Ejecución de binarios de 32 bits en 64 bits

TONI CASTILLO
toni.castillo@fa.upc.edu

Presentamos en este taller *ia32.sh*, un script programado en bash que permite ejecutar los binarios más conflictivos de 32 bits, como Acrobat Reader, los plugins para Firefox de JRE, Flash, etc. en distribuciones Debian o derivadas de 64 bits de modo totalmente transparente para el usuario.

Durante algún tiempo, sobre todo al inicio de las primeras versiones de la distribución inestable de Debian para arquitecturas AMD64/EM64T, era habitual toparse con la problemática de que algunos binarios cuyas fuentes no eran públicas, simplemente no se ejecutaban. Era necesario utilizar algunas técnicas -la mayoría bien conocidas por los administradores de *GNU/Linux*- como *chroot*, *chroot* de librerías de 32 bits, máquinas virtuales, etc.

Quien firma este reportaje se decidió a desarrollar algún tipo de herramienta que pudiera ocultar completamente su funcionamiento interno, pensando siempre en usuarios finales. Elegí *chroot* y *debootstrap*, como veréis líneas más adelante, pero debía programar una herramienta nueva que las usara sin intervención directa del usuario. Así nació el proyecto *ia32.sh* que queremos presentaros aquí. Al principio era poco más que una decena de líneas escritas en C, que rápidamente descarté para desarrollarlo en bash. La elección de bash se debió a que era más sencillo de portar, sin necesidad de realizar compilaciones, y cualquier usuario con pocos conocimientos de GNU/Linux y de la shell podía entenderlo con más facilidad si decidía modificarlo a su gusto. Podríamos decir que de las doce líneas de código C del proyecto se transformaron en dos docenas de código en shell-script. A día de hoy, la última versión liberada disponible en SourceForge contiene unas cuantas líneas más, aproximadamente unas 643.

Como decíamos, queremos compartir con los lectores de **Todo Linux** nuestra experiencia con *ia32.sh* en este tutorial, vamos allá.

ANÁLISIS

El script *ia32.sh* se divide en diferentes fases de ejecución, las cuales son:

- Chequeo del entorno
- Preparación o instalación del *debootstrap*
- Enlace de los sistemas de archivo
- Generación en tiempo de ejecución de un *wrapper*
- Sincronización de cuentas de seguridad
- Ejecución del *wrapper* via *chroot*.

que vamos a describir a continuación.

CHEQUEO DEL ENTORNO

En las primeras fases del script se realizan una serie de tareas previas básicas. Entre las cuales podemos observar la comprobación de la distribución donde se lanza el script, la detección de root y la retrollamada mediante *sudo*. Gracias a dicha retrollamada, el script podrá ejecutar los pasos siguientes como superusuario (uid=0). Para garantizar que la abstracción sea total, es necesario añadir una línea como la que sigue en el archivo */etc/sudoers* (y, como es lógico, disponer del paquete *sudo* instalado):

```
usuario localhost = (root)
 NOPASSWD: /usr/loca
/bin/ia32.sh
```

Bastará substituir "usuario" por el usuario que lanzará el script, y */usr/local/bin/ia32.sh* por el PATH completo donde lo ubicaremos.

Es necesario destacar que la comprobación previa de root es básica para evitar problemas de seguridad. Nuestro script no permite la ejecución de binarios o comandos de shell de 32 bits para root. Durante un breve lapso de tiempo, y gracias a sudo, el usuario se convierte en root para preparar el entorno de ejecución del chroot, pero después el UID y el UUID del proceso ejecutado es exactamente el mismo que el del usuario que ha lanzado el script.

DEBOOTSTRAP

La técnica del *debootstrapping* es todo un clásico en *Debian*: permite instalar una minidistribución totalmente funcional en un directorio de nuestro sistema de archivos de una manera rápida a través de un repositorio de Internet.

Nuestro script usa *debootstrap* para instalar una versión Debian Etch de 32 bits en el directorio indicado, pero lo hace de modo automatizado. Por supuesto, como *debootstrap* no se instala por defecto en Debian ni Ubuntu, el script hace una comprobación trivial y, en caso de ser necesario, lo instalará vía *apt-get*.

Desde hace unas semanas, es posible encontrar en SourceForge una *FileRelease* para AMD64 en formato de paquete binario debian. En sus dependencias hemos añadido todo lo necesario para que pueda ejecutarse sin problemas, como el *debootstrap*. De este modo simplificamos todavía más su instalación o primera ejecución. De esto hablaremos un poco más adelante.

El script permite indicar mediante una variable la distro a instalar. De este modo, si cambiamos *etch* por *stable* en $DISTRO (ver **Cuadro 1**), siempre tendremos la última versión de Debian como referencia. Por eso nuestro script, actualmente, funciona únicamente con Debian o *Debian-Based*, puesto que no nos hemos preocupado excesivamente por buscar alternativas al *debootstrap* en distros basadas en paquetes RPM. En el *TO-DO* lo dejamos y se aceptan voluntarios.

Si miramos el código del script, en la función *install_ia32_environment()* observaremos que hace algunas cosas más, como la instalación de Acrobat Reader , de la JRE de SUN y de los plugins de Firefox. Al mismo tiempo, en las primeras versiones del script teníamos el problema del idioma. Necesitábamos sincronizar las locales del equipo del usuario dentro del entorno *chroot*

El script ia32.sh consigue abstraer totalmente el uso de chroot y debootstrap para usuarios sin conocimientos

para mantener la falsa sensación de que nada había cambiado en realidad. Así que optamos por añadir una nueva función *update_locale()* que arreglaba ese problema.

Si el usuario hace cambios de idioma en su equipo, *ia32.sh* no los sincronizará automáticamente. Decidimos añadir una opción explícita (`--update-locales`) para que el usuario la ejecutase bajo demanda posteriormente a la instalación inicial del *debootstrap* por motivos de eficiencia asintótica. La generación de las locales tarda un tiempo, mayor cuantas más locales tenga el usuario definidas, es decir, es un problema con notación *O* lineal: *O(n)*. Así que nos parecía terrible que se auto-regenerasen en cada ejecución del script. Consideramos que esta solución era razonablemente elegante.

Siguiendo con nuestro diseño, creímos oportuno abstraer al máximo los pasos previos que debía realizar un usuario antes de poder ejecutar sus aplicaciones de 32 bits preferidas, así que optamos por dejarle al script toda la responsabilidad.

La primera vez que lo ejecutemos indicándole una aplicación como parámetro, detectará que no disponemos todavía de un directorio de 32 bits instalado con *debootstrap* y nos lo instalará de manera automatizada, por defecto, en $CHROOTDIR (**Cuadro 1**). Podemos monitorizar todo el proceso de instalación leyendo el archivo */var/log/debootstrap.log:*

```
$ tail -F /var/log/
debootstrap.log
```

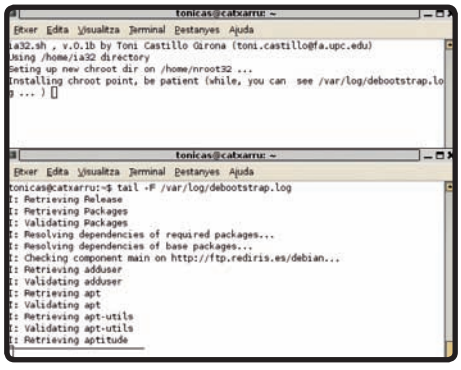

Figura 1. Instalando un entorno de 32 bits en /home/nroot32 manualmente.

Por supuesto, la idea era automatizar al máximo. Pero en cualquier momento, un usuario puede decidir instalar el entorno de 32 bits manualmente y en el directorio deseado:

```
$ ia32.sh -install
/home/nroot32
```

Tal y como puede apreciarse en la **Figura 1**.

ENLACE DE LOS SISTEMAS DE ARCHIVO

Para lograr una abstracción total, el usuario ubicado dentro de un *chroot* necesita acceder a sus archivos en ~, escuchar su música o imprimir sobre sus colas de *cups*. Para lograrlo es indispensable enlazar los diversos directorios dentro del directorio donde se ha instalado el *debootstrap*. Esto se logra gracias a *mount*. La idea es enlazar todos los directorios indicados en $MPOINTS dentro del $CHROOTDIR.

Por ejemplo, para dar acceso al ~ de cada usuario que ejecute *ia32.sh*, es necesario enlazar */home*. Para lograrlo basta con hacer algo como esto:

```
# mount -t none /home
$CHROOTDIR/home -o bind
```

ia32.sh desmontará dichos enlaces en cuanto ya no quede ningún usuario accediendo al *chroot* gracias a la llamada *check_running_environments()*. Se ha dotado de un *trap* al principio del mismo para desmontar los sistemas de archivo enlazados ante una situación de cierre con "CTRL+C" o de *kill*.

Es importante destacar que los diferentes directorios de $MPOINTS se enlazan dentro del *chroot* con permisos de lectura/escritura y siempre dependiendo de los permisos del sistema de archivos (EXT3 en nuestro análisis). Esto significa que un usuario puede trabajar en su */home* con total normalidad, pero que una ejecución de algo tipo:

```
$ rm -rf $CHROOTDIR
```

conlleva, inevitablemente, a la destrucción física de los datos de dicho directorio y de los subdirectorios que cuelgan de él, como los configurados en la variable $MPOINTS.

De momento la única solución al problema de la desinstalación del directorio de 32 bits que hemos encontrado es añadiendo un nuevo flag al script *"--unninstall"* que se encargue, de manera segura, de dicha eliminación. En la página de manual se indica con claridad.

Un caso especial era el de los dispositivos automontados vía *gnome-hal* y *udev*. En la primera versión del script no nos dimos cuenta que, si el usuario decidía pinchar una unidad externa mientras ejecutaba un chroot, simplemente no era visible desde dentro de la jaula. Un caso nada deseable. Así que tuvimos que añadir un poco más de código al mismo para solucionar este problema; de esto se encarga precisamente la función *bind_dynamic_mpoints()*.

Cuando el usuario está ejecutando una aplicación de 32 bits (por ejemplo, Acrobat Reader), y después pincha un *pendrive*, puede hacerlo visible al sistema simplemente ejecutando en otro terminal:

```
$ ia32.sh --sync-dpoints
```

Esto realiza una llamada a la función introducida líneas arriba, que se encargará de asociar ese nuevo punto de montaje dentro

Cuadro 1. Principales variables del script ▶

El script posee unas cuantas variables que definen su comportamiento. Todas ellas están perfectamente documentadas en el propio código, pero vamos a describir de manera resumida las más importantes que aparecen en este tutorial:

- **$CHROOTDIR. Directorio donde debootstrap instalará el entorno de 32 bits.**
- **$MPOINTS. Directorios que se enlazarán dentro del entorno de 32 bits, como, por ejemplo, *"/home"*.**
- **$DBS_LOG. Log de salida de la llamada a *install_ia32_environment()*, la función encargada de instalar el entorno de 32 bits sobre *$CHROOTDIR*.**
- **$CHROOT_ERROR. Log de salida de los wrappers ejecutados via *chroot*.**
- **$DEBUG. Si la ponemos a 1, obtendremos mayor nivel de información sobre su ejecución.**
- **$DISTRO. La distribución Debian que se instalará mediante *debootstrap*. Podemos indicar el nombre de la misma (como Etch). Aunque para dotar de mayor estabilidad y transparencia al script podríamos indicar *"stable"*, así siempre se nos instalará la versión más actual de Debian sin comprometer la estabilidad del sistema.**
- **$MIRRORS. Lista de servidores repositorio de Debian, separados por espacio en blanco, para que la función *install_ia32_environment()* sea redundante. Cuantos más, mejor ;-).**
- **$FILETYPES. Archivo a utilizar por *ia32.sh* para la asociación de archivos. El script se suministra con la entrada por defecto */etc/ia32f.conf*.**

del entorno de *chroot*. Cuando *ia32.sh* finaliza su ejecución, se liberan también dichos puntos de montaje para evitar problemas gracias a la función *umount_points()*.

GENERACION DEL WRAPPER

Programáticamente hablando, un *wrapper* no es más que un paso previo en cualquier ejecución de un binario, permitiendo establecer un entorno inicial para el mismo. De este modo, podemos considerar que nuestro *wrapper* se encargará de ejecutar realmente el programa deseado por el usuario sin que éste lo sepa.

Generamos el *wrapper* (mediante *shell* en *bash*), ubicándolo dentro del directorio de 32 bits instalado durante la primera fase de *ia32.sh*. Para poder ejecutar el *wrapper* vía *chroot* manteniendo el *uid* real del usuario, el script en realidad utiliza *su*. De no hacerlo así, y puesto que solo root puede lanzar un *chroot*, los binarios de 32bits se estarían ejecutando como superusuario, algo que no nos interesaba (aparte de por motivos de seguridad, debido a la abstracción y transparencia deseadas). Se puede ver un extracto de su contenido en la Figura 2.

```
#!/bin/bash
########################################
#  Wrapper32 by Toni Castillo Girona
#       see ia32.sh script for details
#       see /var/log/chroot_error_log for log
#  date:    16/02/2008 12:41
#  uid:     tonicas
#  app:     acroread ./vim.pdf
########################################
su tonicas -c "cd /home/tonicas/docs; acroread ./vim.pdf"
exit 0
```

Figura 2. Contenido del wrapper autogenerado por ia32.sh. ejecutando Acrobat Reader.

Dicho *wrapper* se almacena en *$CHROOTDIR/usr/local/bin* y se nombra siguiendo esta sencilla regla: *wrapper32_usuario.sh*. Nunca tendremos más de un *wrapper* por usuario, claro que esto no impide que se puedan ejecutar múltiples para un mismo usuario.

En la última versión liberada del script, podemos ejecutar binarios de 32 bits pero también podemos pasar como parámetro al mismo un archivo. Por ejemplo, un archivo PDF. *ia32.sh* abrirá dicho archivo utilizando el programa que hayamos configurado mediante el archivo de configuración de *filetypes*, en */etc/ia32f.conf*. Esta funcionalidad no deja de ser totalmente equivalente a la clásica asociación de archivos de cualquier entorno de escritorio.

Se pueden definir tantas asociaciones de archivos como se desee, cada una de ellas en una única línea y siguiendo el formato siguiente:

```
extension:programa|
/path/a/programa
```

Si el programa que queremos que abra el archivo se encuentra en el PATH, no es necesario indicar su ruta completa.

Veamos un sencillo ejemplo para abrir archivos flash usando el navegador y archivos PDF con Acrobat Reader:

```
pdf:acroread
swf:iceweasel
```

Como es evidente, podemos pasarle parámetros a los programas. *ia32.sh* los tendrá en cuenta, insertándolos de manera íntegra justo antes del nombre del archivo introducido como parámetro del mismo:

```
bin:xterm -bg black
-fg white -exec
```

En la línea anterior podríamos ejecutar archivos .bin (como la instalación de GoogleEarth), directamente mediante el *xterm* configurado con fondo negro y texto en blanco.

Cuando un usuario desee abrir un archivo PDF, tan solo debería ejecutar en un terminal:

```
$ ia32.sh /docs/my_pdf.pdf&
```

Se pueden añadir, modificar o borrar asociaciones de archivos simplemente editando el archivo */etc/ia32f.-conf* con cualquier editor ASCII, como *vim*. De todos modos, nos pareció buena idea añadir un nuevo *flag* al script que listara dichas asociaciones:

```
$ ia32.sh -list-filetypes
```

SINCRONIZACIÓN DE LAS CUENTAS DE SEGURIDAD

La sincronización de las cuentas es indispensable puesto que es muy probable que se hagan cambios en la base de datos de usuarios local, o incluso en los archivos de configuración del host. El script únicamente necesita hacer una copia trivial de los archivos más importantes:

```
/etc/shadow,
/etc/passwd,
/etc/group,
/etc/hosts,
/etc/resolv.conf
```

justo sobre *$CHROOTDIR/etc*.

Gracias a este paso, los usuarios no solo existirán dentro del entorno de 32 bits, sino que tendrán sus *passwords* sincronizados. Al mismo tiempo es importante que el archivo de resolución de nombres también se actualice, para evitar problemas de DNS dentro del entorno de 32 bits, algo que sucedía en las primeras versiones del script cuando, por ejemplo, pasábamos de conexión *ethernet* a una *wireless* vía DHCP.

EJECUCIÓN DEL WRAPPER

El último paso destacable del script es la ejecución usando *chroot* del *wraper* generado en el punto anterior. La verdad, no puede ser más simple. Cuando se llega a este paso, lo que *ia32.sh* hace en realidad es algo como:

```
# chroot $CHROOTDIR
/usr/local/bin/
wrapper32_usuario.sh
```

Añade algunas funcionalidades extras, como un log de errores sobre el archivo */var/log/chroot_error.log*. Se registra la fecha y hora de ejecución de cada comando, quién lo ha lanzado, qué ha lanzado y la salida de *stderr* de dicho binario.

EL SCRIPT EN FUNCIONAMIENTO

Podemos usar *ia32.sh* en la shell o mediante iconos lanzadera en GNOME. Lo más interesante es, precisamente, crear iconos en GNOME porque esto permite simplificar su uso al máximo, integrándose totalmente con la distribución de 64 bits (¡y con el escritorio!). En ningún momento se aprecia lo que hay por debajo.

En SourceForge es posible ver una lista de vídeos demostrativos de configuración de una lanzadera en GNOME, de asociación de archivos mediante Nautilus, etc. Como es evidente, esto debe ser fácilmente aplicable a otros escritorios como KDE o WindowMaker.

IA32.SH EN LA SHELL

Tras el análisis del script , vamos a ver las diferentes posibilidades del mismo en la shell:

- Abrir un documento PDF con Acrobat Reader:

```
$ ia32.sh acroread
docs/my_doc.pdf&
```

Es importante destacar que *ia32.sh* almacena el directorio de trabajo del usuario (*cwd* en terminología POSIX), por lo que se puede ejecutar con total normalidad, como si estuviésemos ejecutando un Acroread directamente.

Tal y como hemos comentado algunas líneas atrás, también es factible abrir dicho archivo mediante:

```
$ ia32.sh docs/my_doc.pdf&
```

- Ejecutar una x-terminal de 32 bits:

```
$ ia32.sh xterm -bg
black -fg white &
```

La ejecución de una shell *xterm* de 32 bits permitirá al usuario lanzar aplicaciones no gráficas en el *chroot*. Hay que tener en cuenta que todo este desarrollo comenzó pensando en ejecutar aplicaciones X11. Nos pareció trivial añadir un *xterm* a la lista de paquetes del *debootstrap* para permitir ejecutar binarios de consola.

- Chequear el directorio de 32 bits y, en caso de error, instalarlo de nuevo en otro directorio:

```
$ ia32.sh -check ||
 ia32.sh -install
/home/new_root32
```

- Actualizar las locales del sistema sobre la distro de 32 bits:

```
$ ia32.sh --update-locales
```

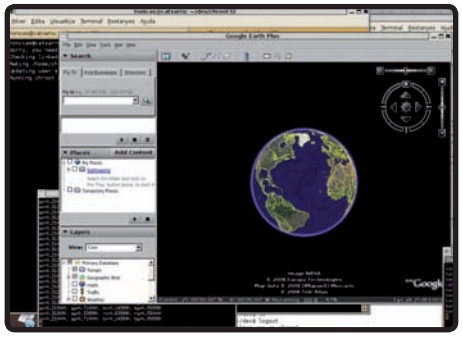

Figura 3. GoogleEarth ejecutándose en un Debian Etch de 64 bits.

En la Figura 3 se puede observar al maravilloso *GoogleEarth* de 32 bits ejecutándose dentro del *chroot* gracias a *ia32.sh*, y en la Figura 4 un ejemplo práctico del porqué de lanzar un *xterm* de 32 bits... ¡para poderlo instalar!

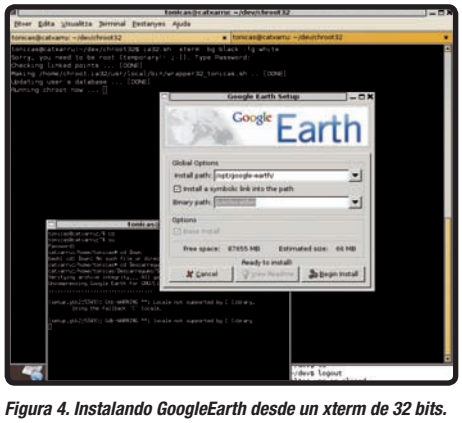

Figura 4. Instalando GoogleEarth desde un xterm de 32 bits.

ABSTRACCIÓN TOTAL: IA32.SH EN GNOME

Crear un icono en el escritorio y dejarlo ahí para que el usuario simplemente lo pulse, era en realidad, la finalidad total y absoluta de nuestro humilde desarrollo. Y para lograrlo, seguimos estos pocos pasos:

PASO 1

Añadimos una nueva lanzadera en el escritorio de la manera habitual.

PASO 2

En el widget *GtkEntry* etiquetado como "comando", escribimos:

```
ia32.sh X11program
```

donde *X11program* es el binario de 32 bits a ejecutar (vemos un ejemplo con Acrobat Reader en la Figura 5)

Figura 5. Creación de una lanzadera con ia32.sh en GNOME.

PASO 3

Podemos usar el icono instalado (de existir) ubicado en *$CHROOTDIR/path/al/directorio/de/recursos*.

PASO 4

Y nada más; el nuevo icono de lanzadera aparecerá en el escritorio de GNOME y podremos ejecutarlo de la manera habitual: ¡con un simple clic de ratón!

INSTALACIÓN DEL SCRIPT EN UN SISTEMA DEBIAN (O DERIVADO)

Tal y como hemos comentado casi al principio de este reportaje, disponemos de un paquete debian para arquitectura amd64 disponible en SourceForge (ver cuadro Referencias). Cualquier usuario puede descargarlo e instalarlo ejecutando:

```
# dpkg -i ia32.sh_0.1-amd64.deb
```

Podemos ver fácilmente qué archivos ha copiado y en qué ruta:

```
# dpkg -L ia32sh
/.
/etc
/etc/ia32f.conf
/usr
/usr/share
/usr/share/man
/usr/share/man/man1
/usr/share/man/man1/ia32.sh.1
/usr/share/man/man5
/usr/share/man/man5/ia32f.conf.5
/usr/local
/usr/local/bin
/usr/local/bin/ia32.sh
```

Si el usuario desea realizar la instalación a mano, basta con que se descargue estos archivos vía CVS (*project name script* y *man*) y los copie a mano en los directorios correspondientes. Puede guiarse mediante el listado anterior.

Nuestro script contiene dos páginas de manual, una para el propio comando *ia32.sh* y otro para el formato de archivo *ia32f.conf*.

Lo único que el paquete debian no hace todavía es automatizar las entradas en el archivo */etc/sudoers*, por lo que el usuario deberá configurarlo a mano elija la opción de instalación que elija. La verdad, en su momento no le dimos excesiva importancia a este trivial problema, porque el script... ¡siempre lo instalamos nosotros!

Recomendamos al lector que siempre utilice el CVS porque allí siempre encontrará las últimas versiones del script, de la página de manual, etc.

CONCLUSIONES

Hemos desarrollado una herramienta simple pero que puede aprovecharse para funcionalidades adicionales a las pensadas. Creemos que puede ser de mucha utilidad para usuarios de GNU/Linux que necesiten ejecutar binarios de 32 bits nativos que no funcionen con las librerías *ia32libs* o que, por motivos de diseño, no puedan recompilarse bajo arquitectura de 64 bits.

En el *TO-DO* nos quedan todavía algunas cuestiones por añadir y mejorar, como una versión gráfica: algo parecido a una jaula con nuestros binarios de 32 bits colocados dentro de la misma. Estamos pensando en una implementación completa de 32 bits con su propio escritorio (menos pesado que GNOME, KDE, etc.), sin necesidad de que dichos binarios estén físicamente instalados dentro del *chroot*. De acuerdo, el concepto sería radicalmente opuesto a la idea original de abstracción e integración con el host de 64 bits, pero no hay duda de que pueden coexistir las dos ideas. Con esto ofreceríamos un entorno completo de ejecución de 32 bits, en su jaula, pero con mejores prestaciones que una máquina virtual. Hemos de analizar con calma las posibilidades.

Para finalizar, pedir a todos aquellos de vosotros que estéis interesados en nuestro proyecto, que colaboréis con él. Ni falta hace decir que todo es código GPL, y que podéis copiarlo, modificarlo, o quemarlo.■

Referencias ▶

Todo nuestro proyecto se encuentra en SourceForge. Podéis obtener sus fuentes, las páginas de manual, el paquete para Debian Etch AMD64/EM64T, algunos vídeos algo "freaks" de su funcionamiento, y algunas cosas más. A continuación listamos los enlaces web más importantes. Sinceramente que esperamos las visitas de los lectores de **Todo Linux**:

- La web principal del proyecto en SourceForge.
  http://sf.net/projects/ia32sh
- Nuestro código CVS de acceso público como read only. Si alguien está interesado en formar parte del proyecto, ¡enviadnos un email!
  http://ia32sh.cvs.sourceforge.net/ia32sh/
- La web "oficial" del proyecto, hospedada en sf.net. De momento tenemos poca cosa, así que esperamos vuestras contribuciones.
  http://ia32sh.sourceforge.net