



EL CONSULTORIO DE KLAUS



Klaus Knopper es el creador de Knoppix y co-fundador de la LinuxTag Expo. En la actualidad trabaja como profesor, programador y consultor. Si tiene algún problema de configuración, o simplemente quiere conocer mejor cómo funciona Linux, no dude en escribir sus preguntas a: klaus@linux-magazine.com

Enlaces Borrados al Reiniciar

? Diversos programas multimedia que uso se basan en la existencia de “dispositivos” como `/dev/dvd` o `/dev/cdrom`. Hace un tiempo generé un enlace suave como root tal que `[user # ln -s /dev/hdc /dev/hdd; ln -s /dev/hdd /dev/cdrom]`.

Ahora, cada vez que reinicio, mi sistema Suse 10.1 borra mis enlaces, y tengo que acordarme de configurarlos de nuevo

(o si no el software “me lo recuerda” negándose a funcionar). Como solución momentánea, he escrito un pequeño script que se ejecuta automáticamente al reiniciar, pero dudo que sea la mejor solución. En resumen, mis preguntas son:

- ¿Por qué se borran mis enlaces al reiniciar?
- ¿Existe una manera mejor de solventar el problema?



Conozco bien lo que me comenta, y coincido en que es un problema realmente molesto.

La razón por la que se borran, renombran o se pierden mágicamente los archivos de dispositivo es que muchas distribuciones conmutan de archivos de dispositivo “estáticos” a dinámicos durante la creación y borrado de estos archivos, en función de los drivers presentes.

En los viejos tiempos, `/dev` era simplemente un directorio que contenía archivos de dispositivo (dispositivos *char*, que mostraban una “c” en los permisos del archivo al teclear el comando `ls -l`, y los dispositivos de bloque, que mostraban una “b”). Si intentábamos acceder a un dispositivo que no tenía un módulo del kernel cargado, para controlarlo, el kernel automáticamente cargaba el driver con `modprobe`.

La filosofía de Unix ha sido siempre “todo es un fichero” (bueno, excepto las tarjeta de red). Esta filosofía aún existe, pero la base técnica cambió con la aparición de udev.

Con udev, `/dev` es una partición de ramdisk. Se crean todos los archivos en el momento de iniciar por el demonio de udev, `udev`, y se desechan cuando se apaga el sistema. Esto explica los cambios en su sistema, como al crear un dispositi-

tivo de nodo para discos floppy con capacidad de 1.7MB con:

```
mknod /dev/fd0u1722 b 2 60
```

simplemente “desaparecerá” al apagarse, y también a veces al usar la funcionalidad de suspensión a disco tras el despertar desde disco.

udev crea dispositivos en `/dev` cuando se carga un módulo del kernel que usa una llamada al sistema para registrar un dispositivo *block* o *char* con un nombre específico. Por tanto, la secuencia es: cargar el módulo del kernel y conseguir un

Listado 1: Archivo de dispositivo de Unidad Floppy Sobreformateada

```
01 # This file does not exist.
    Please do not ask the debian
    maintainer about it.
02 # You may use it to do strange
    and wonderful things, at your
    risk.
03 ...
04 L core /proc/kcore
05 L sndstat
    /proc/asound/oss/sndstat
06 ...
07 # Hic sunt leones.
08 M ppp c 108 0
09 D loop
10 M loop/0 b 7 0
11
12 # My own rules, create a cdrom
    symlink to hdc,
13 # and a floppy with extended
    capacity.
14 L cdrom hdc
15 M fd0u1722 b 2 60
```

archivo de dispositivo con un nombre específico.

Los módulos del kernel a cargar en el arranque se listan, en la mayoría de las distribuciones, en `/etc/modules`. Si sólo necesitamos un dispositivo que funcione con el nombre por defecto, sólo será necesario añadir el módulo coincidente a `/etc/modules`. Pero a veces, también nos gustaría tener los enlaces simbólicos a los dispositivos de “funcionalidades adicionales” que utilizábamos antes de udev.

En lugar de escribir un script init que cree esos archivos de manera manual, podrías usar una característica más o menos “no oficial”. El archivo `/etc/udev/links.conf` (como el de Debian GNU/Linux) contiene líneas que se interpretan al iniciarse udev con su propio script init.

En el Listado 1, el comando `L cdrom hdc` provoca un `ln -s hdc` en el directorio `/dev`, mientras que `M fd0u1722 b 2 60` genera un `mknod /dev/fd0u1722 b 2 60`, creando de esta manera el archivo de dispositivo de la “unidad floppy sobreformateada”.

Nótense los simpáticos comentarios acerca de la inexistencia de este fichero, así como la referencia a los leones, obra del mantenedor del paquete udev. Por supuesto, el archivo existe, y se interpreta al iniciarse udev por su propio script init `/etc/init.d/udev`. El archivo `links.conf` puede no estar presente en todas las distribuciones (aunque la mayoría tienen un mecanismo similar para añadir nuestras propias cosas), y quizá esto esté incluso sujeto a cambio en futuros lanzamientos en Debian, pero de momento, ha demostrado ser muy útil.

Si tenemos una distribución que no implementa un fichero similar, podemos usar el propio mecanismo de udev para crear enlaces simbólicos y dispositivos. Debemos verificar `/etc/udev/cd-aliases.rules` (si tenemos una versión más antigua de udev) o `/etc/udev/cd-aliases-generator.rules` (para versiones de udev ≥ 0.098). Puede que tengamos que familiarizarnos con la sintaxis de configuración de udev (lo cual no es demasiado complicado) para crear los enlaces simbólicos deseados, pero merece dedicarle un rato.


Cuando se recurra a la suspensión a disco y algunos enlaces simbólicos y dispositivos desaparezcan tras reanudar, ha de llamarse a `*udevsynthesize` (para las versiones antiguas de udev), o a `*udev-`

`trigger` (para las versiones actuales de udev)

desde dentro del script de suspensión/reanudación. Esto provoca que udev vuelva a escanear los dispositivos y ejecute los scripts `/etc/udev/rules.d/*`, como lo haría en un arranque normal.

Puede que se esté preguntando por qué udev ha de estar ubicado en un ramdisk tmpfs, volviendo a crear todos los archivos de dispositivo en cada reinicio. Usar un directorio fijo en el disco duro parece que facilitaría la labor de nombrar dispositivos. Pero entonces, udev trataría de llevar el control de los dispositivos internamente. Podría confundirse si los archivos de dispositivo o los enlaces simbólicos ya están allí y duplicaría los archivos con nombres diferentes. No he experimentado mucho con esto aún.

Más Fedora 5

 Cuando cambié de un AMD de 32-bit a un AMD de 64-bit, me era imposible acceder a los archivos que contienen mi información personal. El Knoppix 5.0 ha sido un salvavidas para mí, pues me permitió copiar estos archivos y así rescatarlos. Me gustaría instalarlo (al igual que otras distros) pero de momento no he averiguado cómo.

He instalado Fedora Core 5, y funciona bien en general, aunque se mantienen algunos pequeños fallos. Parece que es imposible utilizar kppp para activar un modem sin facilitar la contraseña de root. Apparentemente todo se ha configurado para que sea accesible por el usuario (`/usr/bin/... /usr/sbin/...consolehelper`), pero nada evita la necesidad de la contraseña de root. Además de esto, tampoco puedo cambiar el icono de kppp.

Es una máquina de un único usuario, a pesar de lo cual no encuentro la manera de evitar tener que loguearme con una contraseña.

Por último, existen componentes de emacs instalados en el sistema, pero no he podido ejecutarlo hasta el momento.

Quizá pueda comentar algo acerca de alguno de estos problemas.

(nota del editor: El lector escribió de nuevo con la siguiente actualización).

He encontrado una solución al primer problema en las páginas de soporte de Fedora: sólo he tenido que cambiar el

enlace `/usr/bin/kppp` a `consolehelper`. El resto de problemas persiste.



Sus comentarios me hacen recordar que Knoppix 5.1 debería haberse lanzado ya, pero aún existen algunas cosas, relacionadas con el Kernel 2.6.19-rc*, que deben arreglarse antes. En el momento de escribir estas líneas, posiblemente se encuentre disponible.

En cuanto a los procesadores de 64 y 32 bits: la mayoría de los de 64 bits (¿todos?) tienen un modo de compatibilidad que permite ejecutar programas escritos para procesadores de 32 bits de su mismo tipo. Pero el código binario (el juego de instrucciones de código máquina) es diferente, lo que significa que podemos ejecutar programas para 64 bits y 32 bits en paralelo, pero no podemos mezclar código para 32 bits con código para 64 bits. Y he aquí el problema: cuando estamos usando un programa con código de 64 bits, las librerías de soporte deben ser todas de 64 bits también. Si la “biticidad” no es consistente, podemos experimentar todo tipo de graciosas eventualidades, desde partes del programa que no funcionan (de ahí el fallo para cargar ficheros que de otra manera se pueden leer sin problemas) hasta cuelgues súbitos con errores del tipo “illegal instruction”.

Como Knoppix (la rama principal) usa código de 32 bits básico en todos los programas, funcionará correctamente con procesadores de 64 bits compatibles con i86. No tan rápido como en entornos 64 bits nativos, pero no apreciablemente más lento, a menos que nos dediquemos a tareas con muchos cálculos. Esto explica por qué era posible acceder a sus archivos personales.

Así que, si ejecuta un sistema operativo de 64 bits, asegúrese de que todas las aplicaciones se ejecutan en 64 bits, y que no haya librerías, módulos o plugins que usen código en 32 bits. Por ejemplo, un OpenOffice compilado en 32 bits arrancará en modo 64 bits, pero tan pronto como comience a usar plugins y librerías del entorno en 64 bits, empezarán a pasar cosas raras.

Si han de ejecutarse aplicaciones de 32 bits, sería recomendable usar un chroot dentro de la instalación GNU/Linux de 32 bits, con lo que estas aplicaciones sólo ejecutarían librerías de 32 bits. A pesar de que yo no lo recomiendo, si sigue queriendo instalar un CD o DVD Knoppix de 32 bits como SO Debian en un hardware de 64 bits, puede usar el instalador de Knoppix. (Por

cierto, estamos trabajando en un nuevo instalador). Pero probablemente lo mejor y más eficiente para su moderno hardware es acudir a una instalación Debian nativa de 64 bits y migrar toda la información en 32 bits al nuevo sistema. Probablemente queramos usar algún que otro programa en 64 bits algún día, y tarde o temprano nos íbamos a topar con los mismos problemas que ya ha sufrido al usar código de 32 bits en entornos de 64 bits.

Respecto a kppp (y algunos otros programas de KDE) que preguntan siempre por la contraseña de root: algunos programas de KDE (y también de GNOME) se han diseñado para que no se ejecuten con el atributo de archivo set-userid. Por tanto, tienen que arrancarse con su o sudo. Knoppix usa sudo para conmutar al usuario root antes de iniciar kppp. Si queremos hacer esto sin necesidad de contraseña, podemos añadir esta entrada en */etc/sudoers*:

```
your_login ALL=NOPASSWD: ALL
```

Debe señalarse que esta configuración implica toda clase de problemas de seguridad relacionadas con la ejecución de scripts maliciosos, a la que nuestro navegador o cliente de correo puede ser conducido sibilinamente, y arrancar de esta manera como root. Estos scripts serán un gran peligro, mientras que el daño causado si hubiesen arrancado con nuestra id de usuario sería como mucho la destrucción de los archivos de usuario. Pero dicho esto, también hay que señalar que no deja de ser un riesgo tener que teclear continuamente la contraseña de root, que puede acabar un día en una ventana de chat vía copiar y pegar...

Cuando habilitamos el sudo sin contraseña, podemos cambiar la parte de "comando" en la configuración del .desktop de un programa a *sudo [-H] command [options]* para iniciarlo como root, en lugar de usar *kdesu* o *gksu*.

Pero incluso con los *kdesu* o *gksu* gráficos, tenemos una opción (seleccionable) para "recordar la contraseña", evitando tener que teclearla de nuevo en el siguiente intento. Puede que haya pulsado accidentalmente esta opción y sea esta la razón por la que kppp no pide la contraseña en sus pruebas. Personalmente no estoy del todo convencido de que esta opción de "recuerdo" sea más segura que el método de sudo. Nunca es recomendable guardar las contraseñas en disco, incluso con (a veces demasiado débil) cifrado.

Para un login sin contraseña, podemos usar un método parecido al de Knoppix con

```
su -c xinit -  
your_login
```

en lugar de iniciar *xdm*, *gdm* o *kdm* como administrador de visualización y sesión dentro de un script *init*.

Pero *kdm* permite también autologuearnos con un usuario dedicado. Usamos el panel de control de KDE para configurar esta opción en *System Management | Login Manager* (Figura 1).

Respecto a la pregunta de emacs: puede que tengamos varias versiones instaladas. Quizá incluso *xemacs* y *emacs* en paralelo (lo que es perfectamente posible a pesar de que la ubicación y denominación de los archivos de configuración pueden llevar a confusión). En lugar de averiguar cuántos elementos de emacs están realmente instalados, la solución más sencilla es eliminar completamente todo lo relacionado con emacs con el comando *rpm -qa | grep emacs* y reinstalar los RPMs de emacs (más plugins) con la misma versión.

Disco SATA

? Tengo un PC con un disco duro SATA. He instalado Linux con anterioridad, pero sólo en ordenadores con discos duros ATA. Cuando trato de instalar Linux en un ordenador con un disco duro SATA, me aparece el mensaje "No se han encontrado discos duros" en el proceso de instalación.

No sé qué hacer. He escrito muchas preguntas en Internet, pero hasta el momento no he recibido sugerencias de utilidad. ¿Podría usted ayudarme?

💡 Si investiga en un buscador con la pregunta "¿Funciona mi controlador SATA bajo Linux?", probablemente hallará muchas respuestas aconsejándole acerca de qué distribución de Linux usar. Pero lo que realmente quiere usted saber es si su controlador SATA se reconoce y funciona. En caso afirmativo, podemos elegir la distribución que nos guste y conseguir el soporte necesario para el controlador.

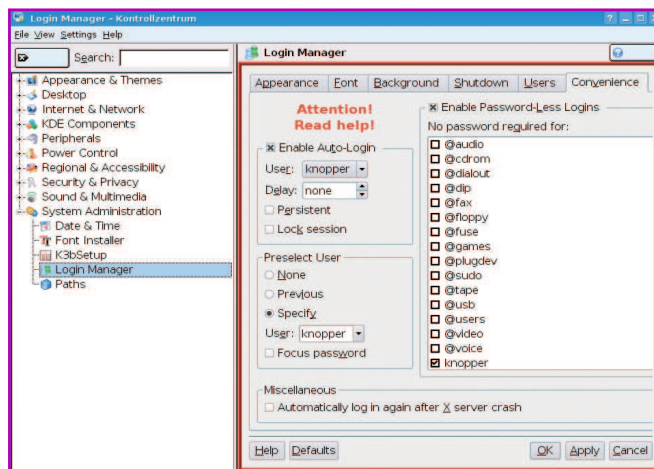


Figura 1: Configuración de login sin contraseña para KDM.

El lugar indicado para buscar el hardware soportado es la lista de dispositivos de hardware soportado por el kernel de Linux. La mayoría de los controladores SATA están soportados convenientemente por las series 2.6.x. Eche un vistazo para averiguar si la suya está en la lista: <http://linux-ata.org/driver-status.html>

Sin embargo, en función de la distribución específica de GNU/Linux, el último kernel puede que no esté disponible en el proceso de instalación. Por tanto, es posible que aparezca un mensaje de error como el que ha descrito, relativo a la ausencia de discos duros.

Si ocurre esto, aún tenemos algunas opciones. En lugar de intentar modificar el CD o DVD de instalación nosotros mismos añadiendo un nuevo kernel, podríamos arrancar una distribución diferente con soporte para su controlador en modo rescate, hacer un chroot hasta el medio de instalación montado, y ejecutar el instalador dentro de un entorno chroot. Aún así este método de instalación es algo avanzado, ya que habrá que configurar e instalar un kernel diferente tras la instalación principal que sea acorde con el hardware.

Otra posibilidad sería usar una configuración IDE a la vieja usanza para la instalación, para luego actualizar el kernel y copiar la instalación entera al disco SATA, y por último cambiar los nombres de dispositivo en */etc/fstab* de manera acorde.

Algunas BIOS permiten habilitar la emulación de "antigua IDE" en los discos SATA, de forma que parezcan discos IDE "normales". Esta emulación IDE no es una opción común, pero puede que queramos probarla. ■