.  Try again later when weather
server is not so busy.\n"
  exit 1
}
```

```
set timeout 60

set env(TERM) vt100     ;# value doesn't matter, just has to be set

spawn telnet rainmaker.wunderground.com 3000
while {1} {
  expect timeout {
          send_user "failed to contact weather server\n"
          exit
  } "Press Return to continue*" {
                # this prompt used sometimes, eg, on opening connection
                send "\r"
  } "Press Return for menu*" {
                # this prompt used sometimes, eg, on opening connection
                send "\r"
  } "M to display main menu*" {
          # sometimes ask this if there is a weather watch in effect
          send "M\r"
  } "Change scrolling to screen*Selection:" {
          break
  } eof {
          send_user "failed to telnet to weather server\n"
          exit
  }
}
send "C\r"
expect timeout timedout "Selection:"
send "4\r"
expect timeout timedout "Selection:"
send "1\r"
expect timeout timedout "Selection:"
send "1\r"
expect timeout timedout "city code:"
send "$code\r"
expect $code            ;# discard this

while {1} {
  expect timeout {
          timedout
  } "Press Return to continue*:*" {
          send "\r"
  } "Press Return to display statement, M for menu:*" {
          send "\r"
  } -re "(.*)CITY FORECAST MENU.*Selection:" {
          break
  }
}

send "X\r"
expect
```