



# EL KONSULTORIO 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](mailto:klaus@linux-magazine.com)

## Detalles USB

**?** ¿Podría ayudarme a entender los detalles de la conexión de dispositivos USB? Le pido también si me podría indicar cómo conectar un disco duro externo Seagate USB. Mi equipo tiene dos puertos en el frontal y varios en la parte trasera. ¿Cómo puedo saber si es seguro usar los puertos del frontal? ¿Da igual conectarlo a cualquier puerto USB aleato-

riamente? ¿Si conecto mi impresora USB a un puerto en concreto, debería conectar la impresora siempre al mismo puerto?. Puedo usar los puertos tanto del frontal como de la parte trasera para conectar con éxito un dispositivo flash USB. Estoy usando actualmente openSUSE 10.2. Sin embargo, no consigo conectar correctamente el disco duro externo Seagate USB. Lo conecto a algún puerto USB de la parte trasera, pero no pasa nada.

Tras seguir los consejos encontrados en una búsqueda Web, hice lo siguiente.

YaST detectó el disco duro y lo denominó `/dev/sdb`. Como root, añadí esta línea a `/etc/fstab`:

```
/dev/sdb /media/seagate ↵
usbfs auto,hotplug,??
defaults 1 2
```

Creé el directorio `/media/seagate` en `/etc/udev/rules.d` y creé `rule 98-mount.rules` con las siguientes líneas:

```
# run mount -a everytime a ↵
block device is added/removed
SUBSYSTEM=="block", ↵
run+="/bin/mount -a"
```

Desafortunadamente esto no ha funcionado. ¿He cometido algún error? ¿Podría ayudarme a conectar este disco duro USB?

**!** Los detalles de la conexión de dispositivos USB están en su mayor parte ocultos y son por tanto algo misteriosos. Generalmente funciona así:

1. Se conecta el dispositivo.
2. La conexión en caliente `/udev` detecta que es un dispositivo USB y carga el módulo del kernel `usb-storage`.

3. El módulo vuelca todas las particiones detectadas en el kernel, y aparecen en `/proc/partitions`.

4. Para `udev`, los dispositivos en `/dev/sd*` se crean para cada partición del dispositivo, y se ejecutan los scripts de `/etc/udev`.

Los pasos 1 a 3 deberían causar la aparición de algún mensaje del sistema en `dmesg`. También, tras algunos segundos necesarios para que se establezca el subsistema USB, las particiones contenidas en el disco duro deberían aparecer en `/proc/partitions`. Para verlas, tecleamos:

```
cat /proc/partitions
```

Ahora, si es un disco nuevo, es posible que no esté particionado aún, a menos que sea una unidad flash, que suele tener una única partición FAT16 ó FAT32. Aunque podemos usar una unidad no particionada con Linux, es más seguro particionarla, ya que el kernel detecta de esta manera la tabla de particiones correctamente. Para ello podemos usar `fdisk` o `cdisk` `/dev/sdb` (en caso de que nuestro disco aparezca como `sdb` en `/proc/partitions`) y formatear así la primera partición como un sistema de archivos Linux con `mkreiserfs /dev/sdb1` o como FAT32 con `mkdosfs -F32 /dev/sdb1`.

Si colocamos un script para `udev`, como nuestro `98-mount.rules`, en `/etc/udev/rules.d`, se ejecutará para este dispositivo si el archivo se puede leer y ejecutar. Nos aseguramos que sea el caso con:

```
chmod 775 /etc/udev/rules.d/↵
98-mount.rules
```

Sin embargo, yo no usaría *mount -a*, ya que monta todo lo que sea automontable en */etc/fstab*. Este script udev monta sólo las particiones detectadas:

```
SUBSYSTEM=="block", ⚡
ACTION=="add", ⚡
RUN+="/bin/mount $DEVNAME"
SUBSYSTEM=="block", ⚡
ACTION=="remove", ⚡
RUN+="/bin/umount -l $DEVNAME"
```

Este script intenta montar cada nueva partición detectada con las opciones y los directorios de destino listados en */etc/fstab*, y desmontará el disco con *umount* al extraerlo (aunque usar *umount* antes de desconectar el dispositivo es altamente recomendable para guardar toda la información aún no escrita en el disco).

La entrada que se añadió a */etc/fstab* coincidiría con un disco formateado, no particionado, pero el *usbfs* como sistema de archivos tiene pinta de ser un error, ya que *usbfs* es un sistema de archivos virtual para describir dispositivos USB en general, no es un sistema de archivos útil para guardar información.

Si se ha particionado el disco duro USB con una partición primaria como un sistema de archivos FAT32, podríamos usar una entrada */etc/fstab* tal que:

```
/dev/sdb1 /media/sdb1 ⚡
vfat noauto,users,umask=⚡
000 0 0
```

en su lugar, sin olvidar hacer *mkdir* para crear el directorio */media/sdb1*.

Las banderas de montaje *users,umask=000* aseguran que podamos montar y desmontar el dispositivo como un usuario normal, no sólo como root, y que podamos acceder a todos los archivos en modo lectura y escritura.

El montaje y desmontaje manual con *mount/umount* también funciona sin el hotplug o udev. Simplemente usamos la entrada */etc/fstab*, siempre que el módulo *usbstorage* esté presente.

```
sudo modprobe usb-storage
# wait some time...
mount /media/sdb1
```

Ahora la partición debería poder montarse y utilizarse.

Para particiones grandes que se están usando fundamentalmente bajo Linux, debería usarse probablemente ReiserFS o Ext3 en lugar de FAT32 (vfat), debido a que este último sólo puede guardar archivos de hasta 4GB. Si queremos usar el disco entero en lugar de crear particiones, como en el ejemplo mencionado, el comando sería:

```
mkreiserfs -f /dev/sdb
```

y la entrada en */etc/fstab* tendría un aspecto como el siguiente:


```
/dev/sdb /media/sdb ⚡
reiserfs noauto,users 0 0
```

Tras el montaje del disco entero como */media/sdb* por primera vez, hacemos un:

```
chown USER /media/sdb
```

como root, donde *USER* es nuestra cuenta normal de usuario, de manera que no tengamos que ser root para acceder al disco.

## Webcams

 Me encanta su sección "El Consultorio de Klaus", porque creo que da mucha y buena información que es fácil de entender. Tengo dos modelos diferentes de unas viejas webcams Logitech Quickcam. La primera es una QuickCam Express, que es peor que la otra cámara, pero que funciona bien bajo Windows.

La segunda cámara es una Quickcam USB:

- Driver: v.0.6.6
- Vendor: 045D
- Product: 0870
- Sensor: HDCS-1000/ 1100

(No tengo el CD de instalación de Windows para esta segunda cámara).

Finalmente pude hacer funcionar esta segunda cámara bajo Linux, pero tras una actualización automática del kernel (uso Debian Etch), la cámara dejó de funcionar. He tratado de instalar el driver adecuado, pero sin éxito. Obtengo un error de compilación que me dice que *gcc not compatible*.

Ahora quiero cambiarme a otra marca de webcams que funcione, pero no sé cuál elegir.

Quiero una con un driver estable para Linux que sea fácil de instalar. Todos

estos años en que he estado usando Debian Sarge, he tenido muchos problemas con estas webcams.

Mi cámara digital funciona a la perfección cuando la conecto al puerto USB.

¿Puede darme alguna pista?



La mayoría de las cámaras digitales (las que hacen fotos) tienen un "modo disco duro" que funciona con el módulo del kernel *usb-storage*, por lo que podemos simplemente montar la memoria flash de la cámara y leer o escribir las fotografías. Esta es la razón por la que la mayoría de las cámaras digitales funcionan bajo Linux sin necesidad de drivers adicionales.

Las webcams, por contra, requieren un módulo de vídeo que capture los fotogramas del vídeo y el sonido desde la cámara a una tasa de fotogramas configurable. Para esto se necesitan módulos del kernel específicos para cada chipset. Desafortunadamente, no pasa como con las tarjetas gráficas que tienen un estándar svga: existen tantos drivers como webcams porque los fabricantes no son capaces de acordar un estándar.

El módulo *gspca* (Generic Software Package for Camera Adapters) y sus herramientas de soporte, creadas y mantenidas por Michel Xhaard, sirve de driver para muchas webcams diferentes (unas 240).

A partir de la lista de dispositivos soportados, existen buenas posibilidades de que al menos la primera de las dos que mencionas, la Quickcam Express, funcione. Pero ojo, que a veces los fabricantes cambian el chipset sin cambiar el nombre del producto.

El módulo *gspca* no está incluido en el kernel estándar, y probablemente ni siquiera esté presente en la mayoría de las distribuciones, a pesar de que tiene licencia GPL y por tanto es libremente distribuible. Por tanto, tendrá que obtener el código fuente actualizado [1] y compilarlo usando el código del kernel que estemos utilizando. Comenzamos con:

```
wget -c http://mxhaard.free.fr⚡
/spca50x/Download/gspcav1⚡
-20070508.tar.gz
```

ahora descomprimos:



```
tar xzf gspcav1-20070508.tar.gz
```

nos situamos en el directorio y completamos la instalación:

```
cd gspcav1-20070508
make
sudo make install
```

El proceso de compilación determina la localización del código fuente a partir del enlace simbólico situado en `/lib/modules/uname -a/build`. Si no es la ubicación correcta, use:

```
make KERNELDIR=?
/path/to/kernel/source
```

en su lugar. Así mismo, tendremos que usar exactamente el mismo compilador C que se usó para compilar el kernel. Para esto podemos añadir la variable `CC=/path/to/gcc-version`. Si falta el compilador C coincidente, no funcionará a menos que también recompilemos e instalemos el kernel con ese compilador C diferente.

Si el sistema ejecuta hotplug, el módulo `gspca` se cargará automáticamente al conectar la webcam. Al menos, este es el caso para cámaras que usan un chipset soportado. Puede comprobar la lista de dispositivos compatibles en [2].

Para cargar el módulo de manera manual tecleamos (como root):

```
modprobe gspca
```

y buscamos en la salida de `dmesg` para ver si ha ocurrido algo que indique que se ha detectado la cámara.

Para obtener una fotografía podemos usar `gqcam`, una herramienta de videoconferencia al igual que Ekiga, o podemos hacer:

```
mplayer tv:// -tv ?
driver=v4l:device=/dev/video0
```

Si tenemos varias tarjetas de vídeo, cambiamos `/dev/video0` al dispositivo adecuado, y añadimos `:width=352:height=288`: a las opciones del driver para cambiar el tamaño de la captura.

## Módems



¡Necesito ayuda! He instalado Mandriva 2007 y posteriormente

Mandriva 2007.1 (Spring), y no puedo configurar mi módem. He buscado por todas partes la manera de ejecutar `slamr`, pero ¡es imposible! El módem funcionaba perfectamente con Mandriva 2006.



Esto va a depender en gran medida de qué tipo de módem esté usando. Usualmente, un módem real consiste en una parte controladora y una bomba de datos, y responde a sus propios comandos de módem. Ahora la industria ha introducido los denominados winmódems, generalmente en las placas base, aunque también como tarjetas. Estos winmódems no funcionan con el driver serie a secas. La razón de ello es probablemente el coste. (Los winmódems están generalmente integrados en la placa base, a veces incluso como parte de la tarjeta de sonido).

Un winmódem necesita un driver o software que sepa cómo enviar y recibir los datos desde el chipset. Por diversos motivos, muchos fabricantes de hardware parece que han “olvidado” proporcionar especificaciones abiertas sobre cómo escribir drivers para estos dispositivos, por lo que, o nos quedamos con un componente hardware que no funciona, o compramos un módem real, o tendremos que intentar que esta “caja negra” haga lo mismo que haría un módem real. Algunos winmódems de los fabricantes más considerados con Linux tienen un driver incluido en los drivers de sonido ALSA (¡de verdad!) que simula con éxito un módem real con el uso de software en espacio de usuario que reemplaza las funciones del hardware que falta (exactamente igual que lo que haría un driver para Windows).

Aparte del módulo winmódem `mwave` incluido en el kernel, y algunos drivers winmódem en ALSA, existen varios proyectos específicos que soportan winmódems. Puede verificar la página de soporte para winmódems bajo Linux en [3]. Algunos proyectos, como `pctel-linux`, ofrecen módulos que contienen archivos objeto binarios de dudoso origen, por lo que yo no los denominaría necesariamente software libre. Los problemas con las licencias podrían ser la razón de que muchas distribuciones ya no incluyan los dri-

vers y dejen los detalles de la instalación para el usuario.

Viendo las capturas de pantalla que me ha enviado, veo que su winmódem tiene un chipset SiS que funcionaba con el driver `slamr/slmodemd` y el demonio en espacio de usuario, que no es libre (es decir, usa una licencia no distributable). Este driver por tanto no se incluye en la mayoría de las distribuciones GNU/Linux, y probablemente no existan paquetes RPM de instalación para el kernel actual. Pero, aún podemos descargarlo, compilarlo e instalarlo por nuestra cuenta [4].

Con los drivers apropiados y (en el caso de `slmodemd`) el demonio en espacio de usuario adecuado, el winmódem se comportará casi igual que un módem normal, pudiéndose usar con herramientas de marcado como `wvdial` o `kppp`. Si el módulo del kernel recién compilado no consigue crear el archivo de dispositivo `/dev/slamr0` al cargarse (`modprobe slamr`, verifique `dmesg`) vía `udev`, puede que tengamos que crear de manera manual al menos `slamr0`:

```
mknod -m 600 /dev/slamr0 c 212 0
```

Tras arrancar con:

```
slmodemd /dev/slamr0
```

`slmodemd` debería crear de manera automática un enlace simbólico `/dev/ttySLO` en `/dev`, que es el dispositivo correcto para conectarse a Internet vía `kppp` o `wvdial`.

Existen instrucciones de cómo conseguir que funcionen otros winmódems [3]. Reemplazar el winmódem con un módem real es siempre una opción. Los módems normales sólo necesitan un driver serie, y deberían funcionar con cualquier sistema operativo y distribución. ■

## RECURSOS

- [1] Código fuente de Gspca: <http://mxhaard.free.fr/download.html>
- [2] Lista de compatibilidad de Gspcat: <http://mxhaard.free.fr/spca5xx.html>
- [3] Soporte de winmódems bajo Linux: <http://linmodems.org/>
- [4] Paquetes Smartlink: <http://linmodems.technion.ac.il/packages/smartlink/>