TERM Como

Patrick Reijnen, patrickr@bart.nl
Traducción de Alfonso Belloso, alfon@bipv02.bi.ehu.es
v1.0, 1 Enero de 1995

Lo que sigue es una guía detallada de configuración del programa de comunicaciones `term` en Linux.

Índice General

1 Información

1.1 Copyright statement

1.2 DISCLAIMER

2 Introducción

2.1 Sobre este Documento

Este COMO intenta aclarar algunas de las confusiones al usar `term`, el gran programa de Michael O'Reilly que permite multiplexar tu línea serie y establecer una conexión de red. De principio a fin, los documentos que vienen con el `term` son bastante buenos, y este COMO no pretende reemplazarlos. La intención de este documento es dar una idea de fondo de cómo trabaja el `term` y detallar los pasos para conseguir algunos de los servicios de red más comunes trabajando bajo `term`. Se debe hacer hincapié en que este documento no cubre todo lo que se debe saber sobre el `term`. Después de leerlo, deberían leerse las *manpages* sobre `term`, ya que éstas incluyen información que no se encuentra aquí.

2.2 ¿Qué es el `TERM`?

`term` es un programa, escrito por Michael O'Reilly, `michael@iinet.com.au` , que corre sobre una línea serie para permitir a conexiones múltiples operar de forma concurrente –es decir, puedes estar recibiendo un fichero con tu módem mientras trabajas en un sistema remoto (distinto)– a través de la misma conexión módem. El `term` también se puede usar para abrir ventanas de cliente X sobre una conexión serie. Por medio de las utilidades `tredir` y `tupredir` de las versiones `2.0.x` y superiores del `term`, éste puede proporcionar muchos de los servicios de red "tradicionales":`mail, news, ftp, telnet, xarchie`, etc. En esencia, `term` es muy parecido a otros protocolos serie como `SLIP` o `PPP`. La ventaja de `term` es que puede correr enteramente desde el espacio de usuario, sin requerir soporte del kernel ni del sistema o administradores de red.

A diferencia de `SLIP` o `PPP`, tu máquina no llega a tener su propia dirección IP. Todo el tráfico deberá ir dirigido al host remoto, y será redirigido a tu máquina mediante `TERM`.

3 Cómo funciona el TERM

Antes de experimentar con `term` es altamente aconsejable leer primero este capítulo completo y el fichero `INSTALLATION` que viene con el paquete. También conviene echar una ojeada a las páginas de manual de `linecheck`, `(term)test` y `term`. Esto te ayudará a trabajar más fácil y más rápido.

3.1 Nomenclatura

Asumo que estás llamando a un sistema a través de algún tipo de servidor de terminal. Utilizo los términos *"local"* y *"remoto"* para referirme a los sistemas conectados en casa y en la red respectivamente (a no ser que los use para referirme a alguna otra cosa :-).

`term` proporciona a la máquina local, que no tiene conexión de red, pero que está conectada por una línea serie a una máquina remota, la cual a su vez está conectada a una red, servicios de red. Observemos cómo una máquina con una conexión de red "tradicional" proporciona estos servicios.

Primero el usuario invoca un programa, como `telnet` o `ftp`, que requiere un servicio de red. Lo que estos programas hacen es hacer una llamada del sistema solicitando servicios de red. El sistema operativo obtiene entonces estos servicios a través de su interface de red (por ejemplo, manda y recibe paquetes sobre la ethernet).

`SLIP` y `PPP` hacen exactamente esto, convirtiendo la línea módem en un interface de red, lo cual en principio no es diferente de una ethernet. La pega está en que estos protocolos hacen de la máquina conectada por módem parte de la red, justo como cualquier otra máquina. Esto exige toda la tarea administrativa asociada al hecho de ser un nodo de la red (más aún, ya que el enlace módem también hay que administrarlo).

En ausencia de una conexión de red como `SLIP` o `PPP`, ¿qué es lo que se hace típicamente?. Bien, llamas a tu máquina conectada a la red, lees tu correo, tus news, etc, si necesitas un fichero, primero te lo transfieres a la máquina remota y entonces te lo envías a la máquina local usando el `kermit` o algún otro programa de comunicaciones.

Esto es una pena, especialmente porque en realidad sólo puedes hacer que una cosa use el enlace módem a la vez. La idea que hay detrás del `term` es básicamente automatizar y *multiplexar* este proceso. El `term` se invoca en ambas máquinas, local y remota, y los dos procesos se comunican entre sí por la línea módem. Cuando necesitas un servicio de red, haces una solicitud al *daemon* del `term` local, el cual transmite la petición al *daemon* del `term` en la máquina remota (conectada a la red). El resultado se devuelve a través de la línea módem.

Para ser más precisos, pongamos que quieres conseguir un fichero por `ftp`. Primero necesitas una versión de `ftp` que pueda hablar con `term`. Invocas `termftp` como lo haces con un `ftp` normal, pongamos `'termftp nethost.gov'`, pero esta versión especial hace su solicitud de red al daemon del `term` local en vez de al kernel. El `term` local transfiere esta petición, a través de la línea del módem, al `term` remoto, el cual establece una conexión con `nethost.gov`, y transmite los datos de vuelta sobre el enlace módem.

`term` es lo suficientemente listo como para tener muchas cosas diferentes funcionando a la vez, por lo que puedes tener varias sesiones de red distintas usando el mismo enlace módem; por ejemplo puedes estar dentro de otra máquina lejana vía `termtelnet` mientras continúa la transferencia del `termftp`.

Si esto es demasiado abstracto (o engorroso) no te preocupes; la información importante que hay que extraer de esta sección es que hay *dos copias* del `term` corriendo, una a cada lado del enlace módem.

4 Poniendo a punto las cosas.

4.1 Qué necesitas tener de antemano.

Antes de comenzar a construir y usar el `TERM` debes asegurarte de que tienes incluido el soporte *TCP/IP* en el kernel. Además, asegúrate de que esté activo el interface *loopback* de *TCP/IP*. Si éste es tu caso, puedes seguir con el resto de esta sección.

4.2 Explicación de conceptos.

En las versiones nuevas del `term` han aparecido dos conceptos nuevos. Estos dos conceptos se explican en los siguientes apartados.

4.2.1 *"Sharing"* (compartición).

A partir de la versión 1.16 aparece el concepto de compartir la conexión `TERM` con otros usuarios. Esto significa que cuando habilitas la característica *"shared"* (compartida), más gente podrá usar la misma conexión `TERM` que tú estás usando, es decir, cuando estás trabajando en tu máquina remota a través de tu conexión `TERM` (has usado `trsh` en tu máquina local para acceder, por ejemplo) otra persona en tu máquina local podrá usar la misma conexión `TERM` al mismo tiempo para transmitir un fichero con `ftp` a su cuenta en tu máquina local desde un `ftp` *site* de cualquier lugar del mundo.

Cuando deshabilitas la característica *"shared"* (compartida) (o sea, ejecutas `TERM` en modo privado) tú y solo tú (sin contar a `root` :-) puede usar la conexión `TERM`.

Por supuesto, sólo necesitas instalar el *"shared"* `TERM` en el extremo en el que quieres permitir a la gente usar la misma conexión `TERM` que tú estés usando. Así que, si otra gente tiene cuenta en tu máquina local y quieren usarlo desde algún lugar de tu red remota habilitas la característica shared en el extremo remoto de tu conexión `TERM`. De esta forma toda esta gente puede acceder a tu máquina a la vez compartiendo la misma conexión `TERM` entre sí y contigo mismo. (NOTA: el primer ejemplo necesitaba habilitar la característica shared en el extremo local de la conexión `TERM`).

NOTA para la instalación como `root`:

Cuando instalas `TERM` como `root`, primero debes crear un grupo `TERM` (antes de compilar) sin miembros, añadiendo la siguiente línea en `/etc/group`:

```
term::16:root
```

o cualquier otro `GID` no usado en lugar del 16 si éste ya está en uso.

Después de compilar e instalar coloca al `TERM` y sus clientes el `SGID` *term*:

```
chgrp term <cliente_term>
chmod g+s <cliente_term>
```

También cualquier programa que hagas utilizable con `TERM` debe tener SGID `TERM`.

4.2.2 "*Full TERM networking*" (conexión de red completamente TERM)

A partir de la versión `2.0.0` de `TERM` se usa el concepto *full TERM networking* (conexión de red totalmente TERM). Cuando tu única conexión con el mundo exterior es una conexión `TERM`, tienes una red *full TERM* (completamente TERM) y deberías compilar el `TERM` con full `TERM` networking. En este caso se ha puesto un fichero llamado `termnet` en el directorio compartido. Esto dice a `TERM` que tu única conexión con el exterior es a través del `TERM`.

Cuando también tienes algún otro tipo de conexión de red además de los programas (pasados a `TERM`) de la conexión `TERM`, primero intenta que realicen su trabajo usando esta otra conexión. Si ésta falla entonces se invoca el `TERM` y se intenta realizar el trabajo a través de la conexión `TERM`. Para aclarar esto, ahora se da un ejemplo en el que el se usa `telnet` hecho utilizable en `TERM`. Este `telnet` debería funcionar con y sin `TERM`.

```
telnet localhost
```

no usa el `TERM` para conectar, pero

```
telnet zeus.cs.kun.nl
```

usará el `TERM` sólo si no tienes otro tipo de conexión de red.

El full TERM networking también implica el mentir sobre el nombre del host local, diciendo que es el host remoto en su lugar. Además, provoca que `bind (0)` actúe siempre en el host remoto. En esencia hace que muchos programas puedan usarse sin ir a través del `TERM` mientras el `TERM` está corriendo.
Desafortunadamente, la mayoría de programas y demonios `UDP` no funcionarán con `TERM` sin estos incómodos trucos.

4.3 Compilación del TERM

Si tienes suerte esto sólo debe implicar un `make`. Sin embargo, lo más probable es que necesites hacer más. Debido a nuevas opciones en las versiones nuevas del `TERM` (`sharing`, `configure`) ahora es un poco más complicado crear el ejecutable del `TERM`. Hoy en día pueden seguirse un par de caminos para obtener el ejecutable.

Para cubrir todos los caminos a seguir con los que puede construirse el `TERM` vamos a dividir esta sección en tres partes:

1. Construir el `TERM` hasta la versión 1.15
2. Construir el `TERM`, versiones 1.16 hasta 1.19
3. Construir el `TERM`, versiones 2.0.0 y superiores

4.3.1 Construir el term hasta la versión 1.15

Para estas versiones del `term`, la compilación no debería implicar ejecutar más que estos comandos

```
make DO=install OS-type
make installman
```

Encontrarás el `term`, sus clientes y las páginas de manual (`man`) cómodamente construidas e instaladas y listas para usar después de esto.

Además, necesitarás crear un directorio `$HOME/term`. Este directorio lo usará el `term` para buscar su fichero `termrc`.

La única cosa que puedes querer hacer es cambiar algunos de los paths en el `Makefile`, o cambiar alguna opción del compilador.

4.3.2 Construir el `term`, versiones 1.16 hasta 1.19

Para construir el `term` ahora puedes elegir una de las siguientes formas:

1. Como un usuario normal, construir el `term` en modo privado
2. Como un usuario normal, construir el `term` en modo shared
3. Como `root`, construir el `term` en modo privado
4. Como `root`, construir el `term` en modo shared

Abajo se explicará cómo habilitar/deshabilitar la opción `shared` durante la compilación del `term`

1. Eres un usuario normal (sin acceso de `root`) y NO quieres COMPARTIR (SHARE) la conexión `term` con otros usuarios.

   Como usuario que no quiere compartir la conexión `term` con otros usuarios deberías hacer lo siguiente para construir el `term`:

   ```
   make DO=install OS-type
   make installman
   ```

   Después de esto, el `term`, sus clientes y las páginas de manual están generados e instalados.

   Además, necesitarás crear un directorio `$HOME/term`. Este directorio lo usará el `term` para buscar su fichero `termrc`.

   La única cosa que puedes querer hacer es cambiar algunos de los paths en el `Makefile` o cambiar algún parámetro del compilador.

2. Eres un usuario normal (sin acceso de `root`) y SI quieres COMPARTIR (SHARE) la conexión `term` con otros usuarios.

   Como usuario que quiere compartir la conexión `term` con otros usuarios deberías hacer lo siguiente para construir el `term`:

   ```
   make DO=installshare USERSHARE=$HOME/term OS-type
   make installman
   ```

   Después de esto, el `term`, sus clientes y las páginas de manual estarán compiladas e instaladas.

   Además, tendrás un directorio `$HOME/term` (por defecto) con permisos `drwxrwxr-x`. En este directorio encontrarás al menos el *socket* usado por el `term` para sus conexiones (`tmp/private/socket=`).

3. Eres `root` y NO quieres COMPARTIR (SHARE) la conexión `term` con otros usuarios.

   Como `root` que no quiere compartir la conexión `term` con otros usuarios deberías hacer lo siguiente para construir el `term`:

   ```
   make DO=install OS-type
   make installman
   ```

   Después de esto, el `term`, sus clientes y las páginas de manual estarán compiladas e instaladas.

   Además, tendrás un directorio llamado `/usr/local/lib/term` (por defecto) con permisos `drwxr-xr-x`. En este directorio encontrarás al menos el *socket* usado por el `term` para sus conexiones (`tmp/private/socket=`).

4. Eres `root` y quieres COMPARTIR (SHARE) la conexión `term`.

   Primero, asegúrate de haber leído la sección sobre “sharing” anterior.

   Como `root` que quiere compartir la conexión `term` deberías hacer lo siguiente:

```
make DO=installshare OS-type
make installman
```

Después de esto, el `term`, sus clientes y las páginas de manual estarán compiladas e instaladas.

Además, tendrás un directorio llamado `/usr/local/lib/term` (por defecto) con permisos `drwxrwxr-x`. En este directorio encontrarás al menos el *socket* usado por el `term` para sus conexiones (`tmp/private/socket=`).

4.3.3 Construir el TERM, versiones 2.0.0 y superiores

Primero asegúrate de haber leído la sección sobre "full term networking" de arriba.

Para las versiones de `term 2.0.0` y superiores hay muchas formas de generar el binario del `term` y los clientes. Todo esto puede hacerse tanto por `root` como por un usuario cualquiera:

1. Generar el `term` en modo privado sin full term networking
2. Generar el `term` en modo privado con full term networking
3. Generar el `term` en modo shared sin full term networking
4. Generar el `term` en modo shared con full term networking

En estas versiones de `term` ha aparecido una nueva forma de compilar, usando el script `configure`. Cuando se ejecuta `configure` éste chequea en que sistema operativo estás tratando de instalar el `term`, si el directorio origen está disponible o no, y si hay puesta alguna opción *runtime*. De acuerdo con las cosas encontradas `configure` crea entonces un `Makefile` usando `Makefile.in` que se entrega con el paquete del `term`.

Dos de las opciones más importantes para `configure` son `--root` y `--user` que establecen si el `term` será instalado por `root` o por un usuario cualquiera. Se pueden usar otras opciones para instalar el `term` de la forma que quieras (con paths no estándar, por ejemplo).

1. Generar el `term` en modo privado sin full term networking

   Para generar el `term` de este modo necesitas ejecutar los siguientes comandos (tanto para `root` como para cualquiera):

   ```
   make install installman
   ```

   Esto genera los binarios, instalándolos junto con las páginas de manual.

2. Generar el `term` en modo privado con full term networking

   Para generar el `term` de este modo necesitas ejecutar los siguientes comandos (tanto para el `root` como para cualquiera):

   ```
   make installnet installman
   ```

   Esto genera los binarios, instalándolos junto con las páginas de manual.

3. Generar el `term` en modo shared sin full term networking

   Para generar el `term` de este modo necesitas ejecutar los siguientes comandos (tanto para el `root` como para cualquiera):

   ```
   make share installman
   ```

   Esto genera los binarios e instala estos binarios y las páginas de manual.

4. Generar el `term` en modo shared con full term networking

   Para generar el `term` de este modo necesitas ejecutar los siguientes comandos (tanto para el `root` como para cualquiera):

   ```
   make share installnet installman
   ```

   Esto genera los binarios e instala estos binarios y las páginas de manual.

4.4 `client.a, libtermnet.a, libtermnet.sa, libtermnet.so`

Con el `term` se suministra una librería con funciones para clientes `term`.

Hasta la versión 1.16 esta libreria se llamaba `client.a`. Durante la compilación de `term` se generaba esta librería, que después se usaba en la compilación de los clientes `term`. No se instalaba en otro directorio.

A partir de la versión 1.16 se cambió el nombre de la librería por `libtermnet.a`. Hasta la versión 1.19 esta librería se crea en el directorio `term` y después se usa durante la compilación de los clientes `term`. No se instala en otro directorio.

A partir de la versión 2.0.0, además de `libtermnet.a` también se crean `libtermnet.so` y `libtermnet.sa` (libreria *shared* y librería *exported initialized*) durante la instalación del paquete `term`. Durante la instalación de todas las partes del paquete, estos tres ficheros de librerías se instalan en el directorio `/usr/local/lib` (por defecto). Y después se hace un enlace desde `libtermnet.so.2` a `libtermnet.so.2.x.x`.

Finalmente se ejecuta `ldconfig` para crear los enlaces necesarios y el caché (para uso del enlazador dinámico, `ld.so`) para las librerías compartidas más recientes que se encuentran en los directorios especificados en la línea de comandos, en el fichero `/etc/ld.so.conf`, y en los directorios permitidos (`/usr/lib` y `/lib`).

Si la instalación se hace correctamente, los tres ficheros de librería podrán ser usados por los clientes de `term` que son generados con librerías dinámicas en lugar de estáticas. Estas librerías también se pueden usar ahora para portar tu software propio a fin de poderlo usar con el `term` (ver más adelante).

4.5 Estableciendo variables de entorno.

`term` reconoce un par de variables de entorno que pueden definir los usuarios. Las tres primeras de éstas que se explicarán son:

- `TERMDIR`
- `TERMSHARE`
- `TERMMODE`

Definiendo estas variables puedes controlar el modo en que se ejecuta el `term`.

Para versiones del `term` hasta la 1.15 sólo es importante la variable `TERMDIR` (estas versiones no reconocen el modo *shared*). Para estas versiones `TERMDIR` se debería definir como sigue:

```
setenv TERMDIR $HOME     # csh o tcsh
export TERMDIR=$HOME     # bash
```

A partir de la versión 1.16 `term` también reconoce las variables `TERMSHARE` y `TERMMODE`. Con estas variables se le puede indicar al `term` que funcione en modo privado o en modo compartido. Explicaremos como definir las variables para ambos modos.

1. Ejecutar `term` en modo privado puede hacerse definiendo las variables `TERMDIR` y `TERMMODE` de la siguiente forma:

   Para `csh` o `tcsh`:

   ```
   setenv TERMDIR $HOME
   setenv TERMMODE 0
   ```

   Para `bash`:

   ```
   export TERMDIR=$HOME export TERMMODE=0
   ```

2. Si quieres usar el `term` en modo compartido hay dos formas de definir las variables:

   (a) Si `term` se instala como un programa `SUID` sólo se debe definir `TERMMODE`. (`README` en el paquete del `term`).

   ```
   setenv TERMMODE 'numero'     # csh o tcsh
   export TERMMODE='numero'     # bash
   ```

   En `'numero'` debe ponerse 1 si se está usando una versión del `term` entre la 1.16 y la 1.19 (`README.share` en el paquete del `term`) y 2 si se está usando la versión de `term 2.0.0` o superior (`README.security` en el paquete del `term`).

   (b) Si `term` se instala como un programa `SGID` las variables se deben definir del siguiente modo:

   Para `csh` o `tcsh`:

   ```
   setenv TERMMODE 1
   setenv TERMDIR /usr/local/lib/term
   setenv TERMSHARE $TERMDIR
   ```

   Para `bash`:

   ```
   export TERMMODE=1
   export TERMDIR=/usr/local/lib/term
   export TERMSHARE=$TERMDIR
   ```

   Poniendo las variables de esta forma hará posible que se ejecuten clientes viejos (*linkados* con una versión antigua del `client.a`) en modo compartido (shared).

A partir de la versión `2.0.0`, `term` también reconoce la variable `TERMSERVER`. Se necesita definir esta variable cuando se tienen varios módems y mas de una conexión a la vez. Para especificar que conexión usar, se debe ejecutar el `term` con un nombre de servidor:

```
nohup term -v /dev/modem1 Connection1 & nohup term -v /dev/modem2 Connection2 &
```

Los usuarios deberían definir la variable `TERMSERVER` con el nombre de conexión que quieran usar:

```
setenv TERMSERVER Connection1           # csh o tcsh
export TERMSERVER=Connection2           # bash
```

4.6 Probar el TERM

Haz un `make test` (o `make termtest` para versiones nuevas del `term`) para generar el demonio de prueba del `term`. `(term)test` funciona ejecutando dos copias del `term` en tu sistema, una "local" y una "remota". Ambas leerán tu `termrc`; de modo que puedas ajustar su comportamiento. Ahora ejecuta `(term)test`. Deberías poder hacer un `trsh` y un `tupload`. Prueba con:

```
tupload ./term /usr/tmp
```

deberías de conseguir tener una copia del binario `term` en `/usr/tmp`). La salida del `term` local debería aparecer en `local.log` y la remota en `remote.log`. Puedes ejecutar `term` con el parámetro `-d255` para poder registrar lo que ocurra en estos ficheros, o habilitar el *debugging* en tu `termrc`.

NOTA: Ejecuta el `test` como `./test` para evitar el `test` del sistema.

4.7 `TERM` y los programas de comunicaciones.

Antes de poder usar `term`, tienes que establecer una conexión vía módem usando un programa de comunicaciones como `kermit` o `seyon`. En la documentación del programa encontrarás qué tienes que hacer para conectarte con la máquina remota.

Cuando hayas establecido la conexión y quieras ejecutar `term`, necesitas suspender o salir del programa de comunicaciones sin cerrar la conexión.

A continuación explicaré cómo hacer esto con algunos programas de comunicaciones.

4.7.1 `kermit`

Iniciar `term` cuando se usa `kermit` es fácil. En el *prompt* local de `kermit` se teclea "*suspend*", de modo que volverás al prompt de Linux. Desde este prompt puedes ya establecer la conexión `term`.

4.7.2 `seyon`

Una forma de iniciar chequeo de línea o `TERM` cuando usas `seyon` es poner `linecheck` y `TERM` en el menú `Transfer` (controlado por el fichero `$HOME/.seyon/protocols`).

Añade al fichero `$HOME/.seyon/protocols` lo siguiente:

```
"Line check" "$cd /tmp; linecheck"
"Term" "$term -c off -w 10 -t 150 -s 38400 -l $HOME/tlog"
```

Ahora, cuando inicies `linecheck` o `term` en la máquina local, selecciona en el menú `Transfer` el ítem "`Line Check`" o "`Term`".

Por supuesto, también podrías usar el botón de comando de shell y teclear en el cuadro de diálogo que se te abrirá los comandos `linecheck` o `term`. Esto mismo hace redirección automática al comando.

4.8 Hacer un enlace transparente.

Presumiblemente, puedes establecer una conexión módem entre tus hosts local y remoto. Típicamente lo que haces es, llamar a algún tipo de servidor de terminales y conectas con tu host remoto a través de él.

También sueles usar para ello software de terminal, como `kermit` o `seyon` para comunicar con tu módem (los ejemplos de este documento usan el `kermit`, ya que es el que usa el autor). Si estás teniendo problemas con el módem, o con el software de terminal, echa una ojeada al *Serial HOWTO*; esto debería ayudarte.

Una vez establecido el enlace, querrás hacerlo lo más transparente posible. Comprueba los comandos en el servidor de terminal (`help` o `?` suele ser un buen comienzo). Busca la opción *8 bits* siempre que sea posible. Esto puede implicar cambiar la forma en que accedes a tu sistema, por ejemplo, si el servidor usa `rlogin`, tendrás que usarlo poniendo el parámetro `-8` para hacerlo transparente.

Especialmente vigila el control de flujo por *xon/xoff*. No lo necesitas. Intenta habilitar el control de flujo por hardware, *rts/cts*. Puede que tengas que mirar la documentación de tu módem para saber cómo configurarlo para hacer comunicaciones *rts/cts* de 8 bits.

4.9 Ejecutar `linecheck`

ATENCIÓN: En algunos documentos las opciones de línea de comandos para `linecheck` se citan en orden incorrecto. He comprobado esto y he hallado que el orden de las opciones que menciono aquí abajo es el correcto.

NOTA: a partir de la versión de `term` 2.3.0 `linecheck` ya no necesitará tener el nombre del fichero de log en la línea de comandos. Escribirá su salida al fichero `linecheck.log` en el directorio desde el que se ejecute `linecheck`.

`linecheck` es un programa que se entrega con el `term`. Comprueba la transparencia de un enlace, produciendo información de configuración que necesita el `term` para funcionar correctamente. `linecheck` manda todos los 256 caracteres de 8 bits posibles sobre el enlace y verifica que cada uno se transmite correctamente.

Hay que configurar `term` para manejar caracteres que no pueden ser transmitidos por el enlace, y `linecheck` determina cuáles son estos caracteres. Debes usar `linecheck` después de haber establecido un enlace módem lo más transparente posible. Para correr `linecheck` haz lo siguiente:

1. En el sistema remoto, ejecuta `linecheck linecheck.log`
2. Vuelve a tu sistema local y suspende tu programa de comunicaciones (`^Z` en `kermit`) (si no te robará caracteres del `linecheck`).
3. En el sistema local ejecuta

   ```
   linecheck linecheck.log > /dev/modem < /dev/modem
   ```

Cuando termine `linecheck` encontrarás un conjunto de números al final de los ficheros `linecheck.log`. Estos son los que deberías poner como 'escape' en el `termrc` al otro lado del enlace. Por ejemplo, mi `linecheck.log` local decía que 'escape' era el 29 y 157. Así que, mi `termrc` local escapa (evita) estos caracteres y mi `termrc` remoto ninguno. Si se escapa (evita) un carácter en un extremo, también debo ignorarlo (ignore) en el otro; así que, en este ejemplo, debería ignorar 29 y 157 en mi sistema remoto.

Si `linecheck` se cuelga, prueba a usar

```
linecheck linecheck.log 17 19
```

en el sistema remoto, y

```
linecheck linecheck.log 17 19 > /dev/modem < /dev/modem
```

en el sistema local. Esto evitará tus caracteres de *xon/xoff* (control de flujo), que colgarán tu línea si tienes control de flujo por software. Si soluciona los problemas de cuelgues, tendrás que escapar/ignorar 17/19 en ambos `termrc`. Si tu servidor de terminal tiene otros caracteres que lo cuelguen, prueba a correr `lineckeck` con esos caracteres escapados como el ejemplo de arriba. Puedes marcar esos caracteres si `linecheck` se cuelga. Si es este el caso, mátalo, y luego mira en los logs. Los últimos caracteres transmitidos es probable que sean los culpables. Vuelve a intentarlo escapando estos caracteres.

En resumen, mi `termrc` local tiene las líneas:

```
escape 29
escape 157
```

y mi `termrc` remoto tiene las líneas:

```
ignore 29
ignore 157
```

ya que mi `linecheck.log` remoto decía que 'escape 29 y 157'.

4.10 Prueba a correr el TERM

Accede a tu sistema remoto, haciendo el enlace lo más transparente posible (si no lo has hecho aún). Arranca `term` en el extremo remoto. Yo lo hago así:

```
exec term -r -l $HOME/tlog -s 38400 -c off -w 10 -t 150
```

Vamos a desgranar las opciones una por una (ten en cuenta que con igual facilidad podría poner estas opciones en mi `termrc`. Lo hice así porque evitaba tener que editar un fichero mientras ponía a punto el `term`).

`exec` implica destruir la *shell* actual, corriendo en su lugar el programa indicado. Ejecuto las cosas con `exec` porque no pretendo usar mi *shell* de login de nuevo; estaría desperdiciando memoria. Si estás depurando el enlace y eres capaz de abortar el `term` remoto, puede que no quieras hacer el `exec`.

La opción `-r` es necesaria sólo en un extremo. `term` verá a este extremo como el remoto de la conexión (ten en cuenta que el extremo remoto del `term` puede ser tu propia máquina local). Si no usas esta opción en un extremo los clientes de `term` se caerán espontáneamente.

`-l $HOME/tlog`. Esto registra los errores en el fichero `tlog` en mi directorio *home*. Muy útil para depurar. No hay razón para no ponerlo.

`-s 38400` : Tengo un módem 14400, con compresión. Para ratios de compresión óptimos, necesito poder mandar bits al *pipe* lo más rápido posible. Para un módem más lento, debería ser un número menor. Fíjate que si tienes una *UART* 16450 en tu puerto serie, las velocidades altas pueden provocar pérdida de datos por desbordamiento del chip de tu puerto serie. `term` se recuperará de esto, pero si ves muchos mensajes en tu log, (o recibes *"overrun warnings"* en versiones del kernel `0.99pl15` en adelante) necesitarás reducir este número.

`-c off` : Desactiva la compresión de datos. Tengo un módem con compresión, y no necesito comprimir las cosas dos veces.

`-w 10 -t 150` : De nuevo estas son opciones para optimizar mi enlace de módem rápido. Pongo mi *ventana* a 10 y mi *timeout* a 150. Esto lo hago de acuerdo con la recomendación de la página de manual de `term_setup`.

Vuelve sobre tu máquina local y suspende tu programa de comunicaciones (`^Z` en `kermit`). No querrás que esté corriendo a la vez que `term`, ya que lucharía con el `term` por el puerto serie. Si puedes convencer a tu módem de que no cuelgue cuando salgas de tu programa de comunicaciones (cuando cambia el `DTR`), podrías salir del programa en este punto.

Ahora lanza el `term` local. Yo uso:

```
term -c off -l $HOME/tlog -s 38400 -w 10 -t 150 < /dev/modem > /dev/modem &
```

Necesito decirle al `term` donde está el módem; Así que apunto tanto la entrada como la salida estándar a `/dev/modem` (eso es lo que hacen < y >). También lo hago correr en *background*; de modo que puedo usar esta consola para otra cosa si la necesito.

`term` debería de funcionar ya `:-)`. Prueba con `trsh`, a ver qué pasa. Si se cuelga, o el enlace parece lento, echa una ojeada a tu `tlog` en ambos extremos. ¿Tienes *timeouts* o mensajes de error? Si es así, entonces es que has configurado algo mal. Vuélvelo a intentar (después de que hayas terminado de leer esto `:-)`. Observa que la conexión no parecerá muy rápida, especialmente si usas compresión - será un poco a saltos. La velocidad real aparece durante transmisiones de ficheros y similares.

4.11 Terminar tu conexión `term`

Muy probablemente, una vez que hayas hecho un montón de trabajo usando el `term`, querrás terminar el trabajo y deshacer tu conexión `term`. Para poder hacer esto hay cuatro formas:

1. Matar (`kill`) los programas `term` a ambos lados de la conexión. Esta es la forma menos recomendada de terminar tu conexión.
2. Una forma mejor es ejecutar el siguiente comando localmente:

   ```
   echo '00000' > /dev/modem
   ```

   Esto terminará la conexión `term` correctamente. Funcionará en todas las versiones del `term`. Hay que asegurarse de que la secuencia contiene al menos cinco ceros.
3. En el `termrc` de versiones 2.0.0 y superiores se puede incluir una sentencia `terminate <cualquier cadena>`. Esto establece una cadena que hará que salga del `term` ("00000" por defecto). Debe ser de una longitud de 5 caracteres por lo menos, para evitar terminaciones accidentales.
4. A partir de la versión 1.14 existe el programa `tshutdown` (realmente para la versión 1.14 está disponible como *patch*, para versiones más recientes se incluye en el paquete). Ejecutando `tshutdown`, la conexión `term` finalizará perfectamente.

4.12 Cómo eliminar `TERM` de tus particiones.

Ok, me habéis preguntado por esto. Así que voy a presentar los pasos a seguir para desinstalar `term`:

- Eliminar directorios con su contenido. Dependiendo de cómo hayas instalado `term`, tendrás uno o más de los siguientes directorios:

  ```
  $HOME/.term/termrc
  $HOME/.term/termrc.<servidor>
  $HOME/term/termrc
  $HOME/term/termrc.<servidor>
  /usr/local/lib/term/termrc
  /usr/local/lib/term/termrc.<servidor>
  /etc/termrc
  /etc/termrc.<servidor>
  ```

  Estos directorios pueden eliminarse con su contenido, usando

  ```
  /bin/rm -rf
  ```

- El grupo *term*. En algún momento de la instalación tuviste que crear un grupo llamado *term*. Busca en `/etc/group` una línea con ese grupo y bórrala.
- Los ejecutables y paquetes del `term`. Esta es la parte más difícil. Para los ejecutables, busca en el directorio `/usr/local/bin` o `$HOME/bin`.

  Con otros ejecutables que hiciste compatibles con `term` no puedo ayudarte. Necesitas saber qué ejecutables modificaste para saber cuáles borrar. No olvides ficheros de configuración y otros que vengan con esos ejecutables.
- Ficheros de librería. Lo mejor es teclear lo siguiente:

  ```
  cd /
  find . -name libtermnet* -exec /bin/rm {} \;
  ```

Esto encontrará y borrará todos los ficheros de librería relacionados por todo tu disco duro.

- Ficheros `include`. De nuevo, lo mejor es que teclees esto:

```
cd /
find . -name termnet.h -exec /bin/rm {} \;
```

- Manuales en línea. Cuando instalaste `term` los manuales se colocaron en uno de los directorios siguientes:

```
/usr/local/man/man1
/usr/local/man/cat1
$HOME/man/man1
$HOME/man/cat1
```

Debes buscar por lo menos las siguientes páginas de manual: `term`, `term_clients`, `term_setup`, `tdownload`, `linecheck`, `trdate`, `trdated`, `termrc`, `termtest`, `tmon`, `tredir`, `trsh`, `tshutdown`, `tudpredir`, `tupload`, `txconnand` y por último, `tiptest`.

- Directorio temporal del usuario. Borra el directorio `/usr/tmp/private` y sus contenidos.

Después de todo este ejercicio, podrás estar bastante seguro de que has borrado todo lo relacionado con `term`.

4.13 Optimización de la conexión

Una vez que consigues ejecutar el `term`, puede que quieras intentar optimizar las cosas. Una buena forma de medir la velocidad de tu enlace es corriendo `tmon` en una ventana mientras transfieres un fichero en otra. Intenta con ficheros de texto y ficheros comprimidos suficientemente grandes; el texto a secas debería dar un factor del doble de rápido que el comprimido. Los parámetros que querrás ajustar son `baudrate` [velocidad del puerto] (`-s`), `compression` (`-c`), `windows` [tamaño de ventana] (`-w`), `timeout` [tiempo de espera] (`-t`) y `retrain` (`-A`).

Cuidado con el parámetro `retrain`. Con la versión 1.19 del `term` obtengo peor rendimiento, que va de un 80% a un 90% comparado con la ejecución del `term` sin el parámetro `retrain`. No está claro si se trata de un *bug* en la versión 1.19 y si este problema existe sólo en la versión 1.19 del `term`.

`Baudrate`: el número máximo de bits por segundo que el `term` intentará enviar a través del enlace serie. `term` evitará enviar caracteres a una velocidad superior a ésta. Por defecto se usa la velocidad del puerto serie del ordenador, pero hay que avisar que ésta puede ser demasiado alta si el módem funciona a una velocidad menor sobre la línea telefónica. La opción `baudrate` está indicada para sistemas que almacenan en *buffer* la salida al módem. Durante la configuración y el ajuste es mejor usar un `baudrate` pequeño que uno que sea demasiado grande. Para enlaces de alta velocidad (> 38400), ponerlo sin límite será probablemente ventajoso. Esto se consigue usando el valor *off*. `term` confiará entonces solamente en el kernel para controlar el flujo.

`Compression`: necesitarás poner en *on* si no tienes un módem con compresión. Si tienes un módem de éstos, pon *compression off*, de lo contrario estarás comprimiendo las cosas dos veces, lo que habitualmente *incrementa* la cantidad de datos transmitidos. Los módems con compresión son aquellos que usan los protocolos *MNP-5* o *V42.bis*. Observa la documentación del módem y el mensaje del módem cuando conecta.

`Windows`: este es el número de unidades de datos, o paquetes, que el `term` enviará por la línea antes de obtener reconocimiento (*ack*) desde el `term` remoto. Para módems rápidos, aumentar esto puede ser una mejora; para enlaces más lentos esto puede saturar el extremo remoto.

`Timeout`: el tiempo que el `term` esperará a un *ack*. Si has aumentado `windows` y estás teniendo `timeouts` en el fichero de log, prueba a aumentar este valor.

Para un *14400/V42.bis*, yo uso `-c off -w 10 -t 150`. Consigo unos 1700 *cps* en ficheros comprimidos y 3500 *cps* en ficheros ASCII usando `tupload`.

4.14 Resolución de problemas

En esta sección se dan algunas opiniones sobre qué comprobar cuando se tienen problemas ejecutando el `term` o alguno de sus clientes.

- ¿Has borrado la estructura de directorios del `term`? En las nuevas versiones del term ha cambiado un par de veces la estructura del árbol de directorios bajo `/usr/local/lib/term`. Si no te has dado cuenta de ello, puedes haber causado todo tipo de mensajes de error. Lo mejor es borrar el árbol de directorios bajo `/usr/local/lib/term` (salvando tu `termrc`) y entonces instalar la nueva versión. De esta forma evitas pelearte con un árbol de directorios lioso.
- ¿Borraste los *sockets* antiguos? Cuando pongas al día tu versión del `term` borra todos los *sockets* (llamados `socket=`) creados por el `term`. El no hacer esto puede causar extraños problemas. Para averiguar qué *socket* está atendiendo el `term`, puedes usar el programa `netstat`.
- ¿El `term` no compila correctamente en SunOS 4.1.3? Has configurado el `term` con `./configure --user`. Durante la compilación estás teniendo un error de ensamblador, relacionado con que el parámetro `-k` no es reconocido. La razón de este error es desconocida. La solución es configurar el `term` con librerías estáticas. Es decir, tendrás que hacer `./configure --user --static` y entonces seguir con el proceso de compilación como sueles hacerlo normalmente. Ahora el `term` debería compilar correctamente.
- ¿El `termtest` te está presentando el error:

  ```
  Term: failed to connect to term socket '/root/.term/sockettest'
  ```

  (Term: falla la conexión al socket de `term '/root/.term/sockettest'`)

  Cuando `termtest` corre espera que el ejecutable `term` esté en el mismo sitio que éste. Cuando haces un `make install` antes de ejecutar `termtest`, el binario de `TERM` es movido a `/usr/local/bin` (u otro similar).

  Para ello hay que hacer algo parecido a esto:

  ```
  ln -s /usr/local/bin/term /usr/src/term-<numero_de_version>/term
  ```

- ¿Estás usando el binario adecuado? El `term` se ha modificado bastante, y muchos sistemas tienen versiones diferentes de los programas rondando por ellos. Asegúrate de que estás usando la versión correcta. Esto también se aplica al `linecheck`. Puede usarse una orden del tipo `bash -a`, o el comando `whereis` para saber qué programa se está ejecutando. Las versiones de `term` posteriores a 1.11 deberían mostrar su número de versión al comenzar. (Aunque la versión 1.14 dice ser la 1.12. Sigh.)
- ¿Tienes el `termrc` correcto en el lugar adecuado? Dependiendo de la versión de `term` que estés corriendo y el modo en que instalaste el `term` (siendo `root` o usuario) este fichero debe estar en uno de los siguientes directorios:

  ```
  /usr/local/lib/term/termrc
  /usr/local/lib/term/termrc.<servidor>
  /etc/termrc
  /etc/termrc.<servidor>
  ```

Algunos sistemas tienen `termrc`'s preinstalados; asegúrate de que no estén antes de instalar. Si estás ejecutando cosas como `root`, busca en `/.term`.

El `term` crea ficheros (*sockets* en realidad) mientras se ejecuta; de modo que tiene su propio directorio, `~/.term`, donde está el fichero `termrc` (nótese, ¡no hay un punto precediendo a `termrc`!).

- ¿Encuentra el `term` su fichero `termrc`? Cuando inicias `term` a ambos lados, deberías ver mensajes como este de abajo:

```
Term version: 2.2.9
Reading file:  /usr/local/lib/term/termrc
Using shared mode.
```

  Cuando falta la segunda línea el `term` no puede encontrar su fichero `termrc` y sabes que algo ha ido mal durante la instalación (a no ser que no estés usando el fichero `termrc` y estés introduciendo todas las opciones en línea de comandos `:-)`. Comprueba la localización y los permisos del fichero `termrc` en el lugar donde `term` no puede encontrar su fichero `termrc`.

- ¿Es correcta la sintaxis de las entradas en el `termrc`? Un problema común es que la gente que necesite usar el escape e ignorar ciertos caracteres, los introduzca así en el fichero `termrc`:

```
escape 1,4,30,255
ignore 1,4,30,255
```

  Aunque si `term` no reconoce lo anterior tampoco avisará de ello. Simplemente lo ignorará.

  Cuando tienes que ignorar o "escapar" caracteres, tendrás que ponerlos en diferentes líneas del fichero `termrc`, cada línea comenzando por la palabra `escape` o `ignore`. Sólo cuando haya que hacerlo con varios caracteres deberá hacerse de la siguiente forma:

```
escape 16-19      # escapar caracteres 16, 17, 18, 19
escape 23         # 23
escape 255        # ...y 255
ignore 16-19      # ignorar caracteres 16, 17, 18, 19
ignore 23         # 23
ignore 255        # ...y 255
```

- ¿Está montado tu directorio `term` o `.term` con *NFS*? Si tu directorio `term` o `.term` está montado con *NFS* necesitas poner el parámetro `-DTERM_NFS_DIR` en la línea `CFLAGS` del `Makefile`. Aunque, al autor el usar este parámetro le produce un error de compilación al compilar `term` 1.19 en una máquina con *SunOS 4.**.

- ¿Pertenecen los ficheros y directorios al usuario y grupo correcto y tienen los permisos adecuados? Esto no debería ser problema ya que estos permisos se ponen durante la fase de instalación. Sin embargo, cuando portas tus propios programas a `term` debes prevenirte de ello. También cuando se cambia el modo en que esta trabajando `term` (por ejemplo de modo privado a modo shared) deben adaptarse las propiedades y permisos de ficheros y directorios.

- ¿Estás obteniendo el error `gethostbyname: <hostname>: Non-authoritative 'host not found', o 'server failed'`?

  Para resolver esto tienes que chequear las siguientes cosas:

  1. ¿Está configurado correctamente el fichero `/etc/hosts`? `<hostname>` no es el nombre de tu host (las versiones viejas de SLS y algunas viejas y nuevas versiones Slackware se entregan con el `hostname` *darkstar*, por ejemplo). Cambia esto en el fichero. Debe contener al menos una línea como la que sigue (el formato se describe encima de ella):

```
# Formato del fichero:
#  IP_NUMBER           HOSTNAME        ALIASES
#
# Este es el nombre de tu maquina, en primer lugar, seguido de aliases
#
127.0.0.1       localhost       linuxpc.dominio  linuxpc
```

Cuando tu única conexión con el exterior se hace mediante `term`, la línea anterior es la única que debe aparecer en `/etc/hosts`. No pongas otros nodos de Internet en ese fichero, pues `term` no trabajará con ellos.

2. ¿Puede leer todo el mundo tus ficheros `/etc/rc*` y `/etc/resolv.conf`?

   ```
   chmod ugo+r
   ```

3. Por último, asegúrate de haber instalado el *loopback-interface* de *TCP/IP* en tu máquina. Puedes comprobarlo ejecutando el comando `ifconfig`. Cuando está instalado dicho interface, se verá lo siguiente en pantalla:

   ```
   lo        Link encap:Local Loopback
             inet addr:127.0.0.1  Bcast:127.255.255.255  Mask:255.255.255.0
             UP BROADCAST LOOPBACK RUNNING  MTU:2000  Metric:1
             RX packets:0 errors:0 dropped:0 overruns:0
             TX packets:4984 errors:0 dropped:0 overruns:0
   ```

   Durante el arranque también puedes ver qué protocolos se utilizan. En mi máquina aparece lo siguiente:

   ```
   IP Protocols: ICMP, UDP, TCP
   ```

   Para más información sobre cómo instalar el *loopback-interface*, léete el *NET-HOWTO*.

- ¿Estás obteniendo todo tipo de mensajes `timed out` en tus ficheros log de `term`? Esto significa que tu conexión de `term` no está optimizada. Un pequeño número de esos mensajes nunca es problema. Estos son debidos muy posiblemente a que los factores temporales influyen en la conexión física entre tus *hosts* remoto y local.

  Cuando tienes un montón de estos mensajes todo el tiempo, tu conexión se ralentizará considerablemente. Tienes que ajustar los parámetros mencionados en la anterior sección 4.13. Además, esta parte de la instalación es un proceso de prueba y error. No se pueden dar unas reglas fijas sobre los valores a colocar en los variados parámetros ya que son muchos los factores que influyen en la conexión. Estos factores difieren entre las conexiones e incluso en el tiempo.

- ¿No te funcionan los `ftp` con puertos redirigidos? Es un problema habitual con `ftp`, que necesita trabajar con los puertos 20 y 21. La única solución es usar una versión TERMificada de `ftp` o `ncftp`. Sin embargo, algunas de estas aplicaciones adaptadas tampoco funcionarán.

5 Clientes TERM

`Term` proporciona varios clientes por defecto. Esto incluye `trsh`, `tmon`, `tupload`, `tredir`, `txconn` y en nuevas versiones `trdate`, `trdated`. Además, a partir de la versión 2.0.0 está disponible `tudpredir` y desde la versión 2.1.0 también `tdownload`. Esta sección hará referencia a `trsh, tmon, tupload, tdownload, trdate` y `trdated`. El resto tiene su propia sección cada uno. No funcionará ningún cliente de `term` hasta que se haya establecido un enlace `term`.

`tmon` es una utilidad simple para monitorizar las estadísticas del enlace. Imprime un diagrama de tiempo de caracteres transmitidos y recibidos. Se invoca simplemente como `tmon`. Desde la versión 1.11, `tmon` ha tenido un bug que provoca que alguna información se trunque (??).

`trsh` es similar a `rsh`. Sin argumentos, genera una shell interactiva en el sistema remoto (esto es, te introduce directamente en el sistema remoto). `trsh` es uno de las principales maneras de acceder al extremo remoto del enlace a través de `term`. Si se le pasa un argumento, `trsh` ejecuta ese argumento como un comando en el sistema remoto. Por ejemplo, con `trsh ls` se obtendría una lista de ficheros del directorio *home* del sistema remoto.

`tupload` transfiere un fichero, si se indica como primer argumento, desde el sistema local al remoto. Por defecto los ficheros se pondrán en el mismo directorio desde el que se invocó `term` en el otro extremo. Para colocar los ficheros en otro directorio se deben poner sus nombres como segundo argumento a `tupload`. Por ejemplo, si se desea poner una copia del fichero `term114.tar.gz` en `/usr/tmp` en el sistema remoto, se escribiría `tupload term114.tar.gz /usr/tmp`.

Cuando se usa `tupload` es posible utilizar comodines como en `tupload a.*`. La shell expande los comodines y llama a `tupload` como `tupload a.1 a.2 ......`

`tdownload` transferirá un fichero, si se indica como primer argumento, desde el sistema remoto al local. Por defecto, los ficheros se colocarán en el mismo directorio desde el que se invocó al `term` en el lado local. Para colocar los ficheros en otro directorio, hay que indicar sus nombres como segundo argumento a `tdownload`. Por ejemplo si se quiere poner una copia del fichero `term114.tar.gz` en `/usr/tmp` del sistema local, escribiría `tdownload term114.tar.gz /usr/tmp`.

Cuando se usa `tdownload` no es posible utilizar comodines como `tdownload a.*.` La razón es que el directorio remoto no está disponible para la shell local cuando se usa `tdownload`; de modo que la shell local no puede expandir los comodines.

`trdate` es una utilidad de puesta en hora. Lee la hora en la máquina remota y pone el reloj local con la hora del remoto. Se debe ejecutar como `root`.

`trdated` es la versión demonio de `trdate`. Cuando se pone en marcha en `rc.local` se ejecuta como un demonio en cuyo caso ajusta el tiempo cada 5 minutos (por defecto). Incluso si no existe conexión `term`, este demonio se pondrá en marcha si se coloca en `rc.local`. Una vez que se cree una conexión `term`, comenzará a ajustar la hora.

6 X y TERM

`term` permite a los usuarios abrir ventanas X en la máquina local desde clientes que están corriendo en una máquina de la red. Esto se hace usando el cliente `txconn`. `txconn` se ejecuta en el remoto, la máquina conectada en red; se invoca simplemente como `txconn`. Se coloca en background y devuelve un número a la salida estándard; este número es el número de *display* que los clientes deben usar para acceder al servidor X de la máquina local. Un ejemplo aclarará ésto. Estoy accediendo a mi cuenta, vía `trsh`, en mi host remoto, llamado "`foo`". En `foo` realizo lo siguiente:

```
foo$ txconn
Xconn bound to screen 10
:10
foo$
```

Ahora, para cualquier host en el que quiera correr un cliente X, o sea, verlo en el servidor X de mi máquina local, haré:

```
setenv DISPLAY foo:10
```

(para `bash` debería usarse `export DISPLAY=foo:10`). En algunos casos además de eso puede ser necesario hacer un `xhost + foo` en la máquina local. Ahora, cuando arranque el cliente, intentará conectar con la pantalla 10 de la máquina `foo`, pero `txconn` estará escuchando en esa pantalla, y pasará todos los paquetes con protocolo X vía `term` al servidor X en el host local; es decir, la ventana se abrirá en la máquina local.

Es posible trabajar en el otro sentido - correr un cliente en la máquina local y tener abierta la ventana en una máquina remota de la red; sin embargo, retrasaremos esta explicación hasta después de haber comentado `tredir`.

`txconn` no es terriblemente seguro; cualquiera puede conectar con el servidor local vía `term` y provocar todo tipo de daños. Si este tipo de cosas son preocupantes, podría ser una buena idea considerar el uso de `xauth` para autorizar las conexiones. Consulte la página de manual sobre `xauth`.

El protocolo X no es muy eficiente; desperdicia algo de ancho de banda. Esto no suele ser un problema en una ethernet, pero puede ser un crimen vía módem. Se supone que X11R6 presenta una versión de bajo ancho de banda del protocolo X, *LBX*. Si, por el contrario, se está usando X11R5 puede recurrirse a una utilidad llamada `sxpc` que comprime el protocolo X, mejorando la respuesta sobre líneas serie. `sxpc` incluye un texto sobre cómo hacerlo funcionar con `term`, y está recomendado. El paquete `sxpc` también explica como usar `xauth`; por lo que está doblemente recomendado.

7 tredir

`tredir` es una de las utilidades más potentes de `term`, permitiendo que la mayoría de los servicios de red importantes puedan obtenerse en un enlace `term`. Antes de explicar cómo se usa `tredir`, es necesario dar algunas nociones sobre los servicios de red.

Ya se ha hablado antes sobre los servicios de red, pero no se ha dicho exactamente qué son. Los servicios son justo eso - servicios que proporciona la red. Ejemplos de servicios incluyen `telnet`, que proporciona logins entre máquinas, el `ftp` (*File Transfer Protocol*), o Protocolo de Transferencia de Ficheros, que transfiere ficheros entre máquinas, y *smtp*, el protocolo de transmisión de correo, que se usa siempre que se envía un correo electrónico.

Cada servicio de red tiene un número de puerto asociado a él. El mapeo de números de puerto con los servicios correspondientes se da en el fichero `/etc/services`. Este fichero debería ser el mismo en todas las máquinas conectadas a Internet.

¿Como se accede a estos servicios? Cada máquina en red corre un demonio llamado `inetd`, el cual escucha los intentos de conexión a los puertos de red. Estas peticiones pueden llegar tanto desde la red, como desde la propia máquina. Un servicio de red se obtiene conectando con un puerto `inetd` en particular. Cuando se hace una solicitud de red, `inetd` conoce exactamente qué servicio está implicado, por el número de puerto al que se hizo la solicitud. Si se configura `inetd` para hacerlo, proporcionará el servicio adecuado a la conexión que lo solicita. La configuración de `inetd` es la que se da en el fichero `/etc/inetd.conf`, que contiene una lista de los servicios que proporciona `inetd`. Para más información vea las páginas de manual de `inetd` e `inetd.conf`.

Se puede comunicar directamente con los servicios de red usando `telnet` (nótese bien, no `termtelnet`). Por ejemplo, para hablar con el demonio de `sendmail` (o *smtp*) en la máquina `nombre_de_máquina;`, se puede hacer un `telnet nombre_de_máquina smtp` , o `telnet nombre_de_máquina 25`, (ya que 25 es el puerto asignado a `smtp` en `/etc/services`). Debería obtener una agradable bienvenida del demonio de la máquina remota. Este es un truco muy útil para depurar problemas de red y chequear puertos redirigidos con `tredir` (ver abajo).

`tredir` funciona de modo similar a `inetd`. Funciona en *background* como un demonio, escuchando los puertos de red, esperando a una petición. Cuando se hace una solicitud de un servicio, en vez de proporcionar ese servicio, como hace `inetd`, `tredir` traslada la solicitud a través del enlace `term` hasta el `term` remoto, quien hace la solicitud a la red, devolviendo el resultado de nuevo por el enlace hasta el cliente local. `tredir` puede trasladar la solicitud a cualquier máquina de la red, pero por defecto la envía a la máquina al otro extremo del enlace `term`. `tredir` "redirige" los servicios `TCP` (*Transmision Control Protocol*) a través del enlace `term`.

Un ejemplo lo aclarará. Vamos a redirigir un puerto local al puerto `telnet` de la máquina remota. Para hacer esto pondríamos `tredir 2023 23`. Ahora, cualquiera que conecte al puerto 2023 de la máquina local será redirigido al puerto 23 (`telnet`) de la máquina remota. Aquí va una sesión de ejemplo; la máquina local es `mimaquina.modem.casa` y la remota es `netsun`.

```
$ tredir 2023 23
Redirecting 2023 to 23
$ telnet localhost 2023
Trying 127.0.0.1...
Connected to mimaquina.modem.casa
Escape character is '^]'.


SunOS UNIX (netsun)
login:
```

Este ejemplo es realmente muy útil. Si en su lugar hiciera el `tredir` sobre `netsun`, entonces podría hacer `telnet` a `mimaquina` desde la red simplemente conectándome al puerto redirigido de la máquina en red (usando `telnet`) - esto es, `telnet netsun 2023`.

El principio general de uso del `tredir` es redirigir el servicio deseado a una máquina de la red. El siguiente ejemplo nos permitirá leer las News en la máquina local a través del enlace `term` desde un servidor de News de la red. Las News las proporciona el servicio `nntp`, puerto 119. Todos los lectores de News decentes permiten especificar qué puerto van a utilizar, ya sea en un fichero de configuración o en una variable de entorno. Vamos a especificar que el puerto local sea el 2119. Ahora supongamos que el servidor de News es `news.domain.org`; entonces le diremos al software de lectura de News que el servidor `nntp` se encuentra en el puerto 2119 del *host* local. Como esto dependerá del lector de News que se use, probaremos el enlace con `telnet` en lugar de ejecutar un lector de News:

```
$ tredir 2119 news.domain.org:119
Redirecting 2119 to news.domain.org:119
$ telnet localhost 2119
Trying 127.0.0.1...
Connected to mimaquina.modem.casa.
Escape character is '^]'.
200 news.domain.org InterNetNews NNRP server INN 1.4 07-Dec-41 ready
(posting ok).
```

Si ha podido llegar tan lejos, todo lo que tiene que hacer es configurar su lector de News para poder leer las News desde casa vía `term`. (nótese bien, si lee las News de este modo, asegúrese de que en todos los mensajes que deje ponga una cabecera `Reply-To:` a una dirección de correo en la que pueda ser localizado, o de lo contrario la gente que quiera ponerse en contacto con Ud. mandará el correo a cualquier dato que su lector de News ponga en la cabecera `From:`).

7.1 ¡`tredir` puede morder!

El astuto lector, tras leer el último ejemplo se preguntará porqué se redirigió en puerto 2119 al puerto 119 –ya que el puerto por defecto de los lectores de News es el 119–, ¿porqué no podría hacer un `tredir 119 news.domain.org:119` y evitar la configuración del lector de News? La respuesta es que todos los puertos con números inferiores a 1024 son "puertos reservados", y únicamente el superusuario puede escucharlos. Si se desea tomar un riesgo en seguridad y hacer de `tredir` un programa *suid*, o ejecutar `tredir` como `root`, entonces se pueden redirigir puertos reservados y evitar así la molestia de renombrar servicios.

Otro problema de usar los puertos reservados es que `inetd` a menudo ya está escuchando en esos puertos, y solamente un programa puede escuchar un puerto a la vez. Si se quiere usar tal puerto, se debe cambiar `inetd.conf` de modo que `inetd` ya no escuche en ese puerto que se quiere redirigir. Esto se hace fácilmente comentando la línea correspondiente al servicio poniendo el carácter # al comienzo de la misma. El superusuario tiene que mandar una señal `HUP` a `inetd` (`kill -1 <inetd-pid>`) para hacer que vuelva a leer su configuración.

7.2 Trucos tontos de `tredir`

En esta sección describiremos algunos de los usos más comunes de `tredir`. Ya hemos descrito como redirigir los servicios `nntp` y `telnet`; Ahora daremos algunos ejemplos más complicados.

7.2.1 X window

En una sección previa, se describió como hacer que un cliente X que corre en la red abra una ventana en la máquina de casa usando `txconn`. La misma técnica se podría usar en la máquina de casa para mostrar un cliente en la máquina del lado remoto del enlace `term`. ¿Pero cómo puede uno ver un cliente X en una máquina de red que no es el extremo remoto? La respuesta se basa en conocer que X usa un servicio de red concreto igual que los otros programas que hemos explicado. Un servidor X escucha peticiones de red en un puerto cuyo número viene dado por la fórmula: *puerto = 6000 + número de display*, p.ej. un servidor X manejando la pantalla 0 en una máquina escucharía el puerto 6000, si estuviéramos manejando la pantalla 2, escucharía el puerto 6002. Si se pone la variable de entorno `DISPLAY` en maquinaX:n, los clientes X tratarán de conectar con el puerto $6000 + n$ de maquinaX.

Podemos usar esto para trucar los clientes X de la máquina local y abrir ventanas en displays remotos. Supongamos que quiero abrir un `xterm`, corriendo en mi máquina local, en el display 0 de la máquina maquinaX, que esta corriendo en algún lugar de la red. Primero escogeré un número de display local, digamos que el 2 (no se usa el 0, ya que es el que estará usando el servidor X local). Mapearé este display al display 0 de maquinaX. En término de puertos, esto significa que quiero redirigir el puerto local 6002 al puerto remoto 6000. Haré lo siguiente:

```
$ tredir 6002 xmachine:6000
$ setenv DISPLAY localhost:2
$ xterm
```

Esto debería abrir un `xterm` en la máquina maquinaX. Observe que he puesto el `DISPLAY` a `localhost:2`. Esto es porque los clientes X usan a veces *sockets* de dominio unix en lugar de *sockets* de dominio Internet, a su propio criterio, cuando conectan con un display local, si `DISPLAY` se pone a `:2`. `localhost:2` indica que use una conexión `TCP`.

Observe que en lo que concierne a maquinaX, la solicitud X viene de la máquina del extremo remoto del enlace `term (máquinaremota)` - de modo que si necesita autorizar la conexión, debería hacer bien `xhost + máquinaremota` en maquinaX, o bien usar `xauth` para actualizar el fichero `.Xauthority` en su máquina local para el display número 2, usando la clave de maquinaX.

De nuevo, para acelerar las conexiones X, se puede usar el programa `sxpc`, que incluye una explicación sobre cómo usar `tredir` para establecer el enlace y autorizarlo usando `xauth`.

7.2.2 Correo con `TERM`

Está bien, vosotros lo pedísteis. El correo electrónico tiene la justificada reputación de ser una de las cosas más dificiles de hacer funcionar bien en un sistema UNIX. Para conseguir que el `term` funcione correctamente con el correo es preciso entender cómo funciona el correo, lo cual va más allá del objetivo de este documento.

Para aprender más sobre correo, debería consultar un libro de administración de sistemas UNIX y/o la FAQ de la conferencia `comp.mail.misc`, disponible en el `ftp` anónimo de `ftp://rtfm.mit.edu/pub/usenet/comp.mail.misc`.

También tiene a su disposición 2 paquetes en el `ftp` anónimo de `sunsite.unc.edu` que le ayudarán a poner en marcha el correo bajo `term` - son `term.mailerd+smail` de Byron A. Jeff y `BCRMailHandlerXXX` de Bill C. Riemers.

Como se ha dicho, haremos una breve descripción de como funciona el correo electrónico. Hay dos partes que hacen funcionar el correo, el envío de mensajes y la recepción de los mismos. Comenzaremos con el envío de mensajes desde su ordenador local a la red.

Hay dos clases de programas de correo. El primero es el *Agente de Correo de Usuario* (*MUA - Mail User Agent*). Los *MUA*s ayudan a leer, componer y mandar mensajes. Ejemplos de *MUA*s son el `elm`, `pine`, `mail` y `vm`. Los *MUA*s no usan para nada la red; solamente agrupan los mensajes - el trabajo duro de envío de correo se hace a través de la segunda clase de programas, los agentes de transferencia de correo (*MTA - Mail Transfer Agent*). Estos son invocados desde los *MUA*s. Toman el mensaje, deciden dónde enviarlo observando la dirección, y finalmente lo envían a través de la red.

Los dos *MTA*s mas comunes en sistemas Linux son `sendmail` y `smail`. La idea básica es hacer que su *MTA* se conecte a otro *MTA* que esté corriendo en otra máquina de la red que sepa qué hacer con su mensaje. Esto se consigue redirigiendo un puerto local hacia el puerto `smtp` de la máquina en red. Entonces debe indicar a su *MTA* que tome todos los mensajes con los que no sepa que hacer, y los envíe fuera a través del puerto redirigido de su máquina local al *MTA* de la máquina remota, la cual encaminará los mensajes hacia su destino correcto.

¿Cómo hacemos esto usando `smail`? Primero redirigiremos un puerto al puerto `smtp` de la máquina de correo de la red (`mailhost`):

```
tredir XXXX mailhost:25
```

donde `XXXX` es el número de puerto al que se conecta `smail` en el host local (tenga en cuenta que hay que dar un nombre al puerto en `/etc/services` para hacer que `smail` lo reconozca). `smail` tiene varios ficheros de configuración que generalmente están en `/usr/local/lib/smail`. Los que nos interesan son `config, routers` y `transports`. Observar que presumimos que ya ha configurado `smail` correctamente para el correo local - envío a ficheros y tuberías y demás cosas. De nuevo, consulte la documentación si no lo ha hecho.

En el fichero `config`, ponemos la siguiente definición:

```
smart_path=localhost
```

`localhost` es la máquina a la que se conecta `smail` cuando no sabe que hacer con un mensaje.

En `routers` ponemos:

```
smart_host:
driver=smarthost,
transport=termsmtp;
path = localhost
```

En `transports` ponemos:

```
termsmtp:        driver=tcpsmtp,
        inet,
```

```
return_path,
remove_header="From",
append_header="From: SU_DIRECCION_DE_RED",
-received,
-max_addrs, -max_chars;
service=SU_SERVICIO_SMTP,
```

En el de arriba, las líneas `header` cambian la cabecera `From` en todo correo saliente por la dirección `SU_DIRECCION_DE_RED`, que será la dirección de red a la que quiere que le envíen el correo. Si su enlace `term` va a ser usado por más de una persona, tendrá que hacer algo más laborioso, como mantener una base de datos de direcciones de red de usuarios locales e insertar las mismas en las cabeceras `From:`.

La línea `service` es el nombre del número de puerto local que ha redirigido al puerto `smtp` de la máquina conectada a la red. En mi versión de `smail` no es posible ponerlo como un número, asi que tengo que ponerlo como un nombre, como "`foo`", y entonces definir "`foo`" en `/etc/services` de modo que sea el número del puerto redirigido. Si usa un *suid* de `tredir` y se redirige el puerto `smtp` (25), no es necesario definir esto.

Esto debería ser suficiente para hacerlo funcionar. Si decide usar `sendmail` la base es la misma pero difiere en los detalles. Ronald Florence (`ron@mlfarm.com`) me dijo que el `sendmail` de *Sun* no mandará mensajes múltiples encolados a través de un puerto redirigido; el `sendmail` 8.6.9 de *BSD* funciona bien. Él hizo los siguientes cambios al `sendmail.cf` para que funcionase con `term`. En este caso se usa el puerto por defecto de `sendmail` (25) para el tráfico sobre una ethernet local de forma que el correo Internet se pasa al puerto `TCP` redirigido.

```
#
# Crear el mailer termsmtp, el cual envia el correo via el puerto TCP
# redirigido
#
Mtermsmtp,P=[TCP], F=mDFMuCXe, S=22, R=22, A=TCP $h PORTNUMBER
```

Aquí, `PORTNUMBER` es el número del puerto redirigido en la máquina local. Este debería ser un puerto sin usar por encima del 2000. Seguidamente le decimos a `sendmail` a que máquina conectarse, y ponemos a `termsmtp` como *mailer* por defecto.

```
#
# relevo de correo principal
#
DMtermsmtp
#
# maquina del relevo principal: usa el mailer $M para enviar el
# correo de otros dominios
#
DR HOSTNAME
CR HOSTNAME
```

Aquí `HOSTNAME` es el nombre de tu host local (¿funcionará `localhost`?). La última entrada va debajo de `Rule 0` para pasar el correo Internet.

```
# Pass other valid names up the ladder to our forwarder
R$*<@$*.$+>$*           $#$M    $@$R $:$1<@$2.$3>$4     user@any.domain
```

Cuando la conexión `term` se haya establecido con el host Internet, ejecute los siguientes comandos en la máquina local.

```
tredir PORTNUMBER internet.host:25
/usr/lib/sendmail -q
```

Pasamos ahora a la recepción de correo electrónico usando `term`. Asumiremos que el correo se envía a su cuenta en el servidor de correo (`mailhost`) de la red. La solución más simple es usar `trsh` o `termtelnet` para acceder al servidor y leer su correo allí.

Sin embargo, también es posible hacer pasar el correo automáticamente a su máquina local. Una forma de hacer esto es usar el *Post Office Protocol*, (*POP*). *POP* fue diseñado precisamente para este propósito: enviar correo a máquinas que tienen conexiones de red esporádicas.

Para usar *POP* ha de tener instalado un servidor *POP* en *mailhost*. Suponiendo que lo tiene, puede usar entonces un cliente *POP* para recoger su correo cada poco tiempo. Esto se hace, como podría esperar, usando `tredir`. El servicio *POP* es el 110 (Observe que hay un protocolo más antiguo, *POP-2*, que usa el puerto 109; en este documento describiremos *POP-3*, que es la última versión de *POP*). Hay varios clientes *POP* disponibles. Uno, escrito en el lenguaje de scripts `perl`, es `pop-perl-1.X`, escrito por William Perry y mantenido por mí mismo - puede encontrarse en `sunsite` en `/pub/Linux/system/Mail`.

Para usar *POP* se redirige un puerto local al puerto 110 de `mailhost` y se configura el cliente para recoger su correo de `localhost` usando el puerto local. Como ejemplo, supongamos que hay un servidor *POP* corriendo en `mailhost`. Redirigiremos en puerto local 2110, y ejecutamos el cliente pop-perl:

```
$ tredir 2110 mailhost:110
Redirecting 2110 to mailhost:110
$ pop
Username: bill
Password: <introduzca su password para mailhost>
Pop Host: name of local
Pop Port: 2110
Starting popmail daemon for bill
```

Si no tiene un servidor *POP* disponible, el paquete `BCRMailHandler` tiene un programa para capturar su correo desde un enlace `term` hasta su máquina local. No lo he usado, pero cualquier comentario de alguien que lo haya hecho será bienvenido. También puede usar el paquete `term.mailerd+smail` para este propósito. Sin embargo, `BCRMailHandler` y `term.mailerd+smail` ya no funcionan con versiones de `term` 2.0.0 o superiores.

8 tudpredir

`tudpredir` es similar a `tredir` si se observa lo que estos programas hacen y cómo se ejecutan. La gran diferencia entre los dos es que `tredir` se usa para redirigir servicios de red `TCP`, mientras `tudpredir` redirige servicios de red `UDP` (*User Datagram Protocol*) a través de un enlace `term`. Una diferencia más importante entre los dos programas es que `tredir` se convierte en un demonio en segundo plano una vez que se ha establecido el puerto local, mientras los comandos `tudpredir` hay que ponerlos en segundo plano manualmente.

El formato de una llamada a `tupredir` es:

```
tudpredir [esta_maquina:]puerto [la_otra_maquina:]puerto
```

9 Automatizando las cosas.

Ahora que ya sabe cómo conseguir todos los servicios de red sobre `term`, sería bonito colocar las cosas de tal modo que su enlace se establezca y configure automáticamente. Básicamente hay infinitas formas de hacerlo, dependiendo de que programa de comunicación utilice y cómo acceda a su sistema remoto.

Un programa que yo no he usado, pero que he oído que es bastante agradable, es **fet**: un *front-end* para **term**. Está diseñado para introducirle en un sistema remoto y poner en marcha **term** y todos sus **tredirs**. Cualquier comentario sobre **fet** será bienvenido.

Le daré un ejemplo de una serie de comandos que usa **kermit** para introducirse en el sistema remoto y que efectúa todas las inicializaciones de **term**. Obviamente, si usa estos ejemplos, tendrá que modificarlos para sus propios procedimientos de acceso.

El comando que se invoca en este caso es la *shell script* **knet**, dada por:

```
#!/bin/sh
/usr/bin/kermit -y $HOME/.kerm_term > $HOME/klog < /dev/null 2>& 1
exec $HOME/bin/tstart >> $HOME/klog 2>& 1
```

La script **.kerm_term** viene dada por:

```
pause 2
# El numero al que llamar
output atdtXXXXXXX \13
# Acceso al servidor de terminal
input 145 {name: }
output MYNAME \13
input 3 {word: }
output MYPASSWORD \13
input 5 {xyplex>}
# Hacer la linea transparente
output term telnet-t \13
output term stopb 1 \13
# Conectar al host remoto
output telnet remotehost.somedomain.org \13
input 10 {ogin: }
output MYOTHERNAME \13
input 3 word:
output MYOTHERPASSWORD \13
pause 5
# Lanzar term en el host remoto
output exec term -s 38400 -l $HOME/tlog -w 10 -t 150  \13
! /usr/bin/term -r -l $HOME/tlog -s 38400 -c off -w 10 -t 150 < /dev/modem  > /dev/modem &
# Abrir otros clientes aqui
suspend
!killall -KILL term
```

y finalmente, el *script* **tstart** que lanza los clientes **term** es:

```
#!/bin/sh
#
# Esto hace que salga el correo, pueda leer news, y pueda recoger correo.
#
/usr/local/bin/tredir 2025 25 2119 newshost:119 2110 pophost:110
#
# Puedo abrir una Xwindow aqui
#
/usr/local/bin/trsh -s txconn
```

```
#
# Ahora recibire el correo....
#
/usr/local/bin/pop
#
# Limpiar la cola, en caso de boo-boos
#
/usr/bin/runq
#
# Acabado
#
echo ^G^G > /dev/console
```

Cuando por fin quiera cerrar la conexión, retoma y termina el `kermit`. La última línea del *script* mata el `term` local y al sistema a su estado inicial.[1]

Como ya dije, hay *zillones* de formas de hacerlo; estas sólo se han citado como ejemplos para que pueda comenzar. Otros ejemplos pueden encontrarse en los paquetes `autoterm` y `JoelTermStuff`.

10 Portando software para usarlo con `term`.

En principio, todos los programas que se puedan usar sobre una red pueden usarse en combinación con `term`. Algunos de ellos podrá encontrarlos ya como binarios con soporte para `term`. Esto incluye `telnet, (nc)ftp, mosaic` y muchos otros. La mayoría de estos programas se han compilado para `term` 1.17 o anteriores. A pesar de eso, aún deberían funcionar con versiones más nuevas de `term`.

Otra forma de hacer que los programas funcionen con `term` es portarlos usted mismo. Este proceso es el que se describe en la siguiente subsección.

El último modo de compatibilizar sus programas con `term` es *termificándolos*.

10.1 Portar y compilar los fuentes.

Portar software a `term` se puede hacer usando un procedimiento bastante sencillo:

Si está instalado en `/usr/local` por `root`:

1. Añada a los flags de compilación `-include /usr/local/include/termnet.h`
2. y añada a la lista de librerías `-ltermnet`

Si está instalado en su directorio *home*:

1. Añada a los parámetros de compilación `-include $HOME/term/termnet.h`
2. y añada a la lista de librerías `-L$HOME/term -ltermnet`

Ahora compile el software como se describe en el documento `INSTALL` o `README` que venga con el software. ¡Ya estaría todo!

En este momento los comandos deberían funcionar con y sin `term`.

```
telnet localhost
```

[1] Nota del autor: en lugar de hacer `!killall -KILL term`, creo que sería posible hacer solamente `!tshutdown`. ¿Funcionará esto también?

no usa `term` para conectar, pero

```
telnet bohr.physics.purdue.edu
```

usará `term` sólo si no hay otro tipo de conexión de red.

Algunos comandos, como `rlogin`, sólo pueden ser usados por `root` y por el propietario de la conexión `term` (personas privilegiadas).

Algunos comandos `term` serán transparentes a `term` y sólo usarán `term` cuando no haya otra opción. Algunos ejemplos típicos son `telnet` y `ftp`.

Otros requieren un parámetro externo para indicarles que les es posible usar `term`. En estos programas se incluyen `xarchie, fsp` e `ytalk`.

Se puede poner el parámetro a estos programas para que usen `term`, bien colocando la variable de entorno `TERMMODE` como se especifica en `README.security`, o bien, ejecutando `make installnet`. Eventualmente, el fichero `termnet` creado contendrá instrucciones de red específicas, pero por ahora sólo está probada su existencia.

Si se añade una *conexión ethernet*, puede simplemente quitar el fichero `termnet` y ¡continuar usando los mismos binarios!

NOTA: Aquellos programas que fueron portados en los tiempos del `client.a`, aún pueden ser recompilados para usarlos con versiones nuevas de `term` cambiando simplemente la referencia a `client.a` por `libtermnet.a`.

10.2 *Termificar* (`termify`).

Este paquete convertirá los binarios enlazados dinámicamente para usar `term`.

Antes de poder *termificar* deberá asegurarse de que tiene una versión 2.2i (en esta versión 2.2.8?) de `term` o posterior y `libc.so.4.5.26` o posterior. Entonces hay que crear el fichero `libt.so.4` en el directorio `/lib` (ver el fichero `README` del paquete).

El problema en este momento es que hay que rehacer el fichero `libt.so.4` cada vez que renueve la versión de `term`.

Después de crear la librería podrá dejar que `termify` *"digiera"* al programa que se quiere hacer *term-compatible*, usando el comando:

```
termify <programa>
```

Si no le gusta el resultado puede *des-termificar* el programa que acaba de termificar, usando el comando:

```
termify -u <programa>
```

Por fin, el paquete también contiene un *script* para *termificar* completamente `smail`; de modo que no son necesarias definiciones especiales de transporte. La única cosa que quizá quiera cambiar es la dirección del `From:`.

11 Clientes `term`.

11.1 Clientes `term` disponibles en los servidores `ftp`.

A continuación se da una lista de aplicaciones que corren con `term`. Yo no digo que esta lista esté completa; así que cualquier añadido será bienvenido. Siempre que sea posible indicaré el *site* y directorio donde se pueda encontrar la aplicación. Si indico `sunsite.unc.edu`[2] como el lugar donde encontrar la aplicación, quiero decir que puede encontrarla en uno de los dos directorios siguientes:

1. `ftp://sunsite.unc.edu/pub/Linux/apps/comm/term/apps`
2. `ftp://sunsite.unc.edu/pub/Linux/apps/comm/term/extra`

¡Allá vamos! :-)

paquete TERM:

```
tupload
tdownload           (versiones 2.1.0 y posteriores)
trsh
tmon
tredir
tudpredir           (versiones 2.0.0 y posteriores)
txconn
trdate(d)
tshutdown
libtermnet
```

Transferencia de ficheros:

```
ftpd                sunsite.unc.edu
termncftp           sunsite.unc.edu
ncftp185            sunsite.unc.edu:/pub/Linux/system/Network/file-transfer
fsp                 sunsite.unc.edu:/pub/Linux/system/Network/file-transfer
```

Sistemas de Información:

```
lynx
Mosaic              sunsite.unc.edu:/pub/Linux/system/Network/info-systems/Mosaic
chimera
netscape            sunsite.unc.edu:/pub/Linux/system/Network/info-systems
httpd
xgopher
gopher              sunsite.unc.edu
```

Acceso remoto:

```
termtelnet          sunsite.unc.edu
rlogin              physics.purdue.edu:/pub/bcr/term/extra
rsh                 physics.purdue.edu:/pub/bcr/term/extra
```

Noticias (news):

```
tin 1.3             sunsite.unc.edu:/pub/Linux/system/Mail/news
news2               sunsite.unc.edu
```

Correo:

[2]En España contamos con un magnífico y veloz *mirror* de SunSite, localizable en `sunsite.rediris.es`

slurp	sunsite.unc.edu
smail	sunsite.unc.edu
term.mailerd+smail	sunsite.unc.edu
BCRMailHandlerXXX	physics.purdue.edu:/pub/bcr/term

Scripts automatizadores:

JoelTermStuff	sunsite.unc.edu
autoterm	sunsite.unc.edu
fet	sunsite.unc.edu

Otros programas:

inetd	sunsite.unc.edu
rdate	sunsite.unc.edu
xgospel	sunsite.unc.edu:/pub/Linux/games/x11/networked
termify	physics.purdue.edu:/pub/bcr/term/extra
xboard	sunsite.unc.edu
ircII	sunsite.unc.edu:/pub/Linux/system/Network/chat
whois	
xwebster	sunsite.unc.edu
sxpc	ftp.x.org:/R5contrib
xztalk	sunsite.unc.edu:/pub/Linux/apps/sound/talk

11.2 El paquete `termnet`.

El paquete `termnet-2.0.4-Linux-bin.tar.gz` (`ftp://sunsite.unc.edu/pub/Linux/apps/comm/term`) contiene un par de clientes precompilados, un par de scripts, páginas de manual y `libtermnet.so.2.00.04`. Los clientes se han compilado usando esta versión de `libtermnet.so`. El paquete contiene los siguientes clientes:

fet	perl	sperl4.036	tmon	tshutdown	xgopher
finger	perl4.036	suidperl	trdate	tudpredir	ytalk
ftp	rcp	taintperl	trdated	tupload	
fwhois	rlogin	telnet	tredir	txconn	
ncftp	rsh	term	trsh	xarchie	

AVISO: El paquete también contiene el conjunto completo de clientes compilados de `term` 2.0.4 incluyendo el mismo `term`. No instale este paquete hasta que esté seguro de lo que quiere. Destruirá otras versiones de `term` y sus clientes si empieza a enredar con los ejecutables.

11.3 Solicitado, pero aún no soportado:

1. `DOOM`: El problema con este juego parece ser el hecho de que usa el puerto 5029 ya sea como cliente o como servidor.

2. `NFS`: El servidor `NFS` se supone que sólo acepta llamadas si el *socket* que solicita la conexión está ligado a un puerto por debajo del 1024. Esto parece ser problemático. Sin embargo, algunos servidores `NFS` tienen una opción *insegura*. En este caso `NFS` podría funcionar ocasionalmente, si se le añade al `term` soporte `RPC`.

12 Term y la seguridad

En esta sección puntualizaré algunos aspectos sobre la seguridad usando TERM. Los problemas serán expuestos junto a un mecanismo para aumentar su seguridad.

12.1 trsh.

trsh es inseguro si se usa para acceder al Linux local desde el sistema remoto. El problema con TERM y sus clientes es que en el otro extremo de la comunicación el superusuario puede ejecutar programas de TERM.

Esto también indica que el "root" del otro sistema puede ejecutar trsh y entrar con los privilegios del propietario de la conexión fácilmente. Si este propietario es "root" la *habremos liado*.

La solución es simple: poner la siguiente línea en el fichero termrc de la máquina local:

```
denyrsh on
```

Con esto, nadie podrá usar trsh desde el sistema remoto para entrar en el local. Cuando tú mismo quieras entrar, podrás hacerlo aún usando telnet y puertos redirigidos.

12.2 txconn y xauth

txconn no es terriblemente seguro; cualquiera puede conectar al servidor local con term y hacer de todo. Si te preocupa, puedes usar xauth para establecer las autorizaciones de acceso. Mira el ejemplo de la siguiente sección.

12.3 sxpc, xhost y xauth

sxpc en combinación con xhost + es muy peligroso si no usas xauth.

Usar xauth es muy importante para mantener la seguridad cuando se usa sxpc. Si no usas xauth al usar sxpc, será muy peligroso tener xhost +. Algunos peligros son:

- Alguien puede saber lo que hay en tu pantalla
- Alguien puede saber lo que tecleas
- Alguien puede teclear sobre alguna de tus ventanas (por ejemplo, un comando que borre tus ficheros :-( )

xauth forma parte de las versiones R4 y posteriores de X. Aquí describiremos cómo usar básicamente el xauth. Esta configuración es vulnerable al husmeo de la red, pero puede convivir con ella fácilmente.

NOTA: cuando uses xauth asegúrate que la variable $DISPLAY no tiene el valor localhost (o localhost:loquesea). Si tu variable $DISPLAY vale localhost, los clientes no podrán encontrar la información de autorización. Lo mejor es usar el nombre real de la máquina. Si sigues las instrucciones de compilación del README, y compilas sin la variable -DNOGETHOSTNAME puede que todo funcione.

Sea C la máquina que ejecuta clientes, y D la máquina que pone la pantalla.

Primero, elige una "clave", de hasta 16 pares de dígitos hexadecimales (números del rango 0-9 y a-f). Necesitarás proporcionar esta clave en el lugar de <clave> en este ejemplo:

En C:

```
% xauth
xauth:  creating new authority file $HOME/.Xauthority
Using authority file $HOME/.Xauthority
xauth> add Nombre_de_C:8 MIT-MAGIC-COOKIE-1 <clave>
xauth> exit
```

En *D*:

```
% xauth
xauth:  creating new authority file $HOME/.Xauthority
Using authority file $HOME/.Xauthority
xauth> add Nombre_de_D/unix:0 MIT-MAGIC-COOKIE-1 <clave>
xauth> add Nombre_de_D:0 MIT-MAGIC-COOKIE-1 <clave>
xauth> exit
```

Cuando inicies el servidor X en *D* deberías poner el parámetro `-auth $HOME/.Xauthority`. Puede que necesites crear o editar el fichero `$HOME/.xserverrc` para controlar el inicio del servidor X. Por ejemplo:

```
#!/bin/sh
exec X  -auth $HOME/.Xauthority $*
```

Asegúrate que el fichero `.Xauthority` es legible sólo por `C` y por *D*.

13 Cosas a recordar

En esta sección intento obsequiarle con una lista de direcciones `ftp` de utilidad, *URL*'s, etc. donde puede encontrar software e información sobre `term`.

Ftp:

- `ftp://sunsite.unc.edu:/pub/Linux/apps/comm/term/`
- `ftp://sunsite.unc.edu:/pub/Linux/docs/HOWTO/`
- `ftp://physics.purdue.edu:/pub/bcr/term/`

URL:

- `http://sunsite.unc.edu/mdw/HOWTO/Term-HOWTO.html`
- `http://www.bart.nl/~patrickr/term-howto/Term-HOWTO.html` (siempre tiene la última versión)
- `http://physics.purdue.edu/~bcr/homepage.html`

Netnews:

```
comp.os.linux.announce          aqui se anuncian nuevas versiones de term
comp.os.linux.help              aqui puede poner sus dudas sobre term
comp.os.linux.misc              o aqui
comp.protocols.misc             las respuestas a las preguntas son enviadas aqui.
```

Cuando comience a hacer preguntas en las news, por favor, asegúrese de que da a la gente del grupo tanta información como necesitan para resolver su problema (versión de `term`, de qué modo establece su conexión, etc.).

En este momento hay en uso muchas versiones de `term` y todas ellas tienen sus problemas comunes y específicos. Además, cuando quiera una respuesta útil, al menos indique qué versión de `term` está usando. De lo contrario, en algunos casos sólo por adivinación será posible ayudarle a resolver sus problemas.

Documentos relacionados:

- *Using Term to Pierce an Internet Firewall HOWTO* de Barak Pearlmutter, `bap@learning.scr.siemens.com`
- *Cortafuegos-Como*[3] de David Rudder, `drig@execpc.com`
- *Serial HOWTO*, de Greg Hankins, `gregh@cc.gatech.edu`
- *Net-2/Net-3 HOWTO* de Terry Dawson, `terryd@extro.ucc.su.oz.au`

14 Estabilidad de las versiones de term

Hay muchas versiones de `term` rondando en estos momentos. El que mantiene el `term`, Bill Riemers, ha hecho una lista de versiones de `term` indicando qué versiones son estables y qué versiones es mejor evitar. La lista es la siguiente:

```
term110         --> no puedo decir con seguridad
term111         --> no puedo decir con seguridad
term112         --> no puedo decir con seguridad
term113         --> no puedo decir con seguridad
term114         --> version BETA bastante estable
term115         --> version BETA inestable BETA
term116         --> version BETA inestable BETA
term117         --> version BETA inestable BETA
term118         --> version BETA semiestable
term119         --> version GAMMA estable
term-2.0.X      --> BETA versiones BETA semiestables
term-2.1.X      --> mas versiones BETA estables
term-2.2.X      --> nuevas versiones BETA
term-2.3.X      -->
```

15 Tabla de velocidad de term.

Gracias a Bill McCarthy ahora disponemos de una tabla con información de la velocidad de `term` para diferentes módems, versiones de `term` y condiciones de conexión. El propósito es dar, tanto a los usuarios nuevos como a los experimentados, una idea de lo que otra gente está usando y los resultados que consiguen.

```
                        LINUX TERM CHART 8/14/94


   |___velocidad/marca____|___vel. linea__|__cps med__|__max.__|__ver term_|
   | 1)  USR SP 14.4      |   9600        |  950      |  963   | 1.17      |
   | 2)  USR SP 14.4      |  14400        | 1376      |  n/a   | 1.18p06   |
   | 3)  Zoom 2400        |   2400        |  220      |  230   | 1.19      |
   | 4)  Boca V.32bis 14  |  57600        | 1400      |  n/a   | 1.01/09?  |
   | 5)  Viva 14.4        |  14400        | 1300      |  n/a   | 1.16      |
   | 6)  USR SP 14.4      |  14400+       | 1550      | 1680   | 1.19      |
```

[3]Disponible en castellano, consulte la sección 19

```
| 7)  Intel 14.4 Fax   |  14400        | 1400      | 1650   | 2.0.4     |
| 8)  cable tv hookup  |  57600        | 1500      | 1800   | 1.18p06   |
| 9)  Twincom 144/DFi  |  57600        | 1500      | 4000?  | 2.0.4     |
| 10) USR SP 14.4      |  14400        | 1200      | 1500   | 1.08      |
| 11) cable tv hookup  |  19200        | 1300      | 1800   | 1.19      |
|-----------------------------------------------------------------------|

+parametros en el termrc:

1) default escapes   2) window 5       3) baudrate 2400    4) n/a
 baudrate 9600        timeout 200       window 3
 window 10                              noise on
 timeout 150

5) compress off      6) baudrate 19200  7) ignore 19+17    8) compress off
 window 10            compress on        window 4            escape 0, 13,
 timeout 150                             timeout 90          16-19, 255
 baudrate 38400                                              baudrate 0
                                                             shift 224
                                                             flowcrtl 500
                                                             window 10
                                                             timeout 70
                                                             retrain on
                                                             breakout 24
9) compress off     10) compress off      11) baudrate 19200
 baudrate 57600       baudrate 38400        compress on
 window 10            escape 17, 19         shift 224
 timeout 200          remote                escape 0, 13 16-17
 noise on                                   19, 255
 share on                                   window 10
 remote                                     timeout 40

Escapar caracteres en un extremo implica ignorarlos en el otro extremo.
```

16 Pistas y trucos encontrados en la red

En los grupos de news relacionados con Linux surgen de nuevo muchas preguntas sobre `term` cada 15 días, junto con las respuestas a estas preguntas. Para reducir el tráfico del grupo de news, intentaré hacer un resumen de esas preguntas y las respuestas a las mismas. Algunas de las respuestas han sido probadas por mí, ya que también tuve los citados problemas. Otras simplemente las he tomado sin comprobarlas.

- Mucha gente, en especial los que usan *Ultrix*, parecen tener problemas con el `vi` porque les muestra menos de 24 líneas en ventanas con 24 líneas. Hay dos formas de resolver este problema:

  1. Acceda al sistema remoto usando:

     ```
     trsh -s telnet <hostname>
     ```

  2. Ponga `resize; clear` en su fichero `.login`
  3. La mejor solución parece ser poner lo siguiente en el remoto:

     ```
     stty 38400
     ```

- Mucha gente parece tener problemas con caídas de la conexión `term`, cualesquiera que sean los motivos de la caída. Así que antes de poner en marcha aplicaciones, la gente quiere saber si su conexión `term` sigue viva o no. Esto se puede probar usando estos pequeños ejemplos de *shell scripts*:

  Si está usando `tcsh`:

```
if ( { trsh -s true } ) then
  ...
endif
```

Si está usando **bash**:

```
if trsh -s true; then
  ...
fi
```

- El navegador de *WWW* `netscape` causa a la gente problemas para funcionar con `term`. La buena noticia es que puede funcionar. He aquí cómo:

  1. Termificar el netscape
  2. Lanzar `termnetscape`. En el menú `Options | Preferences | Mail/Proxys` dejar todos los campos `proxy` en blanco; poner en el campo `SOCKS`: `remotehost` y 80
  3. Ignorar el error que saldrá en el menú de opciones.
  4. Si no funciona bien: en el menú `Options | Preferences | Mail/Proxys` dejar todos los campos `proxy` en blanco, poner en el campo `SOCKS`: `none` y 80
  5. Ignorar el error que saldrá en el menú de opciones.

  Scott Blachowicz me envió un mensaje diciendo que hay una forma fácil de hacer que Netscape u otro navegador funcione bien con `term` si tiene instalado un servidor proxy (como el `httpd` del CERN) para ser usado en el lado *remoto* del enlace `term`. En el lado local tendrás que hacer esto entonces:

  - `tredir local:8080 remota:80`

    Donde `remota` es el nombre remoto del servidor *proxy*.
  - Ejecuta el navegador, y en las opciones o como se llame, pon los proxies como `local`, puerto 8080. En algunos navegadores esto se hace mediante variables de entorno como:

    ```
    export http_proxy=http://localhost:8080/
    export ftp_proxy=http://localhost:8080/
    ```

17 Otras Cosas

Algunas cosas que se podrían incluir:

- Aumento de soluciones a problemas
- Extensión del tema sobre seguridad
- `termwrap`
- Sugerencias

De todos modos, si tiene sugerencias, críticas, o cualquier otra cosa que decir sobre este documento, por favor, ¡hágalo!. Como Bill Reynolds está muy ocupado en estos momentos, yo, Patrick Reijen, me he hecho cargo de la autoría del `Term`-COMO. Se me puede encontrar (actualmente) en `patrickr@cs.kun.nl` o en `patrickr@sci.kun.nl`.

18 Reconocimientos

Hay que dar las gracias a mucha gente. Primero y principalmente a Michael o'Reilly y todos los desarrolladores del `term`, que nos han proporcionado esta gran herramienta. También quisiera agradecer a todos los que dieron su experiencia y contribuyeron a este COMO. Esto incluye a Ronald Florence, Tom Payerle, Bill C. Riemers, Hugh Secker-Walker, Matt Welsh, Bill McCarthy, Sergio y todos los que me olvido de mencionar.

19 Anexo: El INSFLUG

El *INSFLUG* forma parte del grupo internacional *Linux Documentation Project*, encargándose de las traducciones al castellano de los Howtos (Comos), así como la producción de documentos originales en aquellos casos en los que no existe análogo en inglés.

En el **INSFLUG** se orienta preferentemente a la traducción de documentos breves, como los *COMOs* y *PUFs* (**P**reguntas de **U**so **F**recuente, las *FAQs*. `:)`), etc.

Diríjase a la sede del INSFLUG para más información al respecto.

En la sede del INSFLUG encontrará siempre las **últimas** versiones de las traducciones: `www.insflug.org`. Asegúrese de comprobar cuál es la última versión disponible en el Insflug antes de bajar un documento de un servidor réplica.

Se proporciona también una lista de los servidores réplica (*mirror*) del Insflug más cercanos a Vd., e información relativa a otros recursos en castellano.

Francisco José Montilla, `pacopepe@insflug.org`.