Getting started with User-Mode Linux

LINUX IN LINUX

User-Mode Linux feels like Linux because it is Linux. You'll find a hundred uses for this fast and sensible virtual Linux system. **BY FABRIZIO CIACCHI**

The popular and versatile User-Mode Linux (UML) [1] creates a fully operatational virtual Linux system on a Linux host. UML has many uses in the Linux world. Developers rely on UML to test their applications without putting the host system at risk. Linux users run UML to experiment with kernel versions without having to worry about a new or untested patch. System administrators use UML to test system configurations. You can even run multiple versions of UML on the same host to simulate a network.

SELinux into UML

There is a very interesting document that explain how to set up an SELinux-enabled UML system at [15]. An SELinux-enabled UML can be very useful for creating more secure servers and testing SELinux policy without putting the system at risk.

What is User-Mode Linux?

User-Mode Linux is not really an emulator, nor is it an API. The best way to explain User-Mode Linux is to start with a look at the role of the Linux kernel.

The kernel runs processes and talks with the hardware. When a process wants to communicate with a device (for instance, to display something on a monitor, print a document, or copy a file to a floppy), the process asks the Linux kernel to manage the communication with the hardware (Figure 1).

User-Mode Linux is a Linux kernel that runs in Linux as a process. The difference between a UML kernel and an ordinary kernel is that the UML kernel does not communicate directly with the hardware. Commands pass instead to the "real" Linux host kernel, which manages the hardware communication.

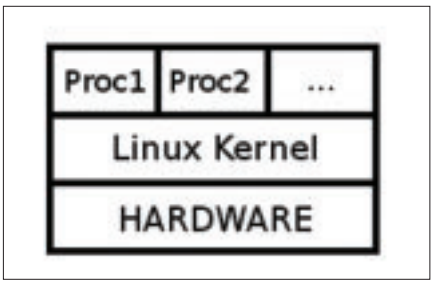

Figure 1: Normal Linux process structure.

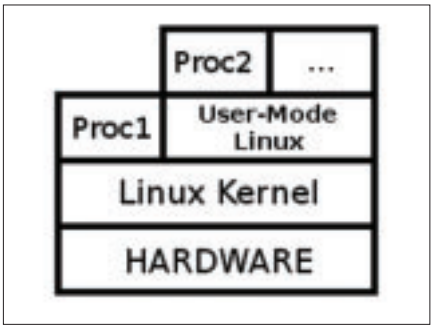

Figure 2: UML runs as a process. In this figure, Proc1 is running on the host Linux system. Proc2 is running on the User-Mode Linux virtual system.

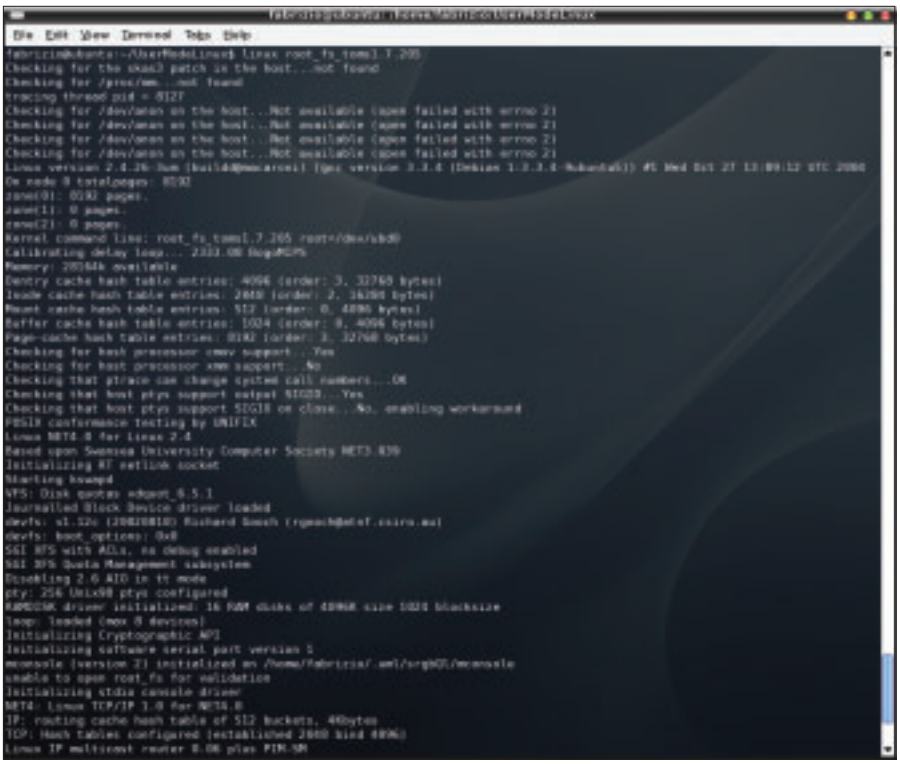

Figure 3: Booting the UML virtual system.

Because the virtual system and the host system are both Linux systems with nearly identical structures, the communication passes very efficiently from the virtual system to the host, requiring minimal overhead for abstraction or translation.

Setting Up UML

You can install User-Mode Linux with your package manager. For example, with Debian, you need to give this command as root:

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

This command installs the UML kernel and also other utilities. Other package managers are equally simple, but if you have a problem installing from a package system, or if you have memory problems during boot [2], you may wish to download a normal linux kernel (we recommend the 2.4.27 version [3]) and the UML kernel patch [4]. You can find other UML patches at [5]. When you have downloaded the patch and kernel files (in the same directory, of course), open a terminal window and execute the following commands:

```
$ 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
```

After you sucessfully complete these commands, you will have a file called "linux" in your original directory. This file is the User-Mode Linux kernel that will be used to boot the virtual linux system.

To make UML work properly, you need to include two other pieces of the puzzle: a root filesystem (a compressed image of a linux partition that contains

Emulation Choice

Perhaps the best way of understanding the advantages of UML is to consider that software of this type comes in three forms:

- Software emulation
- Hardware emulation
- No emulation

Bochs [4] is one of the most famous software system emulators. The principal activity of Bochs is to supply an emulation of a particular hardware architecture (IA-32, called also x86) on top of a particular operating systems, like Windows, MacOS and, obviously, Linux. Once the hardware is emulated, it is possible to install any x86 operating system on it (Linux, Windows, Dos and so on), but the execution is very slow, because every computer instruction needs to be translated from the guest operating system to the host operating system.

Hardware emulation consists of the code built on the native hardware architecture. A hardware emulator is more efficient than the software emulator but it needs to intercept all the calls to the hardware. This solution has the big disadvantage that the code must be specialized for a particular hardware architecture that is the same for host and guest environment. An example of this type of emulator is VMware [5], a commercial system emulator.

User-Mode Linux fits in the last category; it doesn't need to emulate any specific hardware, but it instead talks nearly directly with the real hardware. Instructions pass efficiently from the UML kernel through the host kernel. UML can execute native code and can run with, at worst, a 20% slowdown compared to running the same code on the host.

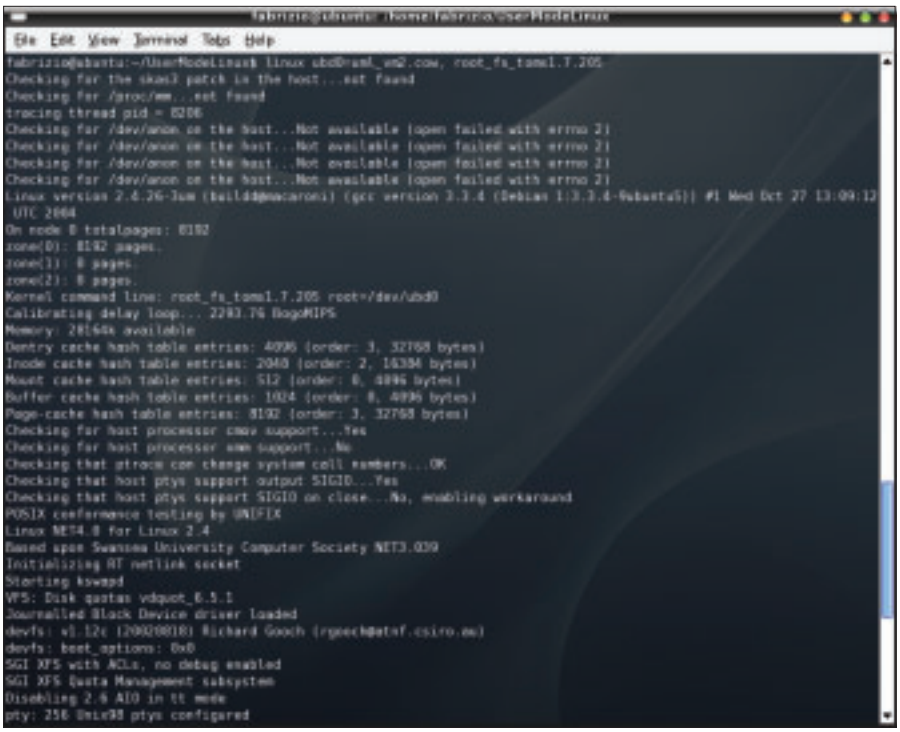

Figure 4: Booting process of the second UML virtual system.

all the programs) and the UML utilities. For the root filesystem, you can find all the available images at [6]. You'll need to download the UML utilities from [7] and type the following commands:

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

You now have a directory that contains the root filesystem. Remember to put the *linux* program in a location that permits you to use it. (If you haven't moved the *linux* program, it is still in the linux-2.4.27 directory.) Then enter the following commands to read the root filesystem:

```
$ bunzip2 ➥
root_fs_toms1.7.205.bz2
$ linux ➥
ubd0=root_fs_toms1.7.205
```

The *ubd0=* parameter tells the virtual system to use the file specified as the root filesystem.

If all goes well, you'll see the virtual system booting (Figure 3), and you can log in to the virtual system with the username *root* and the password *root*.

Sharing the Root File System

It is possible to launch two or more virtual machines using the same root filesystem. The *udb0* driver uses a mechanism called Copy-On-Write (COW), which reads the root filesystem as a read-only shared device and stores the changes in a read/write private file (the COW file). For example, if you want to start two virtual machines (VM1 and VM2) with the same filesystem, you need to open two terminal sessions and write the following commands:

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

All the modifications to the two virtual hosts will be written on the respective COW files. In truth, the filesystem is not shared, but the two executions are independent of each other. The most important thing to avoid when the two COW files are created is booting the filesystem directly (with *ubd0=root_fs_XXX*), because every cow file has registered the size and the timestamp of the root filesystem, and every modification will make the COW files unusable. The correct syntax to use for the next reboot, when we have a COW file, is as follows:

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

Virtual and Real Networking

UML provides several interesting options for networking virtual Linux systems. Once you get your UML virtual system up and running, you may wish to experiment with networking the virtual system with its host or networking it with other virtual systems. You'll find a thorough description of UML networking at [8].

The basic idea behind UML networking is that several optional *transports* are provided for managing the exchange of packets between the virtual system and the host. Table 1 shows some of the transport types available for UML.

QEMU: A Good Alternative

If your goal is to use UML for testing new Linux distributions, you can opt for QEMU [12] system emulator application. QEMU (based on Bochs [13]) is very simple to install, set up, and use. For more on QEMU, see "Virtual Benefits: System Emulation with QEMU" in Linux Magazine Issue #52, March 2005; you can download the article in Pdf format from our archive [14].

Table 1: UML Transport Types

Etherap, TUN/TAP	Transports used for exchanging packets between the virtual system and the real host.
Switch daemon	A transport designed for purely virtual networking with other UML systems.
Multicast	Another transport designed for virtual networking.
Slip, slirp	Transports used primarily when Ethertap and TUN/TAP are not available or if you don't have root access to the networking configuration on the host.
Pcap	A transport that provides a read-only network interface and is, therefore, a good option for network monitoring.

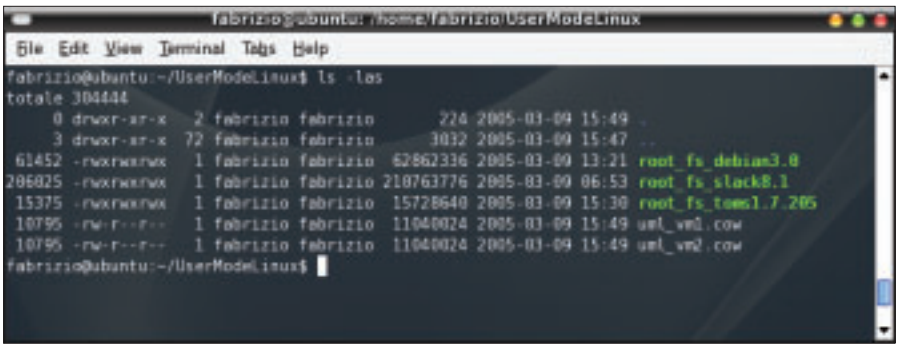

Figure 5: The COW files of two UML Virtual Machines.

To enable a network device in the virtual machine, pass a string like the following to the kernel command line:

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

where <n> represent the real host interface (i.e., eth0) to which the virtual machine will attach. The theoretical explanation is that, in the UML virtual machine, there is an eth0 device that corresponds to a tap0 device on the real host; this tap0 interface is directly connected to the eth0 interface of the real host.

So, we can use the command:

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

to permit UML to set up eth0 in the virtual machine with its own IP address.

The IP address of real tap0 and virtual eth0 can be the same for simpler configurations. (See [8] for more complex network configurations.)

You then need to set up the interface in the virtual machine (*/etc/hosts*, */etc/resolv.conf*, */etc/network*, etc.) to have fully operative Internet access in the UML environment.

Conclusion

User-Mode Linux provides a quick and convenient means for creating virtual Linux systems in Linux. You can use can UML as a tool for planning, modeling, testing, and troubleshooting Linux systems. UML is also the basis for many other open source projects and experiments, as well as for business applications and personalized hosting services. Perhaps User-Mode Linux is not so easy to install and configure, but if you can get it working, you'll find many uses for it. ■

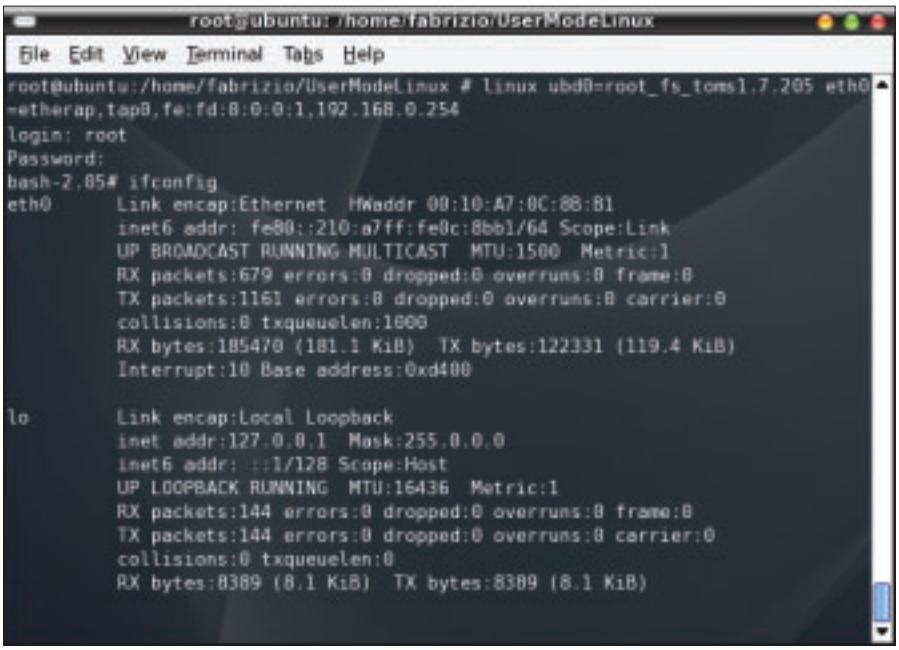

Figure 6: A UML Virtual Machine with network services available.

INFO

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

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

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

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

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

[6] Root filesystem list: *http://user-mode-linux.sourceforge.net/dl-jails-sf.html*

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

[8] UML Network configuration: *http://user-mode-linux.sourceforge.net/networking.html*

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

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

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

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

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

[14] QEMU Article "Virtual Benefits": *http://www.linux-magazine.com/issue/52/QEMU_System_Emulation.pdf*

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

THE AUTHOR

Fabrizio Ciacchi (*http://fabrizio.ciacchi.it – fabrizio@ciacchi.it*) is an italian student of Computer Science at the University of Pisa. His main activities are studying Linux, developing Web Sites in PHP, and programming in Java. He also works as a consultant for several companies and writes articles on Linux.