DVD ROM SUSE 9.2 PROFESIONAL

Versión profesional completa DVD de 7.5 GB

SoundJuicer Ripea CDs desde el escritorio Gnome

Kooldock Un panel de control vistoso

Edición en Castellano

LINUX MAGAZINE

NÚMERO 7 • P.V.P 5,95 €

Emulación de Windows, Mac y Linux

SISTEMAS VIRTUALES

Debianiza tu SuSE

Juguetes

BlueFish p35

Un completo IDE para el desarrollo web

8 413042 594505 00007

Primeros pasos con User-Mode Linux

LINUX DENTRO DE LINUX

User-Mode Linux parece Linux porque es Linux. Encontraremos cientos de usos para este rápido y práctico sistema virtual de Linux. **POR FABRIZIO CIACHI**

El versatil y popular User-Mode Linux (UML) [1] crea un sistema Linux completamente funcional en un host Linux. UML tiene numerosas aplicaciones en el mundo de Linux. Muchos desarrolladores confían en UML para probar sus aplicaciones sin poner todo el sistema en peligro. Los usuarios de Linux pueden ejecutar UML para experimentar con versiones del kernel sin tener que preocuparse de parches nuevos o experimentales. Los administradores de sistemas usan UML para probar configuraciones. Es posible incluso ejecutar varias versiones de UML en el mismo equipo para simular una red.

¿Qué es User-Mode Linux?

User-Mode Linux no es realmente un emulador, ni una API. La mejor manera de explicarlo es comenzar con un vistazo al papel del kernel de Linux.

El kernel ejecuta procesos y se comunica con el hardware. Cuando un proce-

SeLinux dentro de UML

Encontraremos un documento muy interesante que explica cómo configurar un sistema UML con SELinux en [15]. Un sistema UML con SELinux habilitado puede ser muy útil para crear servidores más seguros y probar los principios de SELinux sin poner el sistema en peligro.

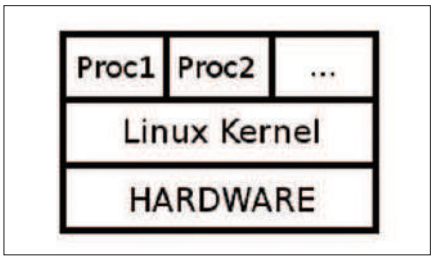

Figura 1: Estructura habitual de procesos en Linux .

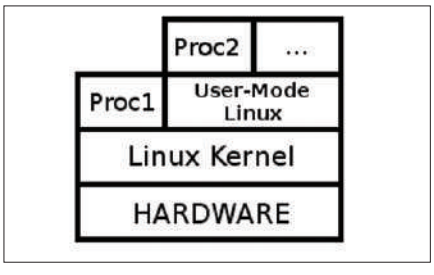

Figura 2: UML se ejecuta como un proceso. En esta figura, Proc1 se ejecuta en el sistema host Linux, mientras que Proc2 lo hace en un sistema virtual User-Mode Linux.

so quiere comunicarse con un dispositivo (por ejemplo para mostrar algo por pantalla, imprimir un documento o copiar un archivo a un disco), el proceso le pide al kernel de Linux controlar la comunicación con el hardware (ver Figura 1).

User-Mode Linux es un kernel de Linux que se ejecuta como un proceso. La diferencia entre un kernel UML y un kernel ordinario es que el kernel UML no se comunica directamente con el hardware. En su lugar, los comandos pasan al kernel real del sistema que lo alberga, el cual controla la comunicación con el hardware.

Debido a que el sistema virtual y el sistema host son ambos Linux con estructuras prácticamente idénticas, la comunicación pasa muy eficientemente del sistema virtual al host, requiriendo muy pocas cabeceras de abstracción o traducción.

Configurar UML

Podemos instalar User-Mode Linux con el administrador de paquetes. Por ejemplo, con Debian necesitamos teclear las siguientes líneas como root:

```
#apt-get install➥
user-mode-linux➥
uml-utilities➥
kernel-patch-uml
```

Esta orden instala el kernel UML y también otras utilidades. Con otros administradores de paquetes el proceso es igualmente simple, pero si aparecen problemas con estos, o tenemos problemas de memoria durante el arranque [2], podemos descargar un kernel Linux normal (recomendamos la versión 2.4.27 [3]) y el parche UML [4]. Podemos encontrar otros paquetes UML en [5]. Cuando hayamos descargado los archivos del parche y el kernel (en el mismo directorio), abrimos un terminal y ejecutamos las siguientes órdenes:

```
$ bunzip2 linux-2.4.27.tar.bz2
$ tar -xvf linux-2.4.27.tar
$ bunzip2 uml-patch-2.4.27-1.bz2
$ patch -p1 -d linux-2.4.27➥
< uml-patch-2.4.27-1
$ cd linux-2.4.27
$ make menuconfig ARCH=um
$ make linux ARCH=um
$ strip linux
```

Después de completar con éxito estas órdenes,tendremos un arhivo llamado "linux" en nuestro directorio original. Este archivo es el kernel de User-Mode Linux que usaremos para arrancar el sistema virtual Linux.

Para conseguir que UML funcione adecuadamente, necesitamos incluir

QEMU: Una buena alternativa

Si nuestro interés en UML es probar distribuciones Linux, podemos optar asimismo por el programa de emulación de sistemas QEMU [12]. QEMU (que está basado en Bochs [13]) es muy fácil de instalar, configurar y usar. Puede encontrar más información sobre QEMU en el artículo "Mundos Virtuales: Emulación de Sistemas con QEMU" en Linux Magazine #5, de abril de 2005 [14].

Tabla 1: Tipos de transporte

Etherap, TUN/TAP	Transportes usados para intercambio de paquetes entre el sistema virtual y el host real.
Switch daemon	Un transporte diseñado para la interconexión de redes virtuales con otros sistemas UML.
Multicast	Otro transporte diseñado para interconexión de redes virtuales
Slip, slirp	Transporte usado principalmente cuando Etherap o TUN/TAP no están disponibles o si no tenemos acceso como root a la configuración de redes del host.
Pcap	Un transporte que facilita una interfaz de red de sólo-lectura y es, por tanto, una buena opción para monitorización de redes.

Elegir emulación

Quizás la mejor manera de entender las ventajas de UML es considerar que existen tres posibilidades en este tipo de software:

*Emulación del software

*Emulación del hardware

*Sin emulación

Bochs es uno de los sistemas de emulación del software más famosos. Su principal característica es emular la arquitectura hardware IA-32, llamada también x86 por encima del sistema operativo, ya sea Windows, MacOS o Linux. Dado que es el hardware lo que se emula, se puede instalar cualquier sistema operativo x86 sobre él (Linux, Windows, Dos y demás), pero la ejecución es muy lenta, ya que se necesita traducir cada instrucción del sistema operativo invitado al sistema operativo que lo alberga.

La emulación del hardware consiste en código compilado en la arquitectura hardware nativa. Un emulador hardware es más eficiente que el emulador software, pero necesita interceptar todas las llamadas al hardware. Esta solución tiene como gran desventaja que el código debe particularizarse para una arquitectura hardware que debe ser la misma tanto para el entorno del host como del alojado. Un ejemplo de este tipo de emulador es VMware [5], un sistema de emulación comercial.

User-Mode Linux pertenece a esta última categoría. No necesita emular un hardware específico, sino que se comunica casi directamente con el hardware real. Las instrucciones pasan eficientemente desde el kernel de UML hasta el kernel del host. UML puede ejecutar código nativo, y alcanza un retardo de un 20% como máximo respecto de la ejecución del mismo código en el host.

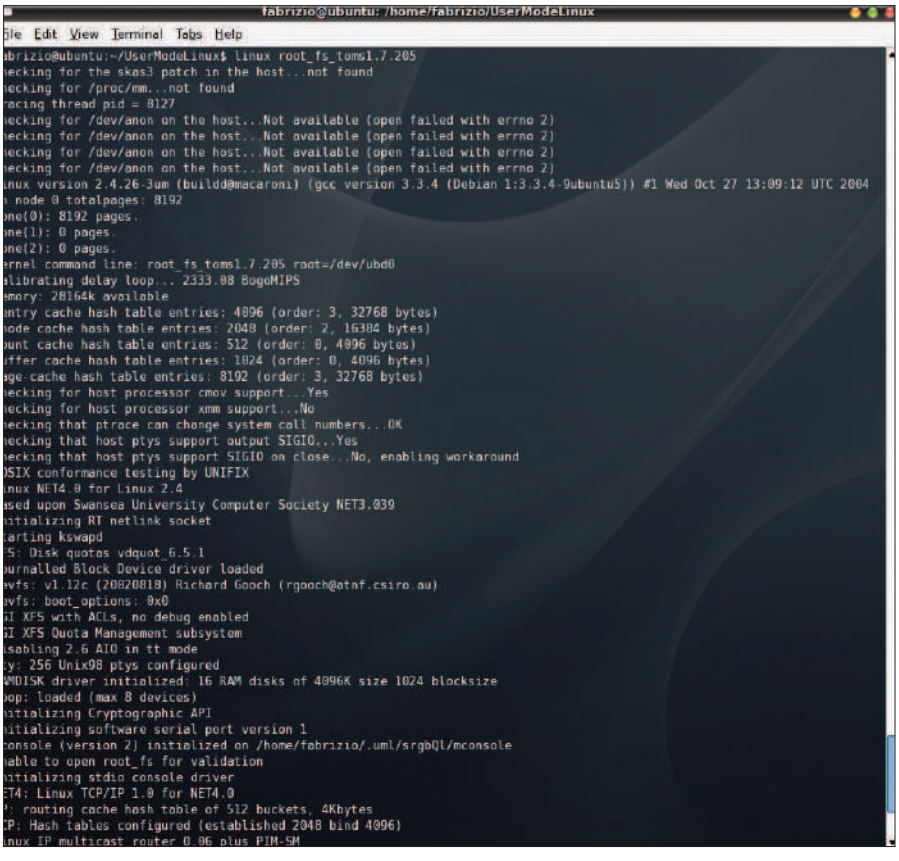

Figura 3: Arranque del sistema virtual UML.

otros dos elementos mas en el puzzle: un sistema de ficheros (una imagen comprimida de una partición de Linux que contenga todos los programas) y las utilidades UML. Para el sistema de ficheros raíz, podemos encontrar todas las imágenes disponibles en [6]. Necesitaremos descargar las utilidades UML desde [7] y teclear las siguientes órdenes:

```
$ bunzip2 uml_utilities➥
_XXXXXXXX.tar.gz
$ tar -xvf uml_utilities➥
_XXXXXXXX.tar
$ cd tools
$ make all
$ make install DESTDIR=/
```

Ahora tenemos un directorio que contiene un sistema de archivos raíz. No olvidemos colocar el programa *linux* en un lugar donde podamos usarlo (si no lo hemos movido, estará aún en el directorio del *linux-2.4.27*). Tecleamos las siguientes órdenes para leer el sistema de archivos raíz:

```
$ bunzip2 ➥
root_fs_toms1.7.205.bz2
```

```
$ linux ➥
ubd0=root_fs_toms1.7.205
```

El parámetro *ubd0=* le indica al sistema virtual que use el archivo especificado como el sistema de archivos raiz.

Si todo ha ido bien, veremos el sistema virtual arrancando (véase Figura 3), y podemos entonces entrar en el sistema virtual con el nombre de usuario *root* y contraseña *root*.

Compartir el sistema de archivos raíz

Podemos arrancar dos o más máquinas virtuales usando el mismo sistema de ficheros raíz. El driver *udb0* usa un mecanismo llamado Copy-On-Write (COW), que lee el sistema de archivos raíz como un dispositivo compartido de sólo-lectura y guarda los cambios en un archivo privado de lectura/escritura (el archivo COW). Por ejemplo, si queremos arrancar dos máquinas virtuales (VM1 y VM2) con el mismo sistema de ficheros, necesitamos abrir dos sesiones de terminal y teclear las siguientes órdenes:

```
[xterm 1]$ linux➥
ubd0=uml_vm1.cow,➥
root_fs_toms1.7.205
[xterm 2]$ linux➥
ubd0=uml_vm2.cow,➥
root_fs_toms1.7.205
```

Todas las modificaciones de los dos hosts virtuales se escribirán con sus respectivos archivos COW. En realidad, el sistema de archivos no se comparte, dado que ambas ejecuciones son independientes una de la otra. Al crear dos archivos COW es importante evitar el arranque del sistema de ficheros directamente (con *ubd0=root_fs_XXX*), por que cada archivo COW registra el tamaño y la marca de tiempo del sistema de archivos raíz, y cualquier modificación hará que los ficheros COW queden inutilizados. La sintaxis correcta para el siguiente reinicio, cuando tengamos un archivo COW es la siguiente:

```
[xterm 1]$ linux➥
ubd0=uml_vm1.cow
[xterm 2]$ linux➥
ubd0=uml_vm2.cow
```

Redes Virtuales y rReales

UML nos ofrece unas cuantas opciones interesantes para redes virtuales en sistemas Linux. Una vez tengamos nuestro sistema virtual UML listo y ejecutándose, puede que queramos experimentar la conexión entre el sistema virtual con su sistema host o con otros sistemas virtuales. Encontraremos una descripción exhaustiva de interconexión de redes con UML en [8].

La idea principal en la interconexión de redes con UML es que

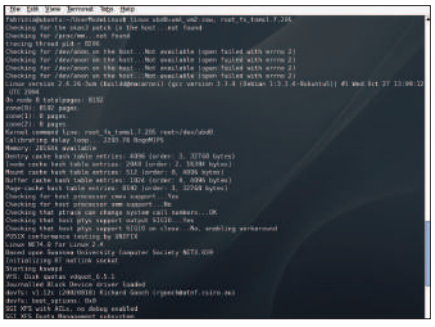

Figura 4: Arranque del segundo sistema virtual UML.servicios de red disponibles.

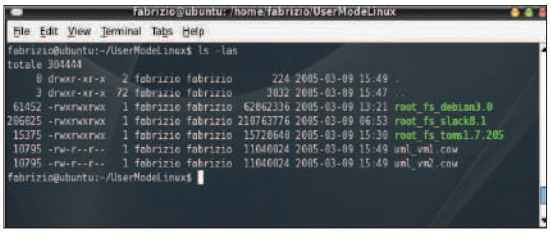

Figura 5: Ficheros COW de dos máquinas virtuales UML.

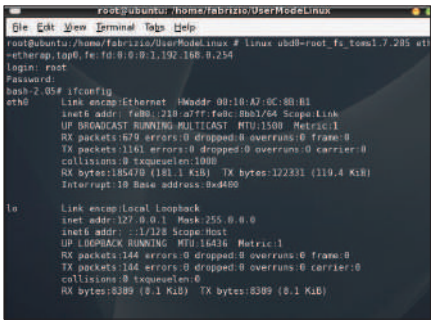

Figura 6: Una máquina virtual UML con los servicios de red disponibles.

nos facilitan distintas opciones en la capa de transporte para la administración de paquetes entre el sistema virtual y su host. La tabla 1 muestra algunos de los tipos de transporte disponibles en UML.

Para habilitar un dispositivo de red en la máquina virtual, le pasamos una cadena como la siguiente en la línea de comandos del kernel:

```
eth<n>=➥
    <transport>,<transport
    args>
```

Donde <n> representa la interfaz del host real (p.ej.: eth0) al que se conectará la máquina virtual. La explicación teórica de esto es que en la máquina virtual UML hay un dispositivo eth0 que corresponde al dispositivo tap0 en el host real. Esta interfaz tap0 se conecta directamente a la interfaz eth0 del host real.

Por lo tanto podemos teclear:

```
linux ubd0=root_fs_slack8.1➥
eth0=ethertap,tap0,➥
fe:fd:0:0:0:1,192.168.0.254
```

para permitir a UML configurar eth0 en la máquina virtual con su propia dirección IP.

La dirección IP del tap0 real y el eth0 virtual puede ser la misma para simplificar la configuración. (Véase [8] para configuraciones de red más complejas).

Necesitaremos configurar la interfaz en la máquina virtual (*/etc/hosts*, */etc/resolv.conf*, */etc/network*, etc.) para tener un acceso a Internet plenamente operativo en el entorno UML.

Conclusión

User-Mode Linux nos permite una manera rápida y conveniente de crear sistemas virtuales Linux dentro de Linux. Podemos utilizar UML como herramienta para planificar, modelar, probar y buscar fallos en sistemas Linux. UML es también la base de muchos otros proyectos de código abierto y experimentos, así como de aplicaciones para negocios y servicios personalizados de hospedaje. Puede que User-Mode Linux no sea fácil de instalar y configurar, pero si somos capaces de ponerlo en marcha, le encontraremos multitud de usos. ■

RECURSOS

[1] Página de User-Mode Linux: *http://user-mode-linux.sourceforge.net*

[2] UML en hosts 2G/2G: *http://user-mode-linux.sourceforge.net/UserModeLinux-HOWTO-4.html#2G-2G*

[3] Kernel Oficial de Linux 2.4.27: *http://ftp.ca.kernel.org/linux/kernel/v2.4/linux-2.4.27.tar.bz2*

[4] Parche UML para el kernel 2.4.27: *http://prdownloads.sourceforge.net/user-mode-linux/uml-patch-2.4.27-1.bz2*

[5] Descargas UML: *http://user-mode-linux.sourceforge.net/dl-sf.html*

[6] Lista de sistemas de archivos: *http://user-mode-linux.sourceforge.net/dl-jails-sf.html*

[7] Utilidades UML: *http://prdownloads.sourceforge.net/user-mode-linux/uml_utilities_20040406.tar.bz2*

[8] Configuración de redes en UML: *http://user-mode-linux.sourceforge.net/networking.html*

[9] Compilación del kernel: *http://user-mode-linux.sourceforge.net/compile.html*

[10] Depurado del Kernel: *http://user-mode-linux.sourceforge.net/debugging.html*

[11] Sesión de depurado UML: *http://user-mode-linux.sourceforge.net/debug-session.html*

[12] Página de QEMU: *http://fabrice.bellard.free.fr/qemu/*

[13] Página de Bochs: *http://bochs.sourceforge.net*

[14] Artículos obre QEMU: *http://www.linux-magazine.es/issue/05*

[15] SELinux y UML: *http://www.golden-gryphon.com/software/security/selinux-uml.xhtml*

EL AUTOR

Fabrizio Ciacchi (http://fabrizio.ciacchi.it – fabrizio@ciacchi.it) es un estudiante italiano de Ciencias de la Computación en la Universidad de Pisa. Sus actividades principales son estudiar Linux, desarrollar páginas Web con PHP y programar en Java. Trabaja también como consultor para distintas compañías y escribe artículos sobre Linux.