RDSI COMO

Antonio Verdejo García, averdejog.galileo@nexo.es
y Francisco J. Montilla pacopepe@insflug.org v0.2, 5 de Julio de 1998.

Este COMO explica cómo configurar tarjetas pasivas RDSI para conexiones de red PPP con Linux (a Internet, servidores...). Describe los pasos para dar soporte tanto físico como lógico, así como el método de conexión, con 1 y 2 canales, y con llamada bajo demanda.

Índice General

1 Introducción

Tanto bajo Linux como bajo cualquier sistema operativo (tal vez con la diferencia de que Linux hará exactamente lo que le digamos, para bien o para mal) los pasos lógicos para hacer uso de cualquier periférico, son los siguientes, del orden más físico al más lógico:

1. Comprobar que la marca y modelo del dispositivo están soportados.
2. Integrar el dispositivo físicamente (*pinchar* la tarjeta o conectarla, y demás conexiones accesorias).
3. Comprobar su integración a nivel hardware: en los casos en que sea posible, que el ordenador reconozca dicho dispositivo. Por ejemplo, si el dispositivo es PnP, en BIOS Plug & Pray, aparecerá al encender el ordenador, en alguna de las fases de testeo.
4. Comprobar / configurar qué parametros usa (IRQ, Direcciones de memoria base, etc).
5. Instalar los controladores de acuerdo a dichos parámetros.
6. Instalar el software / utilidades necesarios para el uso efectivo de dichos dispositivos.

Este proceso **debe** debe realizarse siguiendo estrictamente dicho orden, no pasando a la etapa siguiente a menos de que estemos *completamente* seguros de haberse llevado a cabo bien la previa. Y se aplica a la integración de *cualquier* dispositivo bajo *cualquier* Sistema Operativo, si bien no muchos de los que se hacen llamar así hacen honor a su nombre como lo hace Linux `;-)`.

2 Hardware Soportado - Recomendado

Pese a que las conexiones RDSI se pueden llevar a cabo tanto mediante adaptadores externos como tarjetas pasivas internas, en este documento nos centraremos en (y recomendamos) las tarjetas pasivas, por considerarlas una opción mejor respecto a adaptadores externos RDSI, ya que:

1. Su coste es generalmente 5 o 6 veces menor.
2. La diferencia de rendimiento es inexistente.
3. Es independiente de los puertos serie, no afectando al rendimiento o incluso viabilidad, el tipo de *UART* que se tenga (esto es muy importante si pensamos montar una pasarela a Internet con ese *viejo* 486 que rueda por la oficina).

4. El software y los drivers para las tarjetas pasivas internas permiten hacer auténticas *virguerías*, con una perfecta integración, a diferencia de los módems RDSI.

2.1 Modelos de tarjetas pasivas

La gran mayoría de los modelos que se listan a continuación son tarjetas internas pasivas. El número de tarjetas soportadas crece casi con la misma velocidad que se suceden versiones del núcleo. Tenga en cuenta que si posee un adaptador externo (Zyxel o USR) el método que se empleará será el primero descrito en este documento (usando los scripts levemente modificados que se usan en una conexión con un módem analógico).

Por fortuna, este tipo de dispositivos suelen ser poco comunes (por su alto precio y nula diferencia en rendimiento en comparación con una tarjeta interna, además de no soportar el *dial on demand* —llamada bajo demanda, en adelante DoD— integrado directamente en el `ipppd` que sí soportan los dispositivos `ipppX`).

En cualquier caso, los problemas de configuración que presentan se reducen al mínimo, y pasan por ajustar (en determinados casos) la cadena de inicialización del guión de chat.

Nos centraremos pues, casi en exclusiva, en las tarjetas internas.

La lista está sacada de los ficheros `README.*` que acompañan a la parte del arbol de fuentes correspondientes al soporte RDSI que modifica un fichero `.tar.gz` del que hablaremos más adelante en este documento:

- Teles 8.0/16.0/16.3 y compatibles
- Teles 16.3c
- Teles S0/PCMCIA
- Teles PCI
- Teles S0Box
- Creatix S0Box
- Creatix PnP S0
- Compaq ISDN S0 ISA
- AVM A1 (Fritz, Teledat 150)
- ELSA Microlink PCC-16, PCF, PCF-Pro, PCC-8
- ELSA Quickstep 1000
- ELSA Quickstep 1000PCI
- ELSA Quickstep 3000 (igual que una QS1000)
- ELSA Quickstep 3000PCI
- ELSA PCMCIA
- ITK ix1-micro Rev.2
- Eicon.Diehl Diva 2.0 ISA y PCI (S0 y U, PRO no soportada)
- Eicon.Diehl Diva Piccola
- ASUSCOM NETWORK INC. ISDNLink 128K PC adapter (código I-IN100-ST-D)

- Dynalink IS64PH (versión OEM de ASUSCOM NETWORK INC. ISDNLink 128K adapter)
- Tarjetas basadas en HFC-2BS0 (TeleInt SA1)
- Sedlbauer Speed Card (Speed Win, Teledat 100)
- Sedlbauer Speed Star (PCMCIA)
- USR Sportster internal TA (compatible Stollmann tina-pp V3)
- ith Kommunikationstechnik GmbH MIC 16 ISA card
- Traverse Technologie NETjet PCI S0 card
- Dr. Neuhaus Niccy PnP/PCI

  Notas: PCF, PCF-Pro: por ahora, solo la parte ISDN está soportada
- PCC-8: sin testear
- Teles PCMCIA es EXPERIMENTAL
- Teles 16.3c es EXPERIMENTAL
- Teles PCI es EXPERIMENTAL
- Teles S0Box es EXPERIMENTAL
- Eicon.Diehl Diva U interface sin testear
- ICN-ISDN-card
- PCBIT
- ISDN ISA SpellCaster

Tenga en cuenta que esta es una lista general. Muchos de los modelos que aquí aparecen, no se encuentran en el mercado español. La mayoría de las tarjetas son de origen alemán. La PCBIT es un "honrosa" excepción: está fabricada en Portugal.

De las tarjetas distribuidas por Telefónica y manufacturadas por Alcatel (si no recordamos mal), las infames *SPC-2* en cualquiera de sus versiones (y sobre todo en la primera), ni hablaremos.

Ninguno de los que suscriben tienen noticia (pese a utilizar en parte de su circuitería los chips Siemens) de que alguien haya conseguido hacerla funcionar bajo Linux. En cualquier caso, su funcionamiento (bajo un SO que incluya soporte para la misma) deja mucho que desear.

Y como siempre, visita obligada a la documentación que incluye el código fuente del núcleo (máxime si se aventura a usar un kernel de la serie 2.1) bajo `Documentation/isdn` para obtener una lista lo más actualizada posible.

2.2 Soy muy precavido, y estoy leyendo esto antes de comprar la tarjeta. ¿Cuál recomendáis?

Como decíamos en la sección 2, ante todo, tarjetas pasivas internas (que estén soportadas, claro está) frente a adaptadores RDSI externos.

En general, las tarjetas pasivas con circuitería Siemens, y especialmente las que integran el juego de chipsets *HSCX* e *ISAC*, dado que el driver HiSax es el más desarrollado y el que más *empuje* tiene.

Teniendo como criterios lo anterior, el rendimiento, la calidad, y la colaboración que las marcas tienen con Linux en general, y con los desarrolladores de `isdn4linux` en particular:

1. ELSA
2. Creatix
3. Resto de tarjetas soportadas

La *Creatix PnP* es casi equivalente a la *Teles 16.3 PnP*[1], si bien ha sido desarrollada íntegramente por *Creatix*; además de la actitud positiva de *Creatix* respecto Linux, a diferencia de *Teles*.

¡Apoye a los fabricantes que apuestan por Linux!

3 Integración física

Aparte de las precauciones con la estática, (agárrese antes a algún objeto con bastante masa y conexión a tierra, como una tubería o radiador, para descargarla) es conveniente familiarizarnos con el dispositivo, anotar cuando proceda qué chipset tiene, si tiene o no *jumpers*, si es así para qué valen, etc.

Asegúrese de que la tarjeta queda firmemente asentada, más de un problema *inexplicable* se ha debido muchas veces a esto.

En algunas placas base, especialmente las de 486, no da completamente igual dónde se pincha la tarjeta. Familiarícese con su Placa Base.

Los dispositivos RDSI suelen conectarse mediante cables de pares pin-a-pin, si bien no es estrictamente necesario que el cable sea de pares. El conector suele ser un RJ-45, idéntico a los usados para redes. Normalmente la tarjeta traerá un cable, pero si no es así, o lo necesita más largo, con un cable de red UTP corriente valdrá. Teóricamente, y en configuraciones del bus pasivo (instalación telefónica propiamente dicha) típicas, el cable puede ser de unos centenares de metros, si bien no hemos comprobado nunca de forma práctica este particular.

Una última advertencia, para aquellos que están leyendo esto para instalar en una empresa: los dispositivos RDSI suelen llevarse **muy mal** con las centralitas, especialmente con las Siemens.

Si quiere ahorrarse quebraderos de cabeza, malos funcionamientos, interminables actualizaciones del firmware de la centralita, y en la mayoría de los casos, para que simplemente funcione, exija que el bus pasivo RDSI para la tarjeta sea **dedicado** y **directo**. Se ahorrará infinidad de problemas, y el funcionamiento será el que un entorno de producción exige.

4 Configuración BIOS

La mayoría de las tarjetas de hoy en día son Plug & Play, y esto, aunque parezca mentira, en BIOS con PnP es a veces una ventaja; la mayoría de ellas muestran al arrancar los dispositivos PnP que han encontrado, por lo que si éste es su caso, y no le aparece nada, puede tener la absoluta certeza de que para el ordenador es como si no existiese. En algunas placas, hay que especificar qué recursos se dejan para asignar a los dispositivos PnP.

En el resto de los casos, en combinaciones de placas / dispositivos no Plug & Play, puede ser necesario efectuar algún retoque en la BIOS, por ejemplo, si nuestra BIOS es PnP, pero el dispositivo no, posiblemente deba reservar recursos y/o asignarlos en la BIOS para él.

[1]Ojo, **NO** la *Teles 16.3c PnP*, que pese a estar soportada experimentalmente, no tiene nada que ver en cuanto a calidad

5 Configuración de recursos usados por el dispositivo.

5.1 Dispositivos Plug & Play

Bajo Linux, y mientras se trabaja en un soporte directo en el kernel para este "*estándar*", habremos de usar las herramientas del paquete `isapnptools` para asignar los recursos precisos al dispositivo. Como su nombre indica, **solo valen para dispositivos PnP ISA**, no para los PCI (que de todas formas, casi siempre han sido PnP en cuanto a enchufar y listo, no al *estándar*). La mayoría de los servidores ftp que albergan contenidos Linux las tienen, así como las distribuciones Linux más populares.

Para configurar la tarjeta, use el programa `pnpdump` y vuelque su salida a un fichero, por ejemplo, a `/tmp/isapnp.conf`.

Deberá editarlo para reflejar los valores correctos. Una vez hecho esto, con `isapnp /tmp/isapnp.conf` tendrá la tarjeta lista.

Luego de haber comprobado que los valores son correctos, y que la tarjeta se inicializa correctamente, guarde el fichero definitivamente, en `/etc/isapnp.conf`.

Al arrancar (y suponiendo que haya instalado o tuviera ya instaladas correctamente las pnptools) los scripts de inicialización se encargarán de todo automáticamente. En cualquier caso, y si viera que `isapnp` no se ejecuta al arrancar el Linux, siempre le queda la solución de incluirlo en `/etc/rc.d/rc.local` o similar, o, en el peor de los casos, ejecutarlo a mano.

El fichero generado por `pnpdump` del siguiente modo

```
[root@hal /root]# pnpdump > /tmp/isapnp.conf
```

y suponiendo que sólo tenga una tarjeta PnP, una *Teles.SO 16.3c PnP* en este caso, si tiene una SoundBlaster PnP, esto estará al final generalmente, y será similar a esto:

```
# $Id: pnpdump.c,v 1.8 1997/01/14 21:05:35 fox Exp $
# This is free software, see the sources for details.
# This software has NO WARRANTY, use at your OWN RISK
#
# For details of this file format, see isapnp.conf(5)
#
# Compiler flags: -DREALTIME
#
# Trying port address 0203
# Board 1 has serial identifier 0d 1a 09 0b 44 10 26 27 50
# (DEBUG)
(READPORT 0x0203)
(ISOLATE)
(IDENTIFY *)

# Card 1: (serial identifier 0d 1a 09 0b 44 10 26 27 50)
# TAG2610 Serial No 436800324 [checksum 0d]
# Version 1.0, Vendor version 1.1
# ANSI string -->TELES.S0/16.3c Plug&Play<--
#
# Logical device id TAG2610
#     Device support I/O range check register
#
# Edit the entries below to uncomment out the configuration required.
# Note that only the first value of any range is given, this may be
changed if $# Don't forget to uncomment the activate (ACT Y) when happy
```

```
(CONFIGURE TAG2610/436800324 (LD 0
# Multiple choice time, choose one only !

#     Start dependent functions: priority preferred
#       Logical device decodes 16 bit IO address lines
#             Minimum IO base address 0x0580
#             Maximum IO base address 0x05bc
#             IO base alignment 4 bytes
#             Number of IO addresses required: 2
# (IO 0 (BASE 0x0580))
#       IRQ 3, 4, 5, 9, 10, 11, 12 or 15.
#             High true, edge sensitive interrupt (by default)
# (INT 0 (IRQ 3 (MODE +E)))

#       Start dependent functions: priority acceptable
#       Logical device decodes 16 bit IO address lines
#             Minimum IO base address 0x0500
#             Maximum IO base address 0x05bc
#             IO base alignment 4 bytes
#             Number of IO addresses required: 2
# (IO 0 (BASE 0x0500))
#       IRQ 10, 11 or 12.
#             High true, edge sensitive interrupt (by default)
# (INT 0 (IRQ 10 (MODE +E)))
#       Start dependent functions: priority acceptable
#       Logical device decodes 16 bit IO address lines
#             Minimum IO base address 0x0680
#             Maximum IO base address 0x06bc
#             IO base alignment 4 bytes
#             Number of IO addresses required: 2
# (IO 0 (BASE 0x0680))
#       IRQ 10, 11 or 12.
#             High true, edge sensitive interrupt (by default)
# (INT 0 (IRQ 10 (MODE +E)))

#       Start dependent functions: priority functional
#       Logical device decodes 16 bit IO address lines
#             Minimum IO base address 0x1500
#             Maximum IO base address 0x17fc
#             IO base alignment 4 bytes
#             Number of IO addresses required: 2
# (IO 0 (BASE 0x1500))
#       IRQ 3, 4, 5, 9, 10, 11, 12 or 15.
#             High true, edge sensitive interrupt (by default)
# (INT 0 (IRQ 3 (MODE +E)))

#     End dependent functions
#     Vendor defined tag:  84 0f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
#     Vendor defined tag:  84 06 00 00 00 00 00
# (ACT Y)
))
# End tag... Checksum 0x00 (OK)
```

Simplemente hay que dejar una de las secciones `(IO...   )` e `(INT...)` eliminando los comentarios, y asegurarse (esto es importante) de eliminar el último comentario de la línea donde se lee `# (ACT Y)` para *activar* la incialización de la tarjeta con los valores escogidos...

Es conveniente anotar dichos valores, ya que los que tendremos que utilizar posteriormente (anótelos).

Y ni que decir tiene que no debemos asignarle recursos que ya estén siendo usados por otros dispositivos. Familiarícese con su sistema.

Un `cat /proc/interrupts` o un `cat /proc/ioports` le ayudará, especialmente **antes** de instalar la tarjeta en el ordenador, siempre y cuando todos los dispositivos que tenga su ordenador sean reconocidos por Linux, ya que los que no lo sean no aparecerán en los listados y no podremos saber qué recursos están usando.

Refiérase a la sección 4.

Un fichero `/etc/isapnp.conf`, una vez eliminados todos los comentarios suele tener este aspecto:

```
(READPORT 0x0203)
(ISOLATE)
(IDENTIFY *)
(CONFIGURE TAG2610/436800324 (LD 0
(IO 0 (BASE 0x0580))
(INT 0 (IRQ 3 (MODE +E)))
(ACT Y)
))
```

La salida del comando `isapnp /etc/isapnp.conf`, bien sea a mano o durante el arranque del sistema, suele ser así:

```
[root@hal /root]# isapnp /tmp/isapnp.conf
Board 1 has Identity 0d 1a 09 0b 44 10 26 27 50:  TAG2610 Serial No 436800324 [checksum 0d]
```

5.2 Configuración de dispositivos NO PnP

Se supone que ha leído, entendido, y llevado a cabo con la absoluta certeza de haberlo hecho bien, la sección 4.

No conocemos todos los dispositivos NO PnP disponibles en el mercado que funcionan con Linux. Pero la experiencia muestra que generalmente, para su configuración tiene las siguientes opciones:

- Usar alguna utilidad, generalmente bajo DOS o Windows.
- Usar Jumpers del dispositivo si los tiene
- Usar una utilidad para linux (en contadísimas ocasiones)

Normalmente, la más cómoda y fiable es la de los jumpers, ya que no deberemos preocuparnos de si los reset borran la configuración o no, aunque en algunas tarjetas (*Teles.SO 16.3 NO PnP* por ejemplo) sólo posibilitan la asignación de *IO*s. (Ojo, con esta tarjeta, son unos microrruptores muy pequeños, generalmente con un poco de grasa por encima).

En el primer caso, si son utilidades DOS, siempre podremos arrancar con ese disquete antediluviano que rueda por el cajón, y configurar. Si es windows, y se tiene instalado también, tal vez tras unas encomiendas a San Pancracio, si Murphy está distraído, y la suerte está de nuestro lado, consigamos convencerla de que use los recursos que queremos.

En sistemas en los que *afortunadamente* no esté instalado, siempre podemos probar a pincharla en uno que sí lo tenga, configurarla, y volverla a pinchar en el sistema Linux, si bien no siempre funciona.

Otra posibilidad, si la suerte acompaña, es comprobar (la mayor parte de las veces mediante ensayo-error, y no siempre con absoluta certeza, aunque un vistazo a la documentación de la tarjeta ayuda bastante) qué parámetros por defecto tiene el dispositivo de fábrica, y usarlos, siempre que no entren en conflicto con otros que ya tengamos instalados; si es así, dependiendo de dichos dispositivos puede ser hasta más cómodo reconfigurarlos y dejar *hueco* al nuevo *inquilino*.

Recuerde (incluso anote, insistimos) qué parámetros va a usar. Los necesitará más adelante.

6 Instalación y configuración de controladores

Los controladores son el software que hace funcionar al dispositivo, y que da soporte lógico al Sistema Operativo para interactuar con él. En Linux la integración de este soporte se lleva a cabo configurando y compilando el núcleo o kernel, con lo que obtenemos un *corazón* del Sistema Operativo *a la medida* de cada máquina.

Linux ofrece la posibilidad de compilarlo íntegro en el kernel o en módulos aparte, que se cargan según los necesite el sistema o no. Si no está familiarizado con todo esto, es momento de que lea el *Kernel-Como*, disponible en el Insflug `http://www.insflug.org`.

El kernel necesitará tener dos tipos de soporte; uno genérico, (módulo `isdn`) y otro específico a la tarjeta (`hisax`, etc).

Como algunas tarjetas RDSI, especialmente las que tienen soporte experimental, sólo funcionan con controladores específicos modulares, nos centraremos en este tipo de soporte, por ser más *universal*.

No obstante, en los ejemplos supondremos que hacemos uso de kernels estables (`2.0.xx`), aunque tengamos que *parchearlos*. Puede usar kernels de desarrollo si lo prefiere, tan sólo téngalo en cuenta en los ejemplos que aplique y modifíquelos en consecuencia, sin olvidar que estos kernels son para lo que son, **desarrollo**, no siendo muy idóneos para la instalación por primera vez de algo que se desconoce.

6.1 Soporte específico a la tarjeta

Antes de continuar, suponemos que ha leído la sección 2.1 y que sabe a ciencia cierta que su tarjeta está soportada.

Si no parece estarlo, es conveniente que lea (sí, bueno, relea ;-) de todos modos la documentación que hay en `/usr/src/linux/Documentation/isdn` que siempre estará más actualizada que este *Como*. Si no, no está todo perdido; obtenga el último árbol de desarrollo de `ftp://ftp.suse.com/pub/isdn4linux/v2.0/isdn.tar.gz` y eche un vistazo a los ficheros de `isdn/Documentation/isdn/`, puede que se lleve una grata sorpresa.

Si su tarjeta está soportada en la distribución estándar del kernel actual (2.0.34 a día de hoy), salte a la sección 6.2. Si necesita parchear, siga leyendo.

Para añadir soporte al kernel actual, integraremos una parte del árbol de fuentes modificada, que añada soporte para la misma. Obtenga el fichero `ftp://ftp.suse.com/pub/isdn4linux/v2.0/isdn.tar.gz`, suele ser un enlace simbólico a la última versión del árbol de desarrollo RDSI disponible.

No obstante... el soporte es experimental. Casos típicos son los de las últimamente disponibles popularmente *Teles.SO 16.3c* o las *Asuscom*. Los que suscriben no han visto nada anormal (cierto es que el tiempo de que hemos dispuesto para testearlas ha sido breve) y tienen noticias de varios servidores de los llamados *de producción* que están funcionando sin problemas con kernels estables y estas tarjetas.

No obstante, no se parchea en el sentido estricto, ya que lo único que se sustituye es la parte correspondiente a RDSI.

La parte del árbol modificada lleva un fichero llamado `std2kern` que hace el trabajo de parcheo por nosotros, siempre y cuando `/usr/src/linux` sea el directorio donde residan las fuentes de linux. Asegúrese de que exista; si en su sistema el directorio se llama `/usr/src/linux-2.0.33`, compruebe, y en su ausencia cree un enlace al mismo llamado `linux`; por ejemplo:

```
cd /usr/src ;
ln -s linux-2.0.33 linux
```

Descomprima el árbol de fuentes `isdn`: suponiendo que ha dejado el fichero en `/tmp`:

```
( cd /usr/src; tar zxvf /tmp/isdn.tar.gz )
```

Acceda a `/usr/src/isdn`, y ejecute el comando `std2kern -d`:

```
cd /usr/src/isdn
./std2kern -d
```

no olvide el "./" para dar el path directo al fichero, en la mayoría de los sistemas el directorio actual no está en el `PATH` por seguridad.

Compruebe que no se producen mensajes de error. Si es así, averigüe qué sucede. Lo más típico es que se haya equivocado en la elección de fichero, y haya escogido uno para un kernel de otra versión (2.1.xx por ejemplo).

6.2 Configuración del Kernel

Una vez hemos llevado a cabo los pasos anteriores procederemos a la configuración y posterior recompilación del kernel. Si no está habituado a esto, léase primero el *Kernel-Como*, disponible en Insflug, vea sección 12.

Acceda a `/usr/src/linux` y ejecute su método preferido de configuración. Asegúrese de activar, en la sección principal, *Code maturity level options* el apartado *Prompt for development and/or incomplete code/drivers*, o de lo contrario, el programa de configuración no le dará opción a seleccionar controladores experimentales.

Una vez hecho esto, seleccione:

6.2.1 Soporte genérico en el kernel

Vaya a la sección *ISDN subsystem* del menú principal:

- *ISDN support* como módulo (`M`).
- *Support synchronous PPP*
- *Support generic MP (RFC 1717)* (potestativo, necesario para canales múltiples)
- *Support audio via ISDN* (potestativo)

Esto es para cuanto a soporte RDSI se refiere. En cuanto a soporte PPP, cuestiones específicas de redes, y demás aspectos, recurra al *Como* apropiado.

6.2.2 Soporte específico a la tarjeta

En la sección *ISDN subsystem* del menú principal, active el controlador que dé soporte a su tarjeta. El más popular es el HiSax, si ese es su caso, deberá además especificar:

- Protocolo a soportar: en nuestro caso, *HiSax Support for EURO/DSS1* y
- Cuál de la familia de tarjetas soportadas por él es la suya; si por ejemplo es la veterana Teles.SO 16.3 NO PnP, la PnP, o la pcmcia (**NO la 16.3c OJO**) seleccionaría *HiSax Support for Teles 16.3 or PNP or PCMCIA*.

De nuevo, no conocemos ni podemos conocer todas las tarjetas soportadas por Linux. Es posible que en drivers experimentales haya que indicar alguna otra opción; recurra a su sentido común, a la documentación (a la que no nos cansaremos de remitirle; este documento no es más que una guía) y a nosotros, a fin de actualizar este *Como*.

Salga del menú guardando los cambios, y compile; no olvide el `make modules` y el `make modules_install`, y reinstalar el LILO para dicho kernel.

Para más información de cómo recompilar el kernel, véase el *Kernel-Como*, disponible en el Insflug, vea sección 12.

6.3 Carga de los módulos - comprobación del sistema

Ya he recompilado, instalado los módulos, y arrancado con el nuevo kernel. Además, he usado `isapnp` *y todo parece haber ido bien... ¿Qué hago ahora?*

Se ha ganado un descanso. Tómese algo... ;-) No, en serio. Ahora viene la parte más interesante.

Hay varias formas de cargar los módulos, en cualquier caso, la manera que nunca falla es hacerlo a mano directamente desde la línea de comandos. Supondremos que hacemos uso del soporte específico HiSax. La sintaxis del módulo hisax es la que sigue, si bien es conveniente leer (al final lo conseguiremos ;-), especialmente en drivers experimentales, `/usr/src/linux/Documentation/isdn/README.HiSax`.

```
modprobe hisax type=<codigo tarjeta> protocol=<protocolo> io=<direccion E/S> irq=<interrupcion>
```

Ha llegado el momento de echar mano de donde tuviera anotados (¿los anotó?) los parámetros que asignara en las secciones 4 y 5.

Suponiendo que se trate de la tarjeta *Teles.SO 16.3c PnP*, que al fin y al cabo, fue la causante en origen de este *Como*:

```
modprobe hisax type=14 protocol=2 io=<IO> irq=<IRQ>
```

Por ejemplo:

```
modprobe hisax type=14 protocol=2 io=0x0580 irq=11
```

con lo que si miramos en `/var/log/messages` deberíamos ver algo como:

```
Jun 23 12:05:11 hal kernel: HiSax: Driver for Siemens chip set ISDN cards
Jun 23 12:05:11 hal kernel: HiSax: Version 2.8
Jun 23 12:05:11 hal kernel: HiSax: Revisions 1.15.2.8/1.10.2.5/1.10.2.3/1.30.2.6/1.8.2.5
Jun 23 12:05:11 hal kernel: HiSax: Card 1 Protocol EDSS1 Id=HiSax (0)
Jun 23 12:05:11 hal kernel: HiSax: Teles 16.3c driver Rev. 1.1.2.2
Jun 23 12:05:11 hal kernel: teles3c: defined at 0x580 IRQ 3 HZ 100
```

```
Jun 23 12:05:11 hal kernel: teles3c: resetting card
Jun 23 12:05:11 hal kernel: Teles 16.3c: IRQ 11 count 0
Jun 23 12:05:11 hal kernel: Teles 16.3c: IRQ 11 count 1
Jun 23 12:05:11 hal kernel: HiSax: DSS1 Rev. 1.16.2.3
Jun 23 12:05:11 hal kernel: HiSax: 2 channels added
Jun 23 12:05:11 hal kernel: HiSax: module installed
```

El tipo `14` es el que se corresponde con la *Teles 16.3c PnP*, el protocolo `2` es el usado en España para las conexiones RDSI, *EURO ISDN o EDSS1*. Los otros dos valores (dirección de E/S e interrupción) dependerán de su configuración particular, que anotó en su momento, ¿verdad?

Dependiendo del driver, este puede que se cargue aun con parámetros erróneos, si bien no es el caso del HiSax, que rehusará a hacerlo.

Si sospechamos que pese a haberse cargado (repetimos, no en el caso del HiSax) hay por ejemplo conflictos de IRQ, o no está usando la que le hemos asignado, un indicador claro de esto es que al hacer un

```
cat /proc/interrupts
 0:    9719062   timer
 1:     342221   keyboard
 2:          0   cascade
 4:     495989 + serial
10:    1591809   ICN
12:        681   eth0
13:          1   math error
```

en un sistema con una tarjeta ICN la línea correspondiente a la irq usada por el controlador contase con `0` interrupciones de contador (segunda columna). Esto aplica para todos los dispositivos; si la línea

```
10:    1591809   ICN
```

fuese

```
10:    0         ICN
```

sería un claro síntoma de que el driver `ICN` no está usando dicha interrupción, casi seguro por fallo de configuración. Tan sólo por cargar correctamente, debe de poner el contador a `1` al menos.

Llegados a este punto, respire profundamente y siéntase todo un *gurú* Linuxero... ;-) Ya casi está listo; para no tener que hacerlo en un futuro a mano, y suponiendo que tiene las `modutils` correctamente instaladas, edite o cree su `/etc/conf.modules` o `/etc/modules.conf` e inserte las siguientes líneas, (suponiendo que use por ejemplo una Teles 16.3 NO PnP/ con la IRQ 10 y la io 0x180 :

```
alias isdn hisax
options hisax type=3 protocol=2 io=0x180 irq=10
```

ejecute `depmod -a` para computar/actualizar las dependencias entre módulos; de ahora en adelante un `modprobe hisax` bastará.

7 Instalación y configuración de software de aplicación

Mi tarjeta parece que ya está lista. ¿Puedo usar los scripts de conexión a Infovía que usaba hasta ahora?

No tal cual; necesitará hacer ciertas modificaciones. Usaremos otro método para conectarnos a iNET. En vez de usar el pppd asíncrono de toda la vida, usaremos un pppd especial, síncrono, que permite algunas *lindezas*: el `ipppd`.

Arranque su cliente ftp favorito, y diríjase a
`ftp://ftp.franken.de/pub/isdn4linux/v2.0/isdn4linux*.tar.gz` que es el sitio oficial del ISDN4Linux. Ahí tiene una mágnifica (aunque algo falta de actualización) FAQ en un perfecto inglés que debería tener al menos como punto de referencia.

Le remitiríamos a ella, pero si ha llegado hasta aquí y hacemos eso igual empezamos a sentir agudos pitidos en los oidos... ;-)

Descomprimimos, configuramos, compilamos e instalamos. De la lista de utilidades las que más nos interesan, son `isdnctrl` (directorio `isdn`) y el `ipppd` (directorio `ppp4i4k/ipppd/version`) porque son las que usaremos en el método que describiremos después.

Normalmente, casi todas las distribuciones suelen llevar un paquete de utilidades RDSI que incluyen los programas que mencionamos, amén de abundante documentación y scripts de ejemplo. Busque en su distribución favorita.

No obstante, si por alguna razón no consigue compilar los elementos necesarios, en
`ftp://ftp.insflug.org/pub/RDSI/` tiene a su disposición el software mínimo necesario ya compilado.

Como en todo sistema UN*X la comunicación con los dispositivos físicos (tarjetas, discos...) se realiza por medio de ficheros. Necesitaremos crear los dispositivos que harán que el kernel pueda trabajar con la tarjeta RDSI. Si usa un paquete de una distribución es casi seguro que creará, si no lo están ya, las entradas necesarias en el directorio `/dev`, si no es así, ejecute `make devices` en el directorio raíz de las `isdn4utils` que bajó antes, será sufiente.

7.1 Pero bueno, ¡¿qué cómo conectooo?!

Vamos con ello. Dos métodos, uno de ellos mencionado someramente. Se basa en aprovechar los scripts de conexión (que suponemos le funcionan) usados con un módem analógico normal. Las variaciones son mínimas. Añada en el guión de chat la cadena de inicio

```
ATS14=3&xxxxxxxxx (siendo xxxxxxxxx el numero de su linea RDSI)
```

y sustituya donde corresponda el dispositivo `/dev/modem` por `/dev/ttyI0`. Usaremos el `pppd` normal y corriente que usábamos antes con el módem. Nada más que decir de este método, salvo que no haga uso del parámetro `+ua` en el fichero `options`, está obsoleto en las últimas versiones del paquete `pppd`. .

El segundo hace uso de las utilidades que nos bajamos anteriormente, y nos permitirá conseguir llamadas bajo demanda (*dial on demand*, DoD).

Opción ésta muy interesante en redes donde se vaya a usar la conexión RDSI para dar servicio iNET, por medio del enmascaramiento IP, a varios puestos de una red local, pues posibilitará el que la llamada se efectúe automáticamente por tráfico de paquetes (abrir un navegador, lanzar el programa de correo, hacer un ping, etc.).

La parte más importante de este método reside en los scripts usados para configurar la conexión. Los hay de múltiples formas, más o menos "sofisticados". Los incluidos en este documento puede que no sean para ganar un Nobel, pero funcionan bastante bien. En este sentido, estamos abiertos (no hace falta decirlo) a modificaciones y/o comentarios, pero de eso hablaremos más tarde.

Unos puntos a destacar. Si queremos usar *DoD*, necesitaremos tener dos scripts en `/etc/ppp` también incluidos, para asegurarnos que la ruta por defecto *apunte* siempre a una dirección de iNET y al dispositivo RDSI.

Esto, y, por supuesto, NO tener ninguna ruta por defecto a la(s) tarjeta(s) de red (ethernet normalmente) que ya tuviéramos en nuestro sistema: el demonio de PPP (pppd o ipppd) no reemplaza la ruta por defecto, es un problema muy común en los grupos de noticias y en los canales de Linux de IRC.

El síntoma es que la conexión se establece, pero no podemos *salir* a iNET porque no tenemos *señalizado* por dónde hacerlo. No es el propósito de este documento extenderse demasiado en temas de rutado, pero en condiciones normales, no necesitaremos ruta por defecto, podemos usar rutas estáticas; dejaremos que el (i)pppd la establezca cuando así sea necesario.

Y será uno de los scripts (`ip-down`) el que se encargará de que en todo momento haya una ruta por defecto a iNET por la tarjeta RDSI.

7.2 Scripts

Hace cosa de un mes fueron enviados a la lista de correo (¿aún no está suscrito? ¿a qué espera? ;-) del SLUG (`l-linux@calvo.teleco.ulpgc.es`), de modo que si está suscrito y no borra los mensajes, imagino que los tendrá.

Pero como no todo el mundo está en dicha lista (y este Como, que duda cabe, no sería tal sin ellos), aquí van:

7.2.1 `rc.isdn` para un solo canal

```
#!/bin/sh
#
# Thanks to Rainer Birkenmaier <rainer@kirk.mop.uni.ulm.de>
# Hacked by Antonio Verdejo Garcia <averdejog.galileo@nexo.es>
# & Francisco J Montilla <pacopepe@insflug.org>

PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin

LOCAL_NUMBER="xxxxxxxxx"
REMOTE_NUMBER="xxx"
LOCAL_IP="195.76.154.169" # IP falsa por la que establecer ruta por
                          # defecto, a fin de que salte el DoD
DEVICE="ippp0"
USER="user@ISP"

isdnctrl addif  $DEVICE                 # Creamos un interfaz nuevo,'DEVICE'
isdnctrl addphone $DEVICE out $REMOTE_NUMBER    # Numero al que llamar
isdnctrl eaz $DEVICE $LOCAL_NUMBER      # EAZ: el numero de su RDSI
isdnctrl l2_prot $DEVICE hdlc           # para PPP sincrono
isdnctrl l3_prot $DEVICE trans          #
isdnctrl encap $DEVICE syncppp          # encapsulacion de paquetes IP en
                                        # en  tramas PPP
isdnctrl huptimeout $DEVICE 300         # tiempo de inactividad tras el que
                                        # desconectar: 300 sec. -> 5min
isdnctrl chargehup $DEVICE off          # Colgar antes del siguiente paso
isdnctrl secure $DEVICE on              # Aceptar llamadas de numeros
                                        # autorizados solamente
ifconfig $DEVICE $LOCAL_IP
route add -net 195.76.154.0 $DEVICE
route add default $DEVICE

/sbin/ipppd user $USER remotename infovia -d defaultroute noipdefault \
ipcp-accept-local ipcp-accept-remote mru 1500 mtu 1500 lock -bsdcomp -pc -ac /dev/ippp0 &
```

las últimas dos líneas son una en realidad; puede indicar que se interprete como una sola tal y como se hace en el script con el \; o bien ponerlo en una sola línea sin retorno de carro.

Asegúrese de que `ipppd` está en `/sbin` si transcribe tal cual este script; si no es así, modifique el path en el script.

Vea la sección 7.2.3 para una explicación acerca de qué parámetros ha de modificar y una explicación sobre este script.

7.2.2 `rc.isdn` para dos canales

```
#!/bin/sh
#
# Thanks to Rainer Birkenmaier <rainer@kirk.mop.uni.ulm.de>
# Hacked by Antonio Verdejo Garcia <averdejog.galileo@nexo.es>

PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin

LOCAL_NUMBER="xxxxxxxxx"
REMOTE_NUMBER="xxx"
LOCAL_IP="195.76.154.169" # dummy; the IPCP negotiation overwrite it
DEVICE="ippp0"
USER="user@ISP"

# additional for channel bundling:
DEVICE1="ippp128"

isdnctrl addif  $DEVICE                # Create new interface 'DEVICE'
isdnctrl addphone $DEVICE out $REMOTE_NUMBER    # Set outgoung phone-number
isdnctrl eaz $DEVICE $LOCAL_NUMBER     # Set local EAZ ..
isdnctrl l2_prot $DEVICE hdlc          # for sync PPP: set Level 2 to HDLC
isdnctrl l3_prot $DEVICE trans         # not really necessary, 'trans' is default
isdnctrl encap $DEVICE syncppp         # encap the IP Pakets in PPP frames
isdnctrl huptimeout $DEVICE 300        # Hangup-Timeout is 300 sec. -> 5 min
isdnctrl chargehup $DEVICE off         # Hangup before next Charge-Info
isdnctrl secure $DEVICE on             # Accept only configured phone-number

# additional for channel bundling:
isdnctrl addslave $DEVICE $DEVICE1     # Create new slave interface 'DEVICE1'
isdnctrl addphone $DEVICE1 out $REMOTE_NUMBER   # Set outgoung phone-number
isdnctrl eaz $DEVICE1 $LOCAL_NUMBER    # Set local EAZ ..
isdnctrl l2_prot $DEVICE1 hdlc         # for sync PPP: set Level 2 to HDLC
isdnctrl l3_prot $DEVICE1 trans        # not really necessary, 'trans' is default
isdnctrl encap $DEVICE1 syncppp        # encap the IP Pakets in PPP frames
isdnctrl huptimeout $DEVICE1 300       # Hangup-Timeout is 300 sec. -> 5 min
isdnctrl chargehup $DEVICE1 off        # Hangup before next Charge-Info
isdnctrl secure $DEVICE1 on            # Accept only configured phone-number

ifconfig $DEVICE $LOCAL_IP
route add -net 195.76.154.0 $DEVICE
route add default $DEVICE

/sbin/ipppd user $USER remotename infovia -d defaultroute noipdefault ipcp-accept-local \
ipcp-accept-remote mru 1500 mtu 1500 +mp lock -bsdcomp -pc -ac /dev/ippp0 /dev/ippp1 &
```

7.2.3 Explicación de los scripts

Los scripts no necesitan demasiadas explicaciones. Sustituir `user` e `ISP` por su nombre de usuario y el nombre de su proveedor (`pepe@arrakis` por ejemplo) y poner los valores adecuados en `LOCAL_NUMBER` (el número de su RDSI) y en `REMOTE_NUMBER` (`055` si usa Infovía).

La dirección de `LOCAL_IP` es una dirección falsa, la negociación IPCP la sobreescribe, pero por una simple razón de coherencia, conviene darle una IP válida del rango de su proveedor, y asignarle a ella la ruta por defecto, (lo mismo se aplica para la dirección de red de la ruta del final del script) esto es necesario para que funcione el DoD.

Las direcciones del ejemplo son de Intercom, pero valen de cualquier manera (funciona también usando las mismas con otros proveedores). Estas direcciones son las mismas que aparecen en los scripts `ip-up` e `ip-down`:

7.2.4 ip-up

```
#!/bin/sh
/sbin/route del default
/sbin/route add default ippp0
```

7.2.5 ip-down

```
#!/bin/sh
/sbin/route del default
/sbin/ifconfig ippp0 down
/sbin/ifconfig ippp0 195.176.154.169
/sbin/route add -net 195.176.154.0 ippp0
/sbin/route add default ippp0
```

Es posible que alguno de los comandos que aparecen en estos dos últimos guiones sean redundantes. De nuevo, estamos abiertos a sugerencias.

El `rc.isdn` de la sección 7.2.2 está preparado para el uso de dos canales y por lo tanto una conexión a 128 Kbps, usando uno de los canales como esclavo del primero. La opción `+mp` es necesaria en este caso, además de que haya seleccionado en la compilación del kernel, en la sección general de RDSI, *Support Generic MP (RFC 1717)*. (Compruebe que exista la línea `CONFIG_ISDN_MPP=y` en el fichero `/usr/src/linux/.config`, que es donde se almacena por defecto la configuración del núcleo).

Tenga en cuenta que, como es lógico, pagará el *doble*... Aunque esto en empresas no suele ser un problema, cuidado en casa, o verá como las facturas de Telefónica tienden a infinito... ;-)

Para lanzar manualmente el segundo canal, ejecute `isdnctrl addslave ippp128`; colgará automáticamente tras un periodo sin tráfico, tardando lo que hayamos especificado en el parámetro `huptimeout` del `rc.isdn` (en segundos).

Con determinados proveedores no se nota demasiado el lanzar el segundo canal (Arrakis), con otros sin embargo, y también dependiendo del origen de nuestro tráfico, si se nota, y bastante...

Hay un demonio que se encarga de disparar/colgar el segundo canal según el tráfico y la saturación que detecte; puede obtenerse de `http://www.compound.se`.

En futuras versiones, tendrá sección propia; por ahora, si tiene un trabajo donde permitirse eso, se supone que tendrá nivel como para manejarse con él sin problemas.

8 Problemas Frecuentes

8.1 Al lanzar la conexión miro el `/var/log/messages` y sólo veo (una vez tras otra):

```
Apr 15 10:34:08 wanda kernel: ippp0: dialing 0 055...
Apr 15 10:34:08 wanda kernel: ippp0: dialing 1 055...
Apr 15 10:34:08 wanda kernel: ippp0: dialing 2 055...
```

pero no veo nada más, ¿a qué puede ser debido?

Es un problema físico. Revise la conexión del cable tanto en la tarjeta como en el TR1. Revise la continuidad del cable así mismo. Cámbielo en último término. Asegúrese de que su TR1 tiene servicio... `;-)` y Asegúrese de no estar pasando por ninguna centralita.

8.2 La conexión se corta tras un mensaje como:

```
Apr 15 15:58:28 wanda pppd[208]: Could not determine remote IP address
```

y seguidamente:

```
Apr 15 15:58:28 wanda pppd[208]: LCP terminated at peer's request
Apr 15 15:58:28 wanda kernel: isdn_net: local hangup ippp0
Apr 15 15:58:28 wanda kernel: ippp0: Chargesum is 0
Apr 15 15:58:28 wanda pppd[208]: Modem hangup
Apr 15 15:58:28 wanda pppd[208]: Connection terminated.
```

Es un problema bastante común debido a que Infovía (en el supuesto de que la use para conectar) no nos asigna, —o no lo hace con suficiente rapidez— una dirección remota del enlace PPP. Hay un *solución* que funciona tanto en conexiones RDSI como RTC que consiste en pasarle nosotros una dirección en el establecimiento de la conexión. En el caso de conexiones vía RTC (módem corriente y moliente) incluya una línea en el `/etc/ppp/options` tal que:

```
:172.16.1.96
```

y deje el parámetro que le indica que, a pesar de todo, aceptaremos la IP que el extremo nos asigne como remota (`ipcp-accept-remote`). La IP que pongamos puede ser cualquiera, pero como siempre, y por seguir una regla, ponga una de las que normalmente nos asigna Infovía de su rango (`172.16.x.x` por ejemplo).

Gracias a Horacio J. Peña por este detalle (el primero al que se lo leimos en la lista del SLUG).

El caso de conexiones vía RDSI (sobre todo en el caso de que usemos el primer método) se puede proceder de la misma forma, pues aunque se le pasen parámetros al (i)pppd, el demonio leerá el fichero `/etc/ppp/options`.

8.3 Al inicializar el demonio ipppd obtengo el mensaje "`Can't find usable ippp device''`. ¿A qué es debido?

Según Frank Meyer, del grupo de desarrollo *isdn4linux*, se debe a que al lanzar el `ipppd`, este calcula un número aleatorio basándose en la función `gethostid()` que provoca una resolución DNS, usando para ello el servidor de nombres que aparezca en `/etc/resolv.conf`.

Si no tenemos la conexión activa, esto lógicamente no es posible y el DNS no puede ser alcanzado (y hablamos en el caso general de que no se disponga de un DNS local, como suele suceder comúnmente).

Para solucionarlo, incluya el nombre de su máquina (incluido `localhost`) en el `/etc/hosts` con el dominio completo que haya especificado en `/etc/resolv.conf`. Hay otra solución basada en un parche no oficial para evitar este comportamiento por parte del `ipppd`; el fichero `syncPPP FAQ` incluído en el directorio de documentación de las utilidades ISDN amplía este tema.

9 Por Hacer

- Por supuesto, integrar los comentarios y sugerencias que nos manden amablemente en este documento. Realimentación, graciaaas.
- Estudiar el `ibod` para la gestión dinámica de conexiones a 128K por demanda de tráfico.
- ToDo lo que se nos vaya ocurriendo... ;-)

10 Copyright y Propiedad Intelectual

11 Colofón

Y bueno, por ahora esto es todo. Como una primera versión que es, estará plagada de pequeños (igual otros no tan pequeños) fallos, incorrecciones y seguro que nos dejamos un montón de temas en el tintero.

Eso sí, como hemos mencionado varias veces, nuestro buzón de correo está abierto a todo tipo de sugerencias, correcciones, dudas (que si humildemente podemos, intentaremos responder), así como desinteresadas donaciones para adquirir otras tarjetas... `};)` Lo que se os ocurra. En cualquier caso, prometemos contestar.

11.1 ¿Y no tenéis nada que agradecer a nadie?

Ufff... Al contrario. No acabaríamos nunca. Pero vamos a intentarlo; además, es la parte más relajada de todo esto.

11.1.1 De Antonio Verdejo

Mi lista es interminable (tengo tanto que agradecer a tanta gente, y esta es *la mía* ;-), pero intentaré ser breve.

Para empezar, a Francisco José Montilla, mi *apañero*, porque fue quien me introdujo en esto de la RDSI y el Linux, gracias por tus "SOfritos", y por la paciencia que tienes conmigo y el Quake. Recuerdos a quien ya sabes. Gracias por todo, de verdad. ¡Ah!

y vigila tu espalda, un día apareceré por *DM4* con un bazoca y... ;-))

A toda la gente del Lucas, Insflug e HispaLinux. Inmejorable trabajo el vuestro. A la gente de Enred (saludos *ZoR*) por organizar lo *inorganizable* y darle forma de Party.

A Jesús Fuentes Saavedra, por sus consejos. A Enrique Melero por idéntica razón. A Iñaki Arenaza por estar trabajando también en el tema RDSI. A Miguel Armas del Rio, por mantener la (creo) mejor lista de Linux en castellano. A todos los contertulios de dicha lista por sus sugerencias y ánimo, ¡seguid así!

A Alvaro Villalva (aka *unsCAred*) por que siempre está ahí con su compilador preparado[2] y su buscador a punto... ;-)

A toda (TODA) la peña del canal linux del IRC Hispano (la lista no tendría fin, prometo -prometemos- citaros a todos en una próxima revisión de este documento, aunque sea en un anexo exclusivo ;-) por ser como son, por ser como sois, ¡majísimos!... y a las *linuxeras*, por tener ese par de... `O:-)`

A mi hermano David, y en general a toda mi familia, por su apoyo. Gracias especiales a Isa, Regi y Basi por cuidarme tan bien (¡qué haría yo sin vosotras!).

A mis amigos, mis mejores amigos (Jero, Javi, Alberto) porque siempre se puede contar con ellos, y porque comprenden que a veces pase más tiempo con Linux que con ellos...

A mis amigas, mis mejores amigas. A Begoña, por todo. A Ana Rocío, en la distancia, porque sí.

A N. S. ("no sé") por su mirada. Siempre. A Alberto (aka *Case*) por sus cumpleaños.

A Marc, por la tarjeta que dio el empujón definitivo a este Como. Y a la gente, que, junto a él ("1 para to2 y to2 para 1"), me hacen pensar *diferente*... Are U dudez?

A "el gremio del cuervo" por su música (*cojonuda*) por su directo (*destroyer*) y por darme una idea de lo larga que puede ser (y no cortarme ante ello) una lista de agradecimientos... ;-) Y a Pepe, por supuesto, por descubrirme este *pedaso* grupo. *Hello man, I am the Sun...*

A tod@s l@s que me dejo (y de l@s que sin duda tendré noticias).

Y en general, a toda la gente que hace que, cada día que me levanto, no piense como Séneca, que decía *Mañana será peor...* ;-)

¡Gracias!

11.1.2 De Francisco J. Montilla

A mi *apañero* Toni, por compartir esas madrugadas linuxeras, por tenerme al día de lo que pasa en el mundo Linux, y por dejarme masacrarle al Quake tan generosamente `:P`.

Y por dejarme sin nadie a quien agradecer. Abusooon!!!!!

A mi mujer, por aguantar estoicamente mis trasnochadas Linuxeras, mi apegamiento *ordenadoril*, y animarme todavía a darle duro a esto.

[2] FJM
Y su bazoka...

12 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`.