Tutorial de expect, v0.6

Manuel de Vega Barreiro <mbarreiro@red.madritel.es>

18 de marzo de 2002

Este es un rápido tutorial sobre expect, una herramienta para automatización de tareas interactivas.

Next Previous Contents

1. Introducción a Expect

1.1 Porque surgió Expect?

Los administradores y técnicos de sistemas se encuentran a menudo con la ingrata tarea de realizar aburridas tareas de mantenimiento o prevención en los incontables aparatos que están bajo su responsabilidad.

A menudo es fácil realizar un fichero de comandos, bien directamente en el interprete de comandos Bourne (la shell de toda la vida), bien de forma mas elaborada en Perl o Phyton, que nos permite automatizar este tipo de trabajos, y así dedicarnos a otras actividades mas creativas.

Sin embargo a menudo nos encontramos con aplicaciones de configuración interactivas que no permiten este tipo de solución. Seguramente habéis intentado inyectar comandos directamente desde un fichero desviando la entrada estándar, pero esta solución, aparte de burda y poco elegante, no siempre da los resultados apetecidos:

- No permite distintas acciones en función de las respuestas.
- No contempla los casos de error o temporización.
- Algunos programas se niegan a funcionar de esta manera.

Esto a veces nos obliga a repetir en numerosas ocasiones tediosas sesiones interactivas para configurar un encaminador, determinar el estado de sistemas vitales para el funcionamiento de la red, conectarse a un determinado servidor, etc.

1.2 Para que vale Expect?

Si sufrís a menudo estas penalidades, y estáis buscando una solución sencilla a este tipo de problemas, Expect es la herramienta que necesitáis. Seguramente algunos recordareis Crosstalk, un programa de comunicaciones que permitía automatizar conexiones serie, a base de esperar ciertos mensajes por el puerto y responder de forma apropiada en función de lo recibido.

Pues bien Expect funciona de forma similar y además se puede aplicar a toda clase de aplicaciones interactivas con presentación de tipo textual, incluyendo programas semigraficos basados en curses (lamentablemente los programas de tipo gráfico no tiene cabida en Expect ;-( ).

1.3 Un apunte sobre Tcl.

Expect se apoya en el lenguaje de comandos extensible Tcl, y por tanto puede parecer necesario aprender otro lenguaje de comandos mas. Tal vez seria la mejor forma de exprimir sus posibilidades, pero os puedo asegurar que yo lo uso sin saber apenas nada de tcl (estoy contento con Perl y no me apetece cambiar).

1.4 Un primer ejemplo.

Veamos un sencillo ejemplo del uso de Expect (no, no es el Hola Mundo!). Pongamos que tenemos que conectarnos muy a menudo con un determinado servidor de ficheros ftp y situarnos en un directorio concreto para después ejecutar una serie de comandos. A mano tendríamos que teclear unos cuantos comandos, con expect la cosa quedaría en uno único.

El programa (ftp_1.exp) seria el siguiente:

```
#!/usr/bin/expect -f
#
##############################################################
#
```

```
# este parte se comentara mas adelante, de momento insertala al
# principio de cada fichero de comandos.
#
set force_conservative 1  ;# set to 1 to force conservative mode even if
                          ;# script wasn't run conservatively originally
if {$force_conservative} {
        set send_slow {1 .001}
        proc send {ignore arg} {
                sleep .1
                exp_send -s -- $arg
        }
}
##################################################################
#
spawn ftp localhost
expect -exact "Name (localhost:root): "
send -- "ftp\r"
expect "Password:"
send -- "barreiro@arrakis.es\r"
expect -exact "ftp> "
send -- "binary\r"
expect -exact "ftp> "
send -- "hash\r"
expect -exact "ftp> "
send -- "cd pub/spain/gnome\r"
log_user 1
expect -exact "ftp> "
send -- "ls\r"
expext -exact "ftp> "
send -- quit\r"
expect eof
exit
```

Veamos el significado de los comandos mas importantes de este programa:

spawn... ejecuta la aplicación a controlar

expect... escucha la salida de la aplicación y espera la cadena indicada

send... envía a la aplicación la cadena indicada, '\r' corresponde a la pulsación de la tecla 'Retorno'

expect eof normalmente expect recibe un eof al terminar el comando lanzado con spawn. exit termina la ejecucion del programa expect

El programa lanza en primer lugar el comando 'ftp' y queda a la espera de la peticion de nombre de usuario. Una vez que se recibe se envia 'ftp'. Se queda ahora a la espera de la peticion de password, y se envia esta al recibirla. Una vez que llega el indicador de 'ftp', se envia los comandos deseados de forma succesiva.

Para terminar se ejecuta el comando 'exit' termina el programa expect

---

Next Previous Contents

2. Instalación

evidentemente si queremos utilizar Expect, el primer paso es instalarlo en nuestro sistema.

2.1 Distribuciones Linux.

Si trabajamos con alguna distribución de Linux, seguramente tengamos ya Expect instalado y listo para su uso.

Podemos comprobarlo usando las herramientas de la distribucción. Por ejemplo si se utiliza Red Hat:

```
rpm -qa | grep expect
```

En caso contrario se debe instalar los paquetes tcl y expect:

```
cd /mnt/cdrom
rpm -ivh tcl*.rpm
rpm -ivh expect*.rpm
```

2.2 Solaris.

La forma mas sencilla de instalar expect en Solaris es copiar de la red los paquetes tcl y expect en formato pkg. Estos paquetes se pueden conseguir para distintas versiones de Solaris en:

http://sunfreeware.com

Al descargar los ficheros comprimidos con gzip, los navegadores quitan la extension .gz del nombre del fichero, pero no siempre lo descomprimen.

Una vez que tenemos los pquetes necesarios, procedemos a instalarlos usando los siguinetes comandos:

```
gzip -d tcl-8.0.3-sol7-sparc-local.gz
pkgadd -d tcl-8.0.3-sol7-sparc-local
gzip -d expect-5.28-sol7-sparc-local.gz
pkgadd -d expect-5.28-sol7-sparc-local
```

2.3 HP-UX.

La forma mas sencilla de instalar expect en HP-UX es copiar de la red los paquetes tcl y expect en formato utilizado por Software Distributor (SD). Estos paquetes se pueden conseguir para distintas versiones de HP-UX en:

http://hpux.cict.fr/

Al descargar los ficheros comprimidos con gzip, los navegadores quitan la extension .gz del nombre del fichero, pero no siempre lo descomprimen.

Una vez que tenemos los paquetes necesarios, procedemos a instalarlos usando los siguientes comandos:

```
gzip -d tcl-8.1.1-sd-11.00.depot.gz
swinstall -s tcl-8.1.1-sd-11.00.depot tcl
gzip -d expect-5.31-sd-11.00.depot.gz
swinstall -s expect-5.31-sd-11.00.depot
```

2.4 Windows.

Parece que Expect tambien se puede usar con Windows, eso si como algunas limitaciones.

Podeis encontrar información sobre somo instalarlo y usarlo en: http://bmrc.berkeley.edu/people/chaffee/tcltk.html

---

Next Previous Contents

3. Lo basico de Expect

A continuacion iremos ampliando las posibilidades del ejemplo presentado en el primer capitulo, al tiempo que aprendemos mas acerca de la sintaxis de los comandos de expect.

3.1 El comando Expect

El programa anterior funcionara siempre que la conexion al servidor de ftp sea correcta. Pero que pasa por ejemplo si esta falla (ftp_21)?

Cuando la conexion falla, ftp presentara un determinado mensaje de error. El comando expect puede gestionar distintos mensajes de una forma similar a una sentencia case (ftp_31).

```
expect -exact {
  "connection refused" {puts "Conexion denegada\n";exit}
  "Host name lookup failure" {puts "servidor desconocido\n";exit}
  "Name"
}
```

Y si no se recibe ninguno de estas cadenas, ya sea la esperada o cualquiera de los errores conocidos (ftp_22)?.

Para este caso tenemos la opcion generica 'timeout'. Si expect no recibe ninguna de las cadenas esperadas en el un tiempo determinado (10sg por defecto), se ejecuta lo indicado en la opcion 'timeout' (ftp_32).

```
expect -exact {
  timeout {puts "El programa ha temporizado\n";exit}
  "connection refused" {puts "Conexion denegada\n";exit}
  "Host name lookup failure" {puts "servidor desconocido\n";exit}
  "Name"

}
```

Por supuesto el valor de la temporizacion se puede variar en cualquier momento a lo largo del programa, e incluso anularse si se la asigna el valor -1.

```
set timeout 15
set timeout -1
```

3.2 Los patrones regulares

Otra cosa que salta a la vista en los ejemplos anteriores es que escribir las cadenas que espera el comando expect, puede llegar a ser una tarea ardua y tediosa. Para ayudarnos en esta tarea se puede recurrir a los patrones regulares, que nos permitiran contruir estas cadenas mas facilmente usando caracteres comodin. Expect permite usar dos sintaxis para contruir los patrones regulares:

- Glob, '-gl'
- Regexp, '-re'

El primero es de sintaxis muy sencilla, pero menos flexible. El segundo es utiliza la misma sintaxis que el comando grep.

explicacion de los tipos de patrones (solo tabla).

Con los comodines tambien podemos solventar el problema que plantean aplicaciones en las que el indicador cambia durante la ejecucion, en ejecuciones succesivas o al cambiar de usario o sistema.

exact glob regexp

[a-z] [a-z] cualquier caracter entre a y z

[^a-z] [^a-z] cualquier caracter que no este entre a y z

? . un caracter cualquiera

* .* un numero cualquiera de caracteres

^ ^ el principio de la cadena

$ $ el final de la cadena

\[\\\[\\\[carcater corchete

* * caracter *

. \. caracter .

- Aplicaciones que insertan un numero de secuencia en el indicador
- Aplicaciones que incluyen la fecha en el indicador.
- Ejecucion de programas expect en distintos sistemas que incluyan el nombre de la maquina en el indicador

Aqui telnet_1.exp Aqui algunos ejemplos de patrones tipicos.

3.3 Controlando la presentacion

En los ejemplos anteriores podemos observar que todo el texto de salida , tanto el de respuesta de los comandos como la simulacion de entrada de usuario, aparece en la pantalla. Esto puede ser apropiado durante las pruebas iniciales, pero excesivo y confuso para el uso cotidiano. La presentacion de mensajes de respuesta de los comandos se puede eliminar usando la variable 'log_user'. (ftp_51.exp)

```
log_user 0        elimina los mensajes de respuesta de los comandos
log_user 1        activa la presentacion de respuestas
```

Si, una vez eliminada totalmente la presentacion, queremos enviar mensajes informando al usuario del progreso del programa, podemos hacerlo con el comando (ftp_52.exp) :

```
send_user "mensaje\n"
```

3.4 Paso de parametros a los programas Expect.

Expect recibe los parametros en la variable 'argv' (ftp_61.exp)

Podemos acceder a cada una de las variables de la matriz argv usando la siguiente sintaxis:

[lindex $argv n] parametro n (recordar que el primer elemeto de la matriz es 0)

Tambien podemos asignar los parametros a variables:

set sistema [lindex $argv 0]

Y podemos saber el numero de parametros usando la variable:

$argc

De esta forma podemos controlar facilmente el paso de parametros a los programas:

```
if $argc<2 {
  send_user "$argv0: faltan parametros\n"
  exit
}
set sistema [lindex $argv 0]
set usuario [lindex $argv 1]
send_user "$sistema\n"
send_user "$usuario\n"
send_user "[lindex $argv 0]\n"
send_user "[lindex $argv 1]\n"
```

3.5 El comando Send

Del comando send no hay mucho que contar, simplemente recordar incluir el doble guion para que no interprete parte de la cadena como una opcion propia:

```
send -- "...."
```

Y no olvidar terminar los comandos enviados al programa con un retorno de carro, o el caracter que necesite la aplicacion:

```
send -- "get fichero.tgz \r"
```

3.6 el comando spawn

Este comado inicia la ejecucion de la aplicacion controlando su entrada y salida.

Se pueden lanzar varias aplicaciones simultaneamente, controlando a traves de la variable spawn id cual de ellos atiende a los comandos expect y send.

3.7 el comando interact

Con el comando 'interact' podemos enviar un primer grupo de comandos de forma automatica, recuperar despues el control sobre la aplicacion para enviar ordenes de forma manual, y finalmente terminar la ejecucion nuevamente de forma automatica (ftp_71.exp).

```
send_user "Entro en modo interactivo. Pulsa 'escape' para continuar\n"
interact "escape" return
send_user "Vuelvo al modo automatico\n"
...
```

4. Correccion de errores

4.1 Traza de la ejecucion.

Usando el comando:

```
log_file fichero
```

podemos salvar a un fichero todos los mensajes que veria un usuario durante la ejecucion de un programa expect. Los mensajes se añaden al final del fichero si este existe.

En el caso de que queramos crear un fichero de trazas nuevo usaremos la opcion:

```
log_file -noappend fichero
```

Si queremos detener la captura usaremos el comando

```
log_file
```

Si quisieramos capturar toda la salida incluyendo lo suprimido con la opcion 'log_user 0', debemos usar la opcion:

```
log_file -a fichero
```

si quieres enviar mensajes a la salida de log, podemos hacerlo con el comando:

```
send_log "mensaje\n"
```

4.2 Chequeo de la ejecucion y de los patrones de concordancia

```
exp_internal -f fichero 0    activa la captura de mensajes de diagnostico
exp_internal 0                 cierra la captura de mensajes de diagnostico
```

Se puede enviar la salida de diagnosticos directamente a la salida estandar, pero la cantidad de informacion generada recomienda usar un fichero para analizarlo mas comodamente.

El fichero generado contiene una traza de la ejecucion del programa expect y del proceso de concordancia de cada uno de los patrones.

5. Trucos

5.1 Ralentizacion en el envio de caracteres

el codigo que figura al principio de cada guion:

```
                        ;# script wasn't run conservatively originally
set force_conservative 1  ;# set to 1 to force conservative mode even if
if {$force_conservative} {
        set send_slow {1 .001}
        proc send {ignore arg} {
                sleep .1
                exp_send -s -- $arg
        }
}
```

sirve para ralentizar el envio de las respuestas, simulando de esta manera nuestra cansina forma de teclear.

A partir de la version 5.20 se puede sustituir por la siguiente llamada:

```
set send_slow {1.1}
```

Existe tambien una nueva opcion para simular la cadencia humana al teclear:

```
set send_human {.2 .2 1 0.1 4}
```

5.2 Listados paginados

En algunos ocasiones los resultados de la ejecución de una consultan se presentan de forma paginada. Esto es el dispositivo interrogado presenta una numero de determinado de lineas y espera a que pulsamos una determinada tecla antes de presentar más información.

Para resolver esta situación en nuestros guiones podemos utilizar el siguiente codigo:

```
...
send -- "show test\r"
expect {
        "More" {send -- "\r";exp_continue}
        "onsole>" {}
        }
...
```

En el ejemplo vemos que cuando el listado se pare mostrando el mensaje 'More', expect enviara un retorno de carro y volvera a ejecutar el bucle de espera. Este ciclo se repetira hasta que, finalizado el listado, el dispositico presente de nuevo el indicador de sistema.

5.3 Comandos expect_after y expect_before

Es posible que un dispositivo cierre una conexion de forma inesperada, o un programa aborte por un fallo de programación. En estos

casos nuestro guión recibiría un «eof», por lo que podríamos tratar esta situación insertando el siguiente codigo:

```
expect {
    "onsole>" { send -- "show\r"}
    eof {puts "Final inesperado de la conexion\n"; exit}
}
```

Pero claro, hacer esto para cada secuencia expect seria una tarea tediosa, y ademas el codigo de nuestro guión seria poco claro.

la solución a este problema es usar las sentencia expect_after y expect_before. Como su nombre indica al inlcuirlas en un guión, se comprobara la presencia de la cadena indicada antes o despues de las indicadas en cada secuencia expect.

```
expect_after {
    eof {puts "Final inesperado de la conexion\n"; exit}
}

send -- "show\r";
expect {
    "onsole>" { send -- "\r"}
}
```

En el ejemplo anterior, cada secuencia expect comprobara en primer lugar las cadenas que espera recibir y despues si se ha recibido un «eof».

También puede servir para tratar las respuestas a comandos mal tecleados:

```
expect_before {
    "Invalid command syntax {puts "Comando incorrecto\n"; exit}
}

send -- "sohw -i\r";
expect {
    "onsole>" { send -- "\r"}
}
```

En este caso, sin expect_before, el guión quedaría bloqueado despues de ejecutar el comando, ya que no se espera la cadena "Invalid command syntax".

Next Previous Contents

6. Opciones avanzadas y utilidades

6.1 autoexpect

Con autoexpect se pueden generar guiones para automatizar tareas sin escribir una sola linea de programa. Para conseguir esta maravilla basta ejecutar la sesion que queremos automatizar de forma manual, arrancando previamente el comando autoexpect. Veamos la secuencia:

```
autoexpect -f captura.exp
telnet router
...
quit
<ctrl-D>
```

Tal y como muestra el ejemplo la sesion se cierra tecleando <ctrl-D>. El fichero captura.exp se puede usar directamente para repetir la sesion, aunque en general habra que retocarlo para mejorar su legibilidad y hacerlo mas general.

6.2 expectk

6.3 kibitz y xkibitz

7. Acerca de Expect y de este tutorial

7.1 El Creador de Expect

Expect es un programa de Don Libes del NIST, instituto nacional de estandares y tecnologia de Estados Unidos, que pago parte de su diseño e implementacion. Es un programa de dominio publico y por tanto puede usarse de forma gratuita. No obstante, y como suele ser habitual en los programas con licencias GPL o similares, al autor le gustaria que se le citara en los trabajos derivados.

7.2 Informacion complementaria.

Las paginas man de Expect son una buena fuente de informacion para iniciarse en el manejo de Expect. Ademas con el paquete (al menos en la distribuccion Red Hat) vienen varios ejemplos.

Tambien existen algunos libros dedicados a Expect:

- "Exploring Expect: A Tcl-Based Toolkit for Automating Interactive Programs" by Don Libes, pp. 602, ISBN 1-56592-090-2, O'Reilly and Associates, 1995.
- "expect: Curing Those Uncontrollable Fits of Interactiv- ity" by Don Libes, Proceedings of the Summer 1990 USENIX Conference, Anaheim, California, June 11-15, 1990.
- "Using eexxppeecctt to Automate System Administration Tasks" by Don Libes, Proceedings of the 1990 USENIX Large Installa- tion Systems Administration Conference, Colorado Springs, Colorado, October 17-19, 1990.
- "Tcl: An Embeddable Command Language" by John Ousterhout, Proceedings of the Winter 1990 USENIX Conference, Washington, D.C., January 22-26, 1990.
- "expect: Scripts for Controlling Interactive Programs" by Don Libes, Computing Systems, Vol. 4, No. 2, University of California Press Journals, November 1991.
- "Regression Testing and Conformance Testing Interactive Programs", by Don Libes, Proceedings of the Summer 1992 USENIX Conference, pp. 135-144, San Antonio, TX, June 12-15, 1992.
- "Kibitz - Connecting Multiple Interactive Programs Together", by Don Libes, Software - Practice & Experience, John Wiley & Sons, West Sussex, England, Vol. 23, No. 5, May, 1993.
- "A Debugger for Tcl Applications", by Don Libes, Proceed- ings of the 1993 Tcl/Tk Workshop, Berkeley, CA, June 10-11, 1993.
- "A Debugger for Tcl Applications", by Don Libes, Proceed-
- Documentacion del paquete de testeo DejaGnu, programa que usa Expect de forma intensiva.

7.3 Versiones de este tutorial.

La version mas reciente de este tutorial se encuentra disponible en:

http://www.croftj.net/~barreiro/spain/expect/expect_tutorial.html

Tambien podras encontrar copias de este manual en las paginas de Lucas, Insflug y otras fuentes habituales de documentacion sobre Linux en Castellano.

versión 0.2 (3-11-1998) primera entrega con motivo de la 3ª reunión de usuarios de Linux de Madrid.

versión 0.3 (12-01-1999) varias correcciones

versión 0.4 (20-10-2001) nuevos capitulos

versión 0.5 (2-02-2002) instalación y más ejemplos

7.4 Autor del tutorial

Manuel de Vega Barreiro. barreiro@arrakis.es

Linux Landia. http://www.croftj.net/~barreiro

Este tutorial se escribio con motivo de mi participacion en la 3ª reunion de usuarios de Linux en Madrid. El tutorial esta todavia incompleto, aunque espero terminarlo pronto. (creo que no pondre mas comentarios de este tipo :-)

7.5 Derechos de autor

---

8. Anexo: ejemplos de programas Expect

8.1 ftp_1.exp

```
#!/usr/bin/expect -f
#
set force_conservative 0  ;# set to 1 to force conservative mode even if
                          ;# script wasn't run conservatively originally
if {$force_conservative} {
        set send_slow {1 .001}
        proc send {ignore arg} {
                sleep .1
                exp_send -s -- $arg
        }
}
set timeout 2
puts "\n"
#
spawn ftp localhost
expect -exact "Name (localhost:root): "
send -- "ftp\r"
expect "Password:"
send -- "barreiro@arrakis.es\r"
expect -exact "ftp> "
send -- "binary\r"
expect -exact "ftp> "
send -- "hash\r"
expect -exact "ftp> "
send -- "cd pub/spain/gnome\r"
log_user 1
expect -exact "ftp> "
send -- "ls\r"

puts "\n"
exit
```

8.2 ftp_21.exp

```
#!/usr/bin/expect -f
#
set force_conservative 1  ;# set to 1 to force conservative mode even if
                          ;# script wasn't run conservatively originally
if {$force_conservative} {
        set send_slow {1 .001}
        proc send {ignore arg} {
                sleep .1
                exp_send -s -- $arg
        }
}
set timeout 2
puts "\n"
#
spawn ftp ftp.desconocido.es
expect -exact "Name (localhost:root): "
send -- "ftp\r"
expect "Password:"
send -- "barreiro@arrakis.es\r"
expect -exact "ftp> "
```

```
send -- "binary\r"
expect -exact "ftp> "
send -- "hash\r"
expect -exact "ftp> "
send -- "cd pub/spain/gnome\r"
expect -exact "ftp> "
send -- "ls\r"

puts "\n"
exit
```

8.3 ftp_22.exp

```
#!/usr/bin/expect -f
#
set force_conservative 1  ;# set to 1 to force conservative mode even if
                          ;# script wasn't run conservatively originally
if {$force_conservative} {
        set send_slow {1 .001}
        proc send {ignore arg} {
                sleep .1
                exp_send -s -- $arg
        }
}
set timeout 2
puts "\n"
#
spawn ftp 12.0.0.2
expect -exact "Name (localhost:root): "
send -- "ftp\r"
expect "Password:"
send -- "barreiro@arrakis.es\r"
expect -exact "ftp> "
send -- "binary\r"
expect -exact "ftp> "
send -- "hash\r"
expect -exact "ftp> "
send -- "cd pub/spain/gnome\r"
expect -exact "ftp> "
send -- "ls\r"

puts "\n"
exit
```

8.4 ftp_31.exp

```
#!/usr/bin/expect -f
#
set force_conservative 1  ;# set to 1 to force conservative mode even if
                          ;# script wasn't run conservatively originally
if {$force_conservative} {
        set send_slow {1 .001}
        proc send {ignore arg} {
                sleep .1
                exp_send -s -- $arg
        }
}
set timeout 2
puts "\n"
#
spawn ftp ftp.desconocido.es
```

```
expect {
 -exact "conection refused" {puts "Conexion denegada\n";exit}
 -exact "Host name lookup failure" {puts "Servidor desconocido\n";exit}
 -exact "Name (localhost:root):*"

}
send -- "ftp\r"
expect "Password:"
send -- "barreiro@arrakis.es\r"
expect -exact "ftp> "
send -- "binary\r"
expect -exact "ftp> "
send -- "hash\r"
expect -exact "ftp> "
send -- "cd pub/spain/gnome\r"
expect -exact "ftp> "
send -- "ls\r"

puts "\n"
exit
```

8.5 ftp_32.exp

```
#!/usr/bin/expect -f
#
set force_conservative 1  ;# set to 1 to force conservative mode even if
                          ;# script wasn't run conservatively originally
if {$force_conservative} {
        set send_slow {1 .001}
        proc send {ignore arg} {
                sleep .1
                exp_send -s -- $arg
        }
}
set timeout 2
puts "\n"
#
spawn ftp 12.0.0.2
set timeout 10
expect {
 timeout {puts "El comando ha temporizado\n";exit}
 -exact "conection refused" {puts "Conexion denegada\n";exit}
 -exact "Host name lookup failure" {puts "Servidor desconocido\n";exit}
 -exact "Name (localhost:root):*"

}
send -- "ftp\r"
set timeout 5
expect "Password:"
send -- "barreiro@arrakis.es\r"
expect -exact "ftp> "
send -- "binary\r"
expect -exact "ftp> "
send -- "hash\r"
expect -exact "ftp> "
send -- "cd pub/spain/gnome\r"
expect -exact "ftp> "
send -- "ls\r"

puts "\n"
exit
```

8.6 ftp_41.exp

```
#!/usr/bin/expect -f
#
set force_conservative 1  ;# set to 1 to force conservative mode even if
                          ;# script wasn't run conservatively originally
if {$force_conservative} {
        set send_slow {1 .001}
        proc send {ignore arg} {
                sleep .1
                exp_send -s -- $arg
        }
}
set timeout 2
puts "\n"
#
spawn ftp localhost
expect "Name*:*"
send -- "ftp\r"
expect "assword:*"
send -- "barreiro@arrakis.es\r"
expect "ftp>*"
send -- "binary\r"
expect "ftp>*"
send -- "hash\r"
expect "ftp>*"
send -- "cd pub/spain/gnome\r"
expect "ftp>*"
send -- "ls\r"

puts "\n"
exit
```

8.7 ftp_51.exp

```
#!/usr/bin/expect -f
#
set force_conservative 1  ;# set to 1 to force conservative mode even if
                          ;# script wasn't run conservatively originally
if {$force_conservative} {
        set send_slow {1 .001}
        proc send {ignore arg} {
                sleep .1
                exp_send -s -- $arg
        }
}
puts "\n"
#
log_user 0
spawn ftp localhost
expect "Name*:*"
send -- "ftp\r"
expect "assword:*"
send -- "barreiro@arrakis.es\r"
expect "ftp>*"
send -- "binary\r"
expect "ftp>*"
send -- "hash\r"
expect "ftp>*"
send -- "cd pub/spain/gnome\r"
log_user 1
expect "ftp>*"
```

```
send -- "ls\r"

puts "\n"
exit
```

8.8 ftp_52.exp

```
#!/usr/bin/expect -f
#
set force_conservative 1  ;# set to 1 to force conservative mode even if
                          ;# script wasn't run conservatively originally
if {$force_conservative} {
        set send_slow {1 .001}
        proc send {ignore arg} {
                sleep .1
                exp_send -s -- $arg
        }
}
set timeout 2
puts "\n"
#
log_user 0
spawn ftp localhost
send_user "Conectando al servidor\n"
expect "Name*:*"
send -- "ftp\r"
expect "assword:*"
send -- "barreiro@arrakis.es\r"
expect "ftp>*"
send_user "Acceso concedido\n"
send -- "binary\r"
expect "ftp>*"
send -- "hash\r"
expect "ftp>*"
send_user "Cambiando al directorio elegido\n"
send -- "cd pub/spain/gnome\r"
log_user 1
expect "ftp>*"
send -- "ls\r"

puts "\n"
exit
```

8.9 ftp_61.exp

```
#!/usr/bin/expect -f
#
set force_conservative 1  ;# set to 1 to force conservative mode even if
                          ;# script wasn't run conservatively originally
if {$force_conservative} {
        set send_slow {1 .001}
        proc send {ignore arg} {
                sleep .1
                exp_send -s -- $arg
        }
}
set timeout 2
puts "\n"
#
log_user 0
if $argc<2 {
```

```
  send_user "$argv0: faltan parametros\n"
  exit
}
set sistema [lindex $argv 0]
set usuario [lindex $argv 1]
send_user "$sistema\n"
send_user "$usuario\n"
send_user "[lindex $argv 0]\n"
send_user "[lindex $argv 1]\n"


spawn ftp $sistema
send_user "Conectando al servidor\n"
expect "Name*:*"
send -- "ftp\r"
expect "assword:*"
send -- "$usuario\r"
expect "ftp>*"
send_user "Acceso concedido\n"
send -- "binary\r"
expect "ftp>*"
send -- "hash\r"
expect "ftp>*"
send_user "Cambiando al directorio elegido\n"
send -- "cd pub/spain/gnome\r"
log_user 1
expect "ftp>*"
send -- "ls\r"

puts "\n"
exit
```

8.10 ftp_71.exp

```
#!/usr/bin/expect -f
#
set force_conservative 1  ;# set to 1 to force conservative mode even if
                          ;# script wasn't run conservatively originally
if {$force_conservative} {
        set send_slow {1 .001}
        proc send {ignore arg} {
                sleep .1
                exp_send -s -- $arg
        }
}
set timeout 2
puts "\n"
#
log_user 0
spawn ftp localhost
expect "Name*:*"
send -- "ftp\r"
expect "assword:*"
send -- "barreiro@arrakis.es\r"
expect "ftp>*"
send -- "binary\r"
expect "ftp>*"
send -- "hash\r"
expect "ftp>*"
send -- "cd pub/spain/gnome\r"
log_user 1
expect "ftp>*"
interact "" return
send_user "Salgo de la zona interactiva\n"
```

```
send -- "cd ..\r"
expect -exact "ftp> "
send -- "ls\r"
expect -exact "ftp> "

puts "\n"
exit
```

8.11 telnet_gl.exp

```
#!/usr/bin/expect -f
set force_conservative 1  ;# set to 1 to force conservative mode even if
                          ;# script wasn't run conservatively originally
if {$force_conservative} {
        set send_slow {1 .001}
        proc send {ignore arg} {
                sleep .1
                exp_send -s -- $arg
        }
}
puts "\n"

spawn telnet localhost
expect "ogin:*"
send -- "mante\r"
expect "assword:*"
send -- "a12345\r"
expect {
timeout {exit}
-gl "\\\[*]\$*"
}
send_user "\nestoy en el directorio: "
send -- "pwd\r"
expect -gl "\\\[*]\$*"

puts "\n"
exit
```

8.12 telnet_re.exp

```
#!/usr/bin/expect -f
set force_conservative 1  ;# set to 1 to force conservative mode even if
                          ;# script wasn't run conservatively originally
if {$force_conservative} {
        set send_slow {1 .001}
        proc send {ignore arg} {
                sleep .1
                exp_send -s -- $arg
        }
}
puts "\n"

spawn telnet localhost
expect "ogin:*"
send -- "mante\r"
expect "assword:*"
send -- "a12345\r"
expect -re "\\\[.*]\\\$.*"
send_user "\nestoy en el directorio: "
send -- "pwd\r"
expect -re "\\\[.*]\\\$.*"
```

```
puts "\n"
exit
```

Next Previous Contents