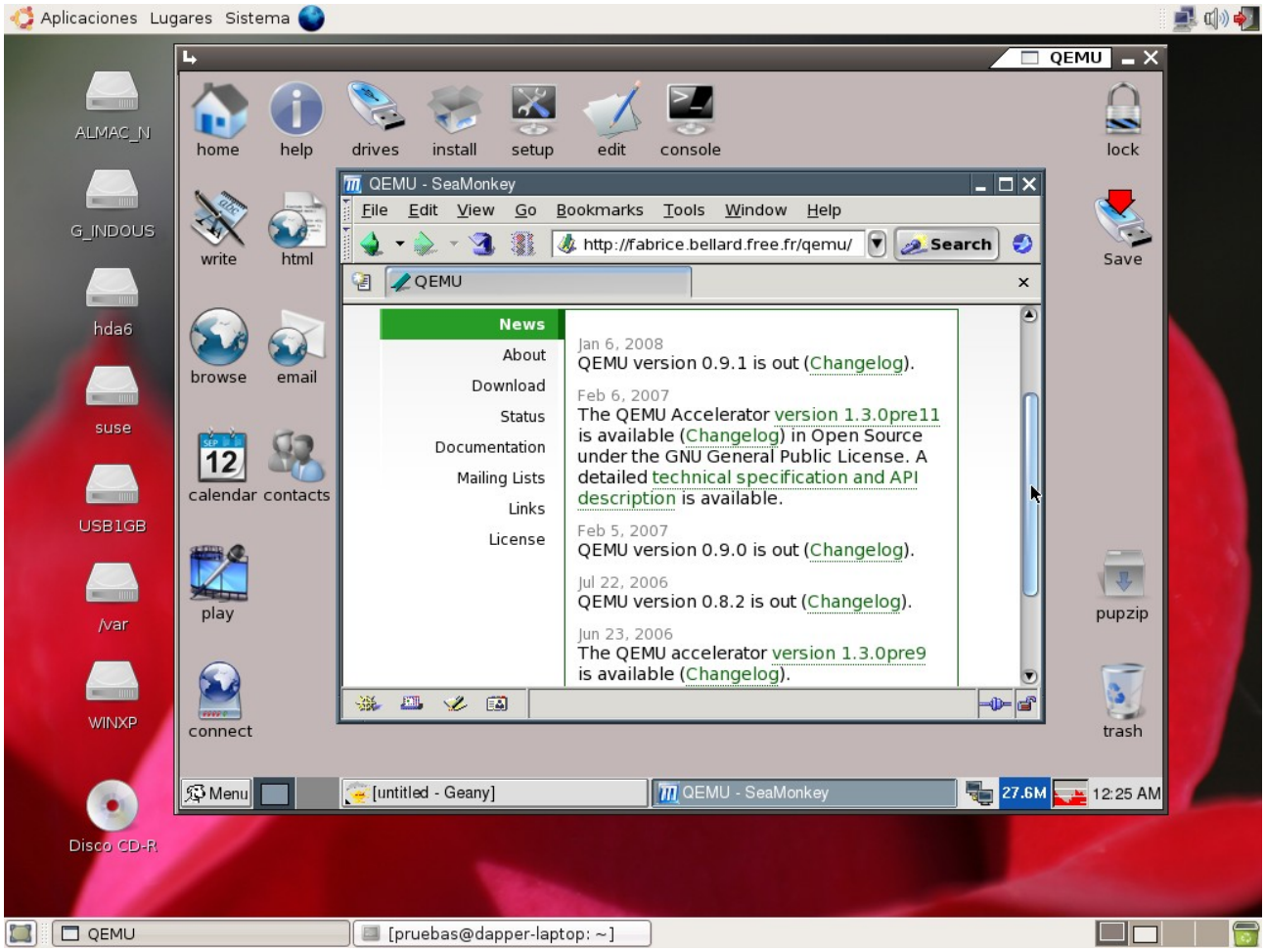
Si te acabas de incorporar al mundo de la virtualización de sistemas operativos, seguramente te suenen nombres como vmware o virtual box, pero hay hay muchos más programas, entre ellos qemu, que suele quedar olvidado aunque parte del código de software de virtualización como Xen, VirtualBox o KVM (*Kernel-based Virtual Machine*) para Linux procede de este proyecto. Quizá sea porque tiene fama de "complicado" por ser una aplicación que se maneja básicamente mediante la consola, pero yo no estoy de acuerdo con que sea tan difícil (es más, con las versiones actuales si todo lo que queremos hacer es un uso básico, me parece el MÁS sencillo (repito, es mi opinión)). Este artículo pretende demostrar que puede ser una buena opción a tener en cuenta.

Qemu es un programa gratuito y multiplataforma (disponible para Windows, Solaris, Linux, FreeBSD, NetBSD, OpenBSD, Mac OS X, ZETA y BeOS, así que no tienes excusa para no usarlo :-P) que nos permite tener un ordenador virtual ejecutándose en una ventana dentro de nuestro sistema operativo (entre otras cosas, pues también puede ejecutar programas compilados para un tipo de CPU en otro tipo de CPU – por ejemplo: ejecutar binarios para PowerPC en nuestro PC x86). Es muy bueno para hacer pruebas, pues podemos salvar el estado de la máquina virtual sin dificultad, y restaurarlo en cuanto queramos.

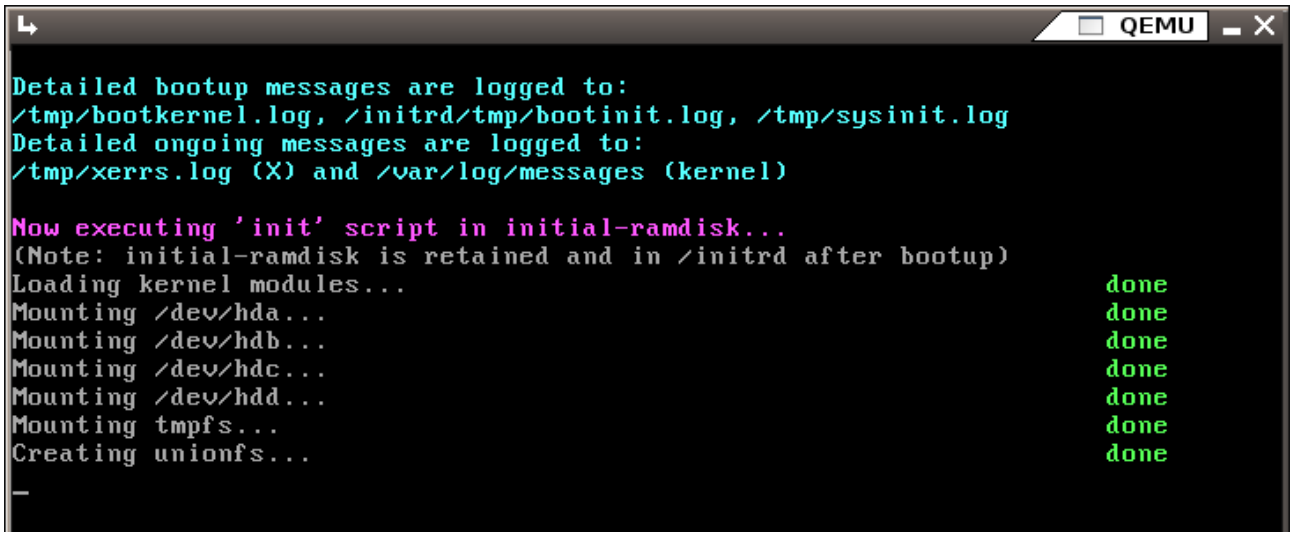


Sistema Linux ejecutándose en un PC virtual (que a su vez se ejecuta en otra instalación de Linux)

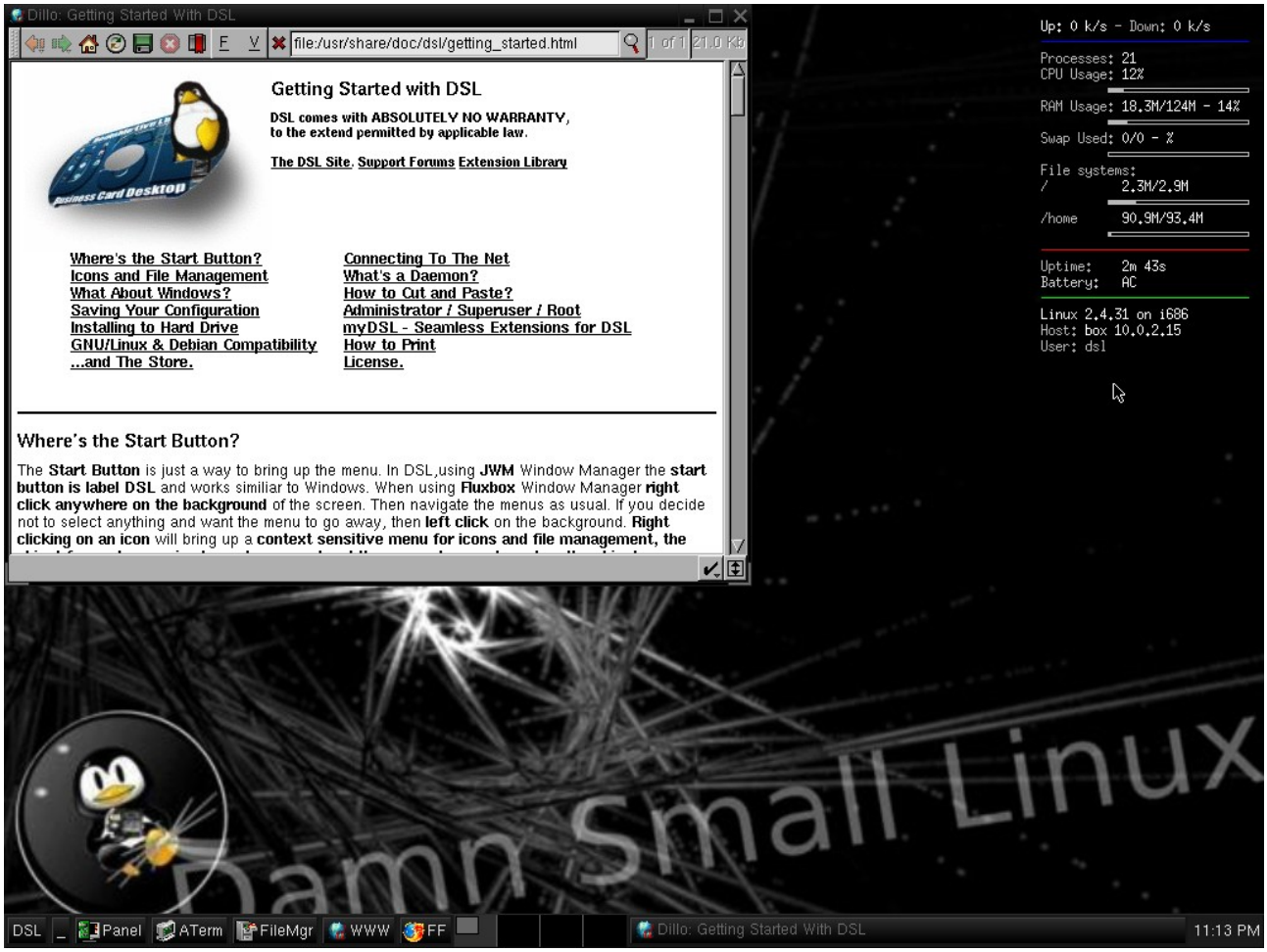
Comenzando a usar Qemu

Si tienes un sistema típico, es difícil que tengas problemas con la instalación. Aún así, ni siquiera tienes que "complicarte" con ello: puedes usar Qemu sin siquiera haberlo instalado. Hay muchas páginas que ofrecen paquetes comprimidos en los que se incluye una versión portátil de qemu + un sistema operativo listo para usar. Voy a destacar 2 de ellas: Qemu-Puppy (http://www.erikveen.dds.nl/qemupuppy) de unos 86 MB y Damn Small Linux, de 50 (http://www.damnsmalllinux.org), cuya versión *Embedded* se distribuye con qemu. Ambos se pueden descargar mediante bittorrent gracias a http://www.linuxtracker.org, o por medios más tradicionales (como FTP o HTTP), desde distintos *mirrors* que pueden consultarse en las páginas de los respectivos proyectos. Para comenzar, recomiendo Qemu-Puppy. Una vez descargado el paquete (ej:

qemu-puppy-2.17-1.tar.gz), lo desempaquetamos donde vayamos a usarlo (un buen lugar es un pendrive USB, para poder transportar fácilmente nuestro PC Virtual a cualquier parte) y ejecutamos el archivo *puppy.exe* o *puppy.sh* y obtendremos una ventana con una distribución ligera de Linux funcionando dentro.



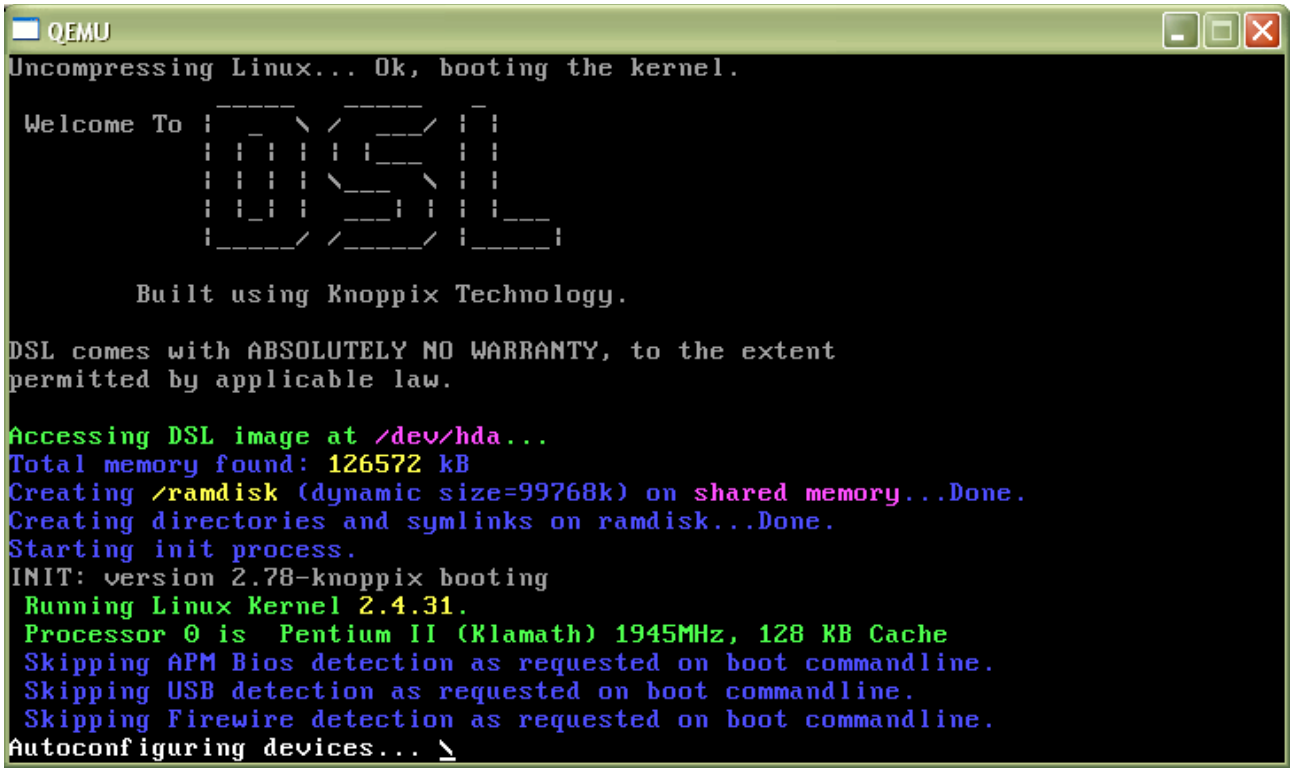
Si queremos tener el sistema en modo de pantalla completa, lo tendremos al pulsar la combinación de teclas Ctrl + Alt + f (vuelve a pulsarlas y estarás de nuevo en modo ventana). Otra combinación que debemos tener siempre presente es Ctrl + Alt, pues al pulsar con el ratón en la ventana del sistema emulado, el puntero queda "atrapado" dentro. Esta combinación permite devolver el control del puntero al sistema operativo anfitrión.



Pulsando Ctrl+Alt+f se alterna entre el modo de ventana y el modo a pantalla completa

En el caso de usar Damn Small Linux, no lo tendremos tan fácil, pero casi. Sólo se incluye la versión de qemu para Windows, y el nombre del lanzador (un archivo BAT) varía bastante de una versión a otra, por lo

que habrá que leer el readme.txt para saber cuál ejecutar. Por supuesto, el uso de estos lanzadores no es obligatorio; son sólo una ayuda para quien empiece, pero en cuanto veamos un poco más de este programa, sabrás cómo usar en su lugar nuestra propia instalación de qemu.



Ejecución de Damn Small Linux en un PC virtual sobre Windows

Instalación y uso

La instalación no tiene mucho misterio; en mi caso, desde Ubuntu Dapper, sólo tengo que hacer:

```
# apt-get install qemu
```

(claro, esto es una instalacion básica; veremos que hay complementos que pueden requerir más complicaciones). La instalación de este paquete y otros 3 más que se añaden como dependencias, requiere descargar unos 4 MB (una vez instalado, ocupa el triple), así que incluso quienes tengáis una conexión por modem podreis instalarlo sin agotar vuestra paciencia, y quienes tengais un equipo con poco espacio disponible, no supone un gran sacrificio tener instalado este programa. Me parece recomendable tenerlo siempre a mano, pues es muy útil para comprobar rápidamente si cierto disco que tengamos por ahí es capaz de arrancar y cómo lo hace (si alguna vez tratas de elaborar tu propio disco arrancable, verás lo maravilloso que es este programa para evitar tener que andar arrancando una y otra vez para probar el dichoso disco).

Por supuesto, no es necesario hacer uso de apt, puedes usar el programa de instalación que prefieras o que necesite tu sistema: synaptic, yum, urpmi... Si por alguna razón te fallan los repositorios o quieres estar a la última, o conseguir las fuentes, puedes encontrar el paquete adecuado en la página de su autor: http://fabrice.bellard.free.fr/qemu/ desde donde también podemos bajar complementos y algunas imágenes de disco (mira en las secciones *Donloads* y *Links*).

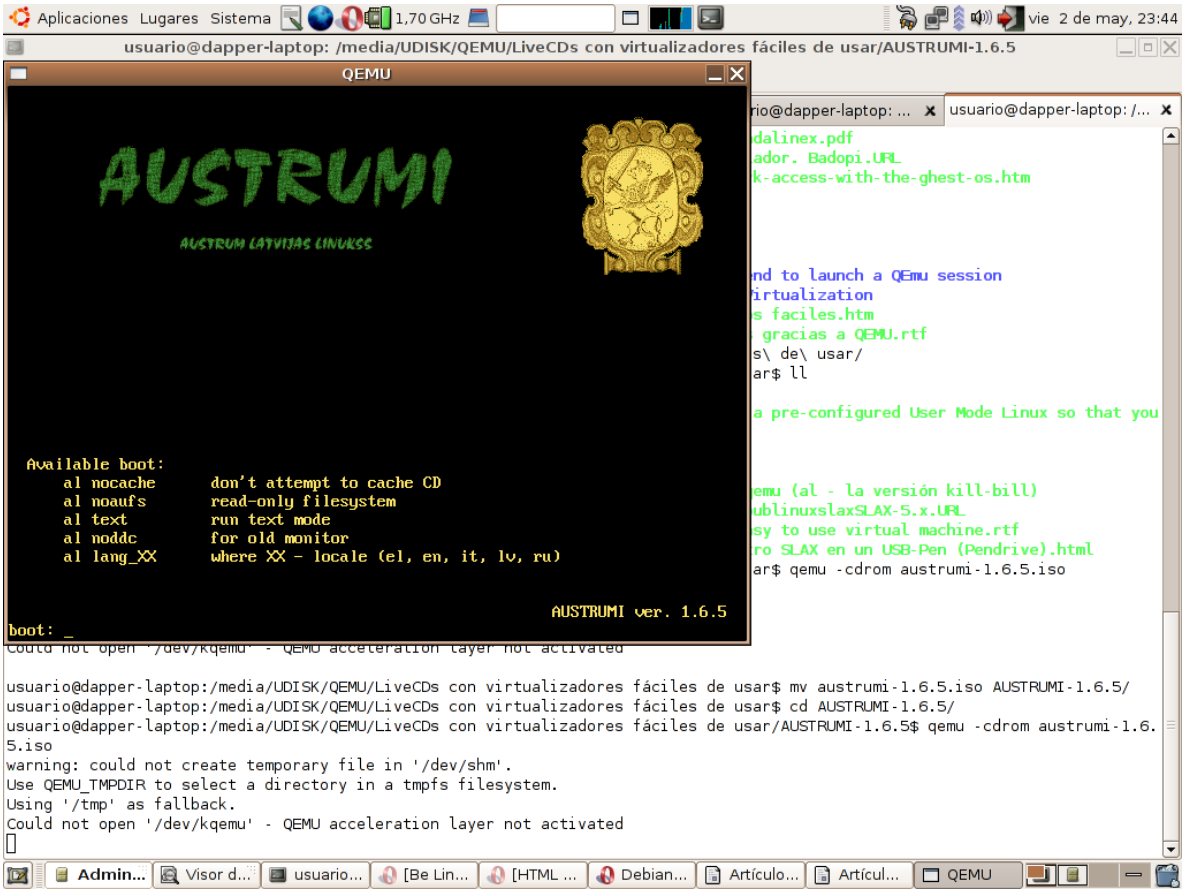
Hay interfaces gráficas para manejar qemu, pero de momento voy a mostrar el manejo desde la línea de órdenes y más adelante mencionaré algunas interfaces que pueden usarse para quien prefiera manejar el programa con el ratón. Para ejecutar qemu, debemos pasarle una serie de parámetros; como mínimo el "dispositivo" desde el que ha de arrancar la máquina virtual. Lo he puesto entre comillas porque puede ser un dispositivo real, pero también puede ser una imagen de disco (como una imagen ISO de un CD-ROM o una imagen de un disquete).

Por ejemplo, me acabo de bajar la distribución Austrumi y quiero ver qué tal es sin tener que grabarla a CD,

reiniciar el PC, etc. Con qemu es bien sencillo; sólo tengo que ejecutar:

```
$ qemu -cdrom austrumi-1.6.5.iso
```

y al momento aparece una ventana conteniendo la máquina virtual:



Si en vez de tratarse de una imagen, quisiera arrancar un CD/DVD real, bastaría con cambiar el archivo hacia el dispositivo que simboliza el lector de discos ópticos:

```
$ qemu -cdrom /dev/hdc
```

Por defecto, el programa reserva 128 MB de memoria para la máquina virtual; esto puede ser excesivo o demasiado poco, según nuestras necesidades, así que podemos asignar la cifra que queramos detrás de la opción **-m**.

```
$ qemu -cdrom /dev/hdc -m 96
```

Para que nuestro PC virtual tenga disqueteras, usaremos las opciones **-fda** y **-fdb**. Por ejemplo, arrancaré una imagen de disquete de FreeDOS (bajada de la página de qemu) y le doy acceso físico a la disquetera con:

```
$ qemu -fda odin1440.img -fdb /dev/fd0
```

Si queremos usar discos duros o lector(es) de CDs usaremos las opciones -hda, -hdb, -hdc y -hdd para indicarle los distintos dispositivos IDE con los que contará el PC virtual. **Nota:** no se pueden usar las opciones -cdrom y -hdc al mismo tiempo (la opción -cdrom crea un lector virtual como maestro del canal secundario, con lo que entra en conflicto con -hdc).

```
$ qemu -hda imagen_de_disco_duro.img -hdb /dev/cdrom -hdc imagen_de_cd.iso -hdd otra_imagen.bin
```

Opción de arranque

En los primeros ejemplos no se incluyó esta opción porque si sólo incluímos un dispositivo, arrancará con él. Sin embargo, al incluir varios, quizá necesitemos indicarle específicamente con cuál arrancar; para ello usaremos la opción -boot *X*, siendo *X*:

a para disquete,
d para CD-ROM y
c para el disco duro.

Si no usamos la opción -boot, por defecto el PC usará el disco duro para arrancar. Podemos incluir esta opción varias veces para que trate de iniciar desde otro dispositivo si falla el primero (empieza a probar por el último *boot* añadido):

```
$  qemu -fda odin1440.img -cdrom ../LiveCDs/AUSTRUMI-1.6.5/austrumi-1.6.5.iso
-m 32 -hda imagen_disco.dsk -boot d -boot a
```

En todos los casos, si hay varios dispositivos del mismo tipo, tratará de arrancar desde el principal (ej: disco primario maestro, disquetera primaria, etc.).

Como puede verse, la longitud de la línea a ejecutar va aumentando rápidamente según añadimos dispositivos (¡y estamos viendo sólo las opciones más básicas!). Si no quieres estar tecleando todo esto cada vez, puedes crear un simple *shell script*; copia la línea en un archivo de texto (ej: *archivo.sh*) con permisos de ejecución (chmod +x archivo.sh) que comience con la siguiente línea:

#!/bin/sh
(debajo pega la orden a ejecutar)

Luego, cada vez que quieras ejecutar el sistema virtualizado no tendrás que hacer nada más que llamar al script:

```
$ ./archivo.sh
```

Imágenes de discos

Las imágenes de discos que usamos para arrancar el PC virtual o para contener el sistema pueden haber sido creadas por nosotros o haberlas conseguido de otro sitio (de la web, redes de intercambio entre pares, etc). Hay muchas páginas que recopilan máquinas virtuales y una de las más recomendables es el proyecto FreeOsZoo (http://www.oszoo.org/), donde se pueden encontrar bastantes imágenes de sistemas operativos libres listos para ser ejecutados con qemu. Algunas de ellas incluso pueden ejecutarse en línea (aunque este es un servicio bastante experimental y poco seguro), mediante un navegador con el plugin de Java o el *appletviewer*.

A la hora de crear nuestras propias imágenes de disco, tenemos unas cuantas opciones. Una posibilidad es hacer uso de la orden dd, la cual nos permite sacar una copia de un disco físico o crear una imagen a nuestro gusto “copiando” el dispositivo /dev/zero. Por ejemplo:

```
$ dd if=/dev/zero of=imagen_disco count=2048 bs=512
```

Crea una imagen de 2048 bloques de 512 KB cada uno (o sea, un “disco” de 1 MB). En esa imagen podremos crear un sistema de ficheros por medio de las órdenes mkfs* o se lo pasamos a la máquina virtual directamente, para formatearlo desde el sistema operativo invitado.

Este tipo de imágenes es muy básico y para crear discos virtuales grandes, puede que nos interese otros más avanzados, que permitan comprimir el disco y variar el tamaño del fichero real según necesitemos ocupar más espacio. Sin embargo, este tipo básico merece tenerse en cuenta por dos razones: para empezar, la comprensión de disco puede afectar al rendimiento, sobre todo si al ir variando de tamaño se va fragmentando el fichero en cuestión. Otro motivo es que si necesitamos acceder desde el anfitrión a algún archivo almacenado, no nos hará falta ejecutar qemu, conectar en red el sistema anfitrión y el invitado, etc. Todo lo que tendríamos que hacer es montar el archivo usando un dispositivo *loop* como en el ejemplo:

```
$ mount -o loop imagen_disco directorio_de_montaje/
```

A pesar de lo dicho, puede interesarnos usar otro tipo de discos; y con qemu, tenemos varios para elegir. Una manera muy fácil de crear una imagen es:

```
$ qemu-img create imagen.img 3500M
```

Lo que crea el fichero imagen.img para usarlo como disco duro virtual de cerca de 3,5 GB. El fichero creado es una imagen raw; es bastante similar al creado con dd, pero permite que al formatearlo con sistemas de ficheros como ext2, ext3 o NTFS, sólo ocupe el tamaño de los archivos almacenados y no los 3500 MB al completo. Para saber cuál es el tamaño real y del fichero, se usa la opción **info** de qemu-image, pues ls puede devolver información erronea sobre estos ficheros:

```
$ qemu-img info imagen.img
image: imagen.img
file format: raw
virtual size: 33G (35651584000 bytes)
disk size: 0
```

Podemos especificar otros formatos de imagen añadiendo al final -f tipo_de_imagen, pudiendo usar, **vmdk** (lo que permitiría usar el disco virtual en VMWare), **qcow2** (que admite compresión zlib y cifrado AES) y más formatos que puedes consultar ejecutado man qemu-img o en la documentación de qemu. También podemos convertir ficheros de un formato a otro mediante la opción **convert**. Por ejemplo:

```
$ qemu-img convert -f raw imagen.img -O vmdk imagen.vmdk
```

En este caso, -f raw indica el formato del fichero de origen y -O vmdk el del que vamos a crear, ('imagen.vmdk').

El “disco virtual” también puede ser un disco físico; supongamos que tengo un problema de virus con otro ordenador y quiero arrancar su sistema en una máquina virtual. Puedo conectar el disco en cuestión (por ej, asociado a /dev/hdb) y arrancarlo así:

```
$ qemu -hda /dev/hdb
```

Sin embargo, hacer esto puede tener sus riesgos si el disco también lo monta el sistema anfitrión. Al estar el sistema montado a la vez por el anfitrión y el invitado, puede acabar corrompiéndose el sistema de ficheros. Por lo tanto, es aconsejable montar los discos físicos con la opción **-snapshot**

```
$ qemu -hda /dev/hdb -snapshot
```

Lo cual indica a la máquina virtual que guarde los cambios realizados en un fichero temporal, y no en el disco; si finalmente decidimos que queremos que los cambios se guarden en el disco, usaremos la orden **commit** del *modo monitor* (ahora veremos lo que es) para ello.

Manipulando la máquina virtual

De momento, nos hemos limitado a ejecutar un sistema en una máquina virtual, pero no hemos visto cómo manipular dicha máquina; ¿cómo podemos pausar su ejecución, salvar su estado, modificarla, etc? Muy fácil: llamando al *monitor*. Esto lo podemos realizar de dos formas: cuando estemos en la ventana de la máquina virtual, pulsando Ctrl+Alt+2 (Ctrl+Alt+1 para volver) o bien al ejecutar qemu, pasándole la opción **-monitor stdio**. Normalmente, al ejecutar qemu la terminal desde la que lo hemos lanzado queda a la espera de que termine este programa; con la opción mencionada, tras ejecutar la máquina virtual, aparecerá la línea de órdenes del modo monitor. Ésta se reconoce por el prompt:

```
(qemu)
```

y nos permite, entre otras, ejecutar estas órdenes:

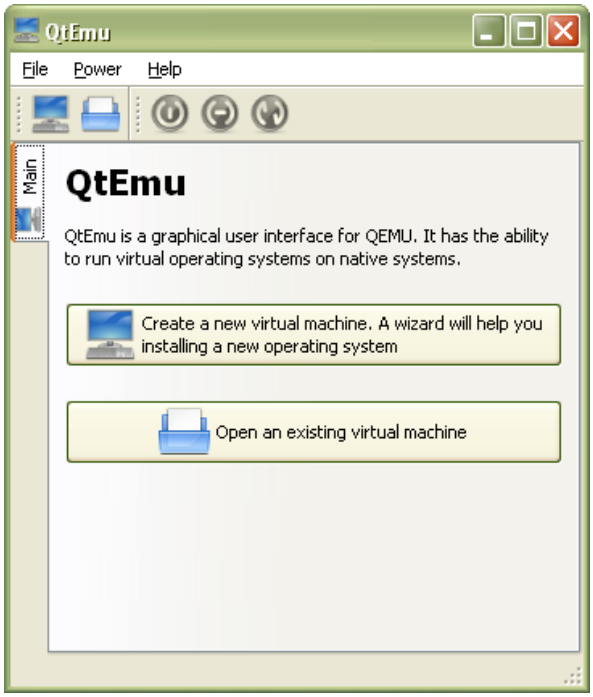
stop: pausar la ejecución de la máquina virtual
cont (o simplemente **c**): reanuda la ejecución
eject dispositivo: expulsa un medio removible
change dispositivo nuevo_dispositivo: cambia un medio removible por otro.
savevm nombre_archivo: guarda el estado de la máquina virtual en el archivo nombrado.
loadvm nombre_archivo: restaura el estado de la máquina virtual.
help (o simplemente **?**): listado de órdenes disponibles (hay muchas más que las mencionadas)

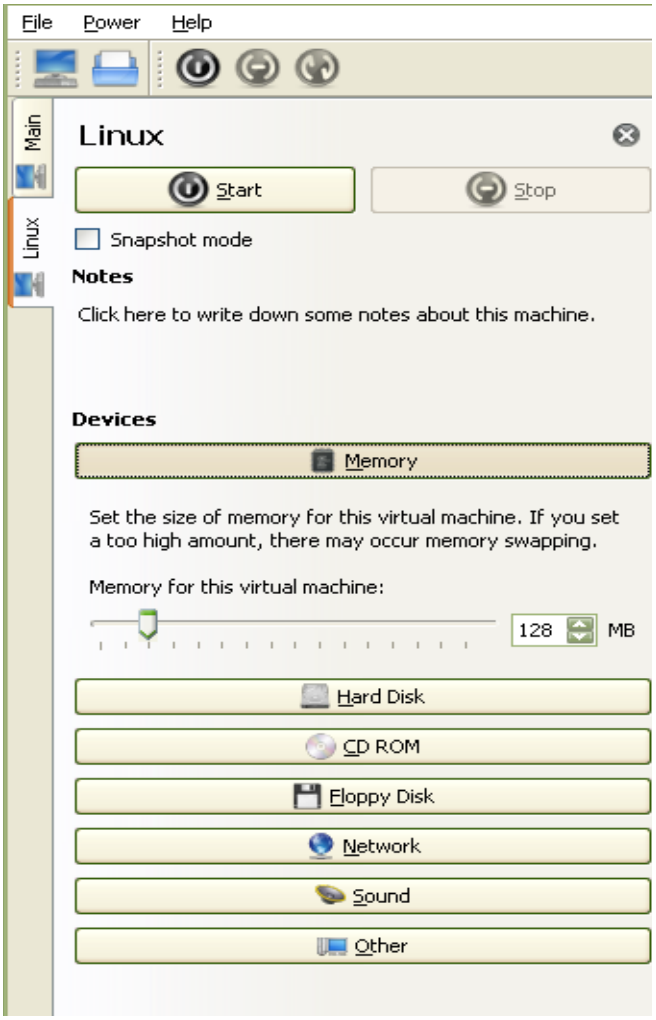
Se me acaba el espacio y aún no he tocado dos temas que me parecen importantes: el módulo acelerador *kqemu* y las interfaces gráficas para Qemu. El primero suele estar disponible con los últimos paquetes, pero no está de más comprobarlo. Puede mejorar MUCHO la velocidad del sistema virtual y si no lo tienes ya, puedes bajarlo de la página de Qemu.

Interfaces gráficas para Qemu

En cuanto a las interfaces gráficas; hay muchas y aquí mencionaré sólo las más recomendables (en mi opinión):

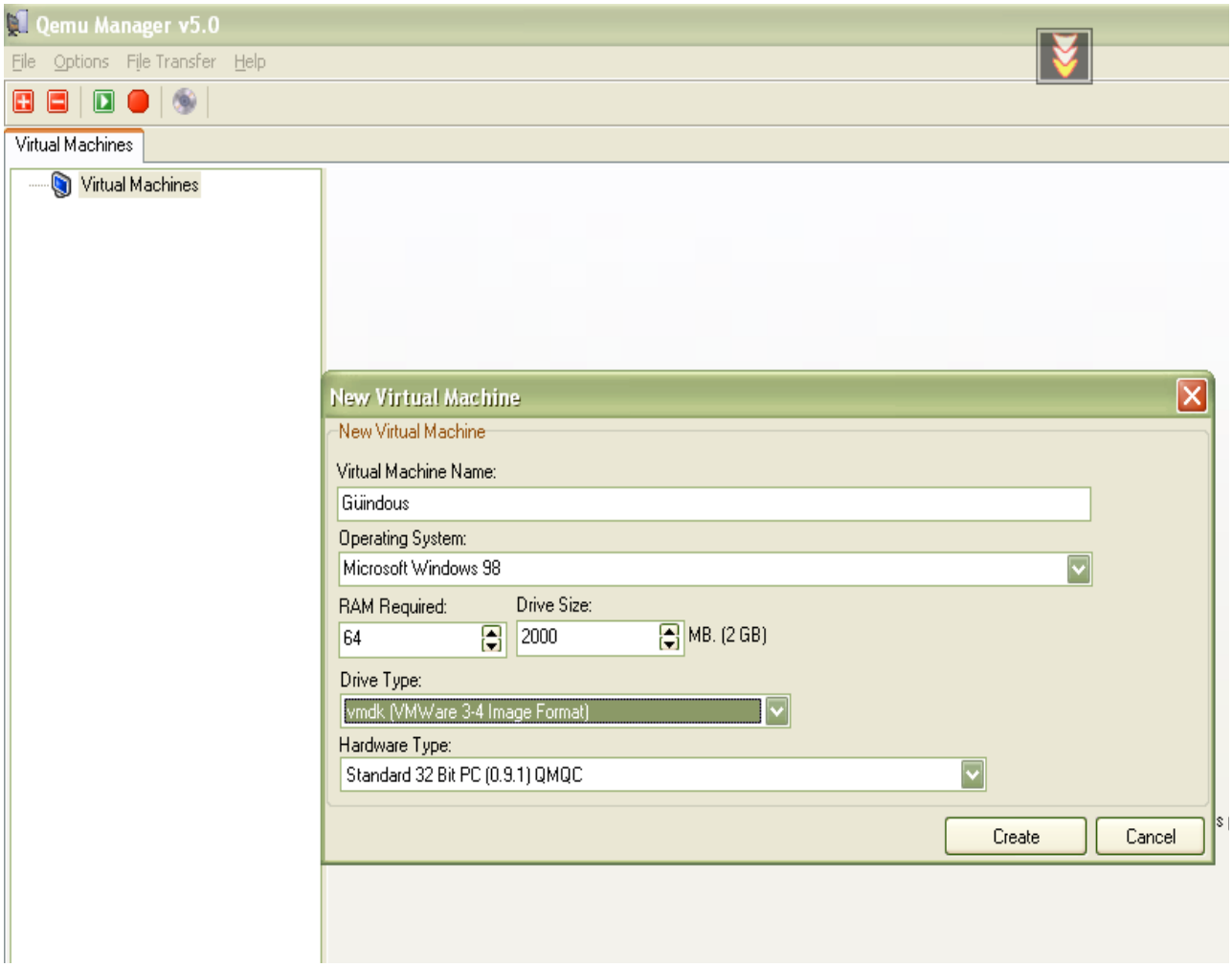
QTemu (http://qtemu.org): es un programa multiplataforma muy sencillo de instalar y ejecutar; el único problema que podríais tener con su instalación son las dependencias: requiere las librerías Qt 4.2 (el instalador para Windows ya las trae).





QGui (http://perso.wanadoo.es/comike): es un programa muy pequeño y que no necesita instalación, por lo que podremos llevarlo con nosotros en un disquete / disco USB. Sus principales inconvenientes son que sólo dispone de versión para Windows y que la versión de qemu que incorpora es bastante antigua (lo podemos arreglar facilmente, sustituyendo los archivos por otros de una versión más nueva de qemu). Por último, es más engorroso de manejar que las otras interfaces que comento.

Qemu Manager (http://www.davereyn.co.uk): de nuevo, otro programa sólo disponible para Windows, pero tampoco necesita instalación, por lo que podemos llevarlo con nosotros sin problemas (aunque este no cabe en un disquete; ocupa unos 15 MB y más del doble al descomprimirlo).



Qemoon (http://ebellard.free.fr/qemoon/): programa multiplataforma y que no requiere instalación (aunque debes tener Java 1.5 o superior para que funcione). Aún está en fase de desarrollo, pero si necesitas una interfaz para realizar tareas simples con qemu, puede ser tu opción. Por desgracia, es bastante pesado, aunque quizá su rendimiento mejore según vaya progresando.

Conclusión

Llego al final del artículo y sólo he llegado a tocar unos cuantos aspectos de este programa.No sé si habré conseguido mi objetivo de comunicar la buena opción que es Qemu. Muchos usuarios quizá tengan la impresión de que es el “primo pobre” de otros programas como VirtualBox o las distintas soluciones de VMWare. Quizá en algunos casos sean más apropiadas las opciones que acabo de mencionar, pero para la mayoría creo que qemu es la opción a elegir. ¿Por qué? Veamos: requiere muy pocos recursos de hardware, después de tantos años de su aparición está más que probado, se puede introducir en un disco USB sin problemas (al menos, la versión para Windows), lo que permite llevar nuestro PC Virtual a cualquier lado y ejecutar el sistema operativo que queramos en un ordenador típico (que suele ser un PC con Windows), con sólo enchufar el disco y sin siquiera tener que reiniciar. Es libre, al contrario que VMWare (lo cual a muchos les puede dar igual, pero para otros es un dato a tener muy en cuenta). El programa por sí mismo no tiene interfaz gráfica, pero si estamos dispuestos a teclear un poco, podremos ver que precisamente esa interfaz de texto tiene una flexibilidad y potencia difícil de igualar por sus rivales – y si no nos interesan más que las opciones más básicas, creo haber mostrado que son bastante sencillas. Quizá algún usuario se sienta intimidado por las interfaces de texto. Desde aquí le animo a que pruebe este programa, aunque sólo sea por ver sus opciones más simples. No creo que quede decepcionado.

Espero que os sea útil artículo sobre un programa tan interesante como Qemu. La verdad es que sólo se han llegado a ver las opciones que a mi juicio son más importantes para poder beneficiarse de este programa sin complicarse demasiado, pero éste tiene muchas, muchas posibilidades que no se han llegado

ni a mencionar. Quizá en un futuro pueda escribir otros artículos detallando más las posibilidades de Qemu. Mientras tanto, si me quieres hacer llegar algún comentario sobre este artículo, puedes mandarme un correo a danifp25@yahoo.es. También me puedes encontrar en la web, en mi blog danubuntu.wordpress.com, donde comento noticias e información sobre software y cultura libres. ¡Nos leemos!