# Capítulo 5

# “Ejecución de Instancias UML”

## 5.1 Introducción

En el capítulo anterior se preparó el entorno para comenzar a trabajar con las máquinas virtuales UML, sin embargo es necesario previo a la ejecución de una máquina virtual tener a disposición un sistema de archivo ya que si se ejecuta *User Mode Linux* sin un sistema de archivos se presentara el error ilustrado en la Figura 5.1. Una máquina virtual se inicia desde la consola utilizando el comando en su forma más simple: # ./linux

En esta situación el *kernel* UML busca un sistema de archivo raíz denominado *root_fs* dentro del subdirectorio donde se ejecuta se produce un *Kernel Panic* que aborta el inicio de la máquina virtual.

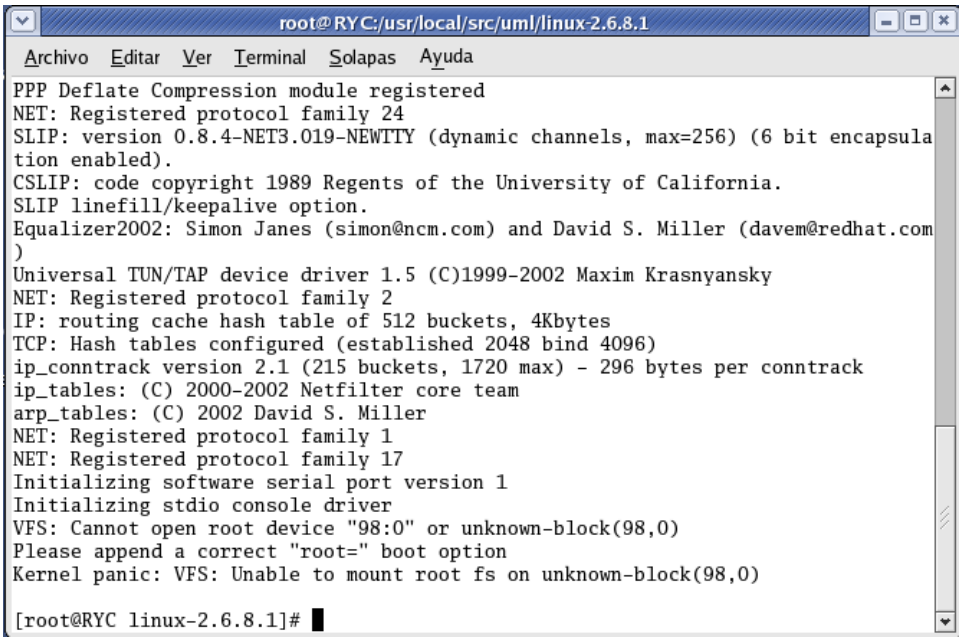


Figura 5. 1: *Kernel Panic* al ejecutar una instancia UML

El comando ./l*inux* permite incorporar varios parámetros adicionales, a continuación se explicaran los más importantes:

- *showconfig*: Imprime el archivo de configuración que fue generado por el UML.
- *mem=<cantidad de RAM deseada>*: Permite determinar la cantidad de

memoria que se le asignará a la máquina virtual. El tamaño es especificado por un número seguido de 'k', 'K', 'm', 'M', cuyo significado es obvio. Ejemplo: mem=64M.

- *versión*: muestra en pantalla el número de versión del *kernel*.

- *help*: Mediante la opción help es posible acceder a la ayuda relacionada con los distintos argumentos que se pueden introducir a continuación de ./linux.

- *umid=<nombre>*: Con este parámetro se asigna un nombre de identificación único para la máquina UML.

- *eth[0-9]+=<transporte>,<opciones>*: Configura un dispositivo de red, para mayor detalle ver el apartado 5 de este capítulo, donde se describen los modos de transporte y sus opciones.

- *ubd<n>=<nombre de archivo>*: Es utilizado para asociar un dispositivo (unidad disco virtual, CDROM, Floppy, etc.) con un archivo del host anfitrión. *User Mode Linux* requiere un sistema de archivos root_fs y el dispositivo ubd (user block device) provee un mecanismo para mapear root_fs y que pueda ser accedido como sistema de archivos desde la máquina virtual. Los discos en UML son referenciados como ubd seguido de una letra o un número que los distingue y particulariza. Ejemplo: #./linux ubd0=root_fs ubd1=/dev/fd0 ubd2=/dev/cdrom

- con[0-9]*=<descripción de canal>: añade una consola UML a un canal del host. En este proyecto se utilizarán los siguientes canales:

    - *none: el dispositivo desaparece completamente, es decir la consola correspondiente no se levantará.

    - *null: el dispositivo es dirigido a /dev/null, se obstruye si se trata de leer.

    - *xterm: será ejecutado con los argumentos apropiados para añadir la consola del UML a una terminal del host. Facilita el manejo de la consola principal.

    - *otros: existen otros canales para ser utilizados [1].

Para continuar es necesario un sistema de archivos sobre el que ejecutar *User Mode Linux*. En este punto existen dos posibilidades, crearlo o descargar de Internet un sistema de ficheros preconstruido de alguna de las numerosas distribuciones de Linux existentes. En el sitio web de UML [1], se dispone de varios sistemas de archivos preparados y configurados. Estos cuentan con todos los comandos de sistema y programas de aplicación requeridos, así como también con algún espacio libre para guardar datos y descargar e instalar aplicaciones de usuario. Se pueden descargar desde diferentes sitios [33] *filesystems* basados en distribuciones como *Red Hat, Mandrake, Slackware, etc*; todas ellas con algunos servicios instalados tal como: SMTP, POP3, SAMBA, NFS, etc. En el ámbito de esta tesis se utilizó la distribución Slackware v 9.1 que cuenta con un sitio web en donde se encuentran disponibles los paquetes necesarios para adaptar el sistema de archivos de acuerdo a las necesidades y a la versión de *Slackware* con la que se esté trabajando [16].

Luego de haber descrito UML y los componentes necesarios para su instalación es preciso tratar con aspectos generales relativos a inconvenientes que se pueden presentar a la hora de ejecutar una instancia de UML.

### 5.1.1 Adaptando el Sistema de Archivos

Uno de los mayores inconvenientes de los sistemas de archivos preconstruidos es que raramente se disponen de todas las aplicaciones para cada perfil de usuario. Cuando este es el problema no queda otra alternativa que incorporarlas al sistema de archivos. En el ámbito de esta tesis fue necesario incorporar herramientas de compilación, *sniffer*, etc, etc. En este caso en particular fue necesario descargar los siguientes paquetes de *software* [24].

- binutils-2.14.90.0.6-148-1.tgz
- make-3.80-i386-1.tgz
- gcc-g++-3.2.3-i486-1.tgz
- gcc-3.2.3-i486-1.tgz
- gdb-5.3-i386-1.tgz
- ipsec-tools-0.5.1-i486-1maew.tgz
- mc-4.6.0-i386-1.tgz
- tcpdump-3.8.3-i486-1jim.tgz
- wget-1.8.2-i386-1.tgz

El usuario puede descargarlo desde la página oficial de *Slackware* y guardarlo en una carpeta en el host anfitrión. Una vez que se dispone de los paquetes listados, el siguiente paso es ejecutar una terminal virtual para proceder a la instalación de los mismos:

```
#./ linux ubd0 = root_fs
```

Por defecto no existen recursos compartidos o conexión entre ambos *filesystems*, de las máquinas anfitriona y virtual. Para acceder a los paquetes guardados en el host anfitrion, se debe crear una carpeta en el sistema de archivos de la máquina virtual y a continuación se montará en ella el sistema de archivos de la máquina anfitriona. A continuación se presenta un breve resumen de los comandos utilizados para lograr lo anteriormente mencionado:

En la máquina virtual se crea la carpeta *host*:

```
uml#mkdir /mnt/host
```

Mediante el comando *mount* se monta el sistema de archivos del host anfitrión para poder accederlo desde la máquina virtual[1]:

```
uml# mount -t hostfs none /mnt/host
```

[1] Nota: Para un correcto funcionamiento durante la compilación del *kernel* se debieron haber seleccionado las siguientes opciones:
*support for host-based *filesystems*
* *host filesystem*

Ahora los paquetes de *software* están disponibles desde la máquina virtual y el siguiente paso será instalarlos, para lo cual es necesario copiarlos en el directorio *root* del *filesystem* de la máquina virtual:

```
uml#cd /mnt/host/usr/local/src/uml

uml# cp binutils-2.14.90.0.6-148-1.tgz  /root

uml#cp make-3.80-i386-1.tgz  /root

uml# cp gcc-g++-3.2.3-i486-1.tgz  /root

uml# cp gcc-3.2.3-i486-1.tgz /root

uml# cp gdb-5.3-i386-1.tgz  /root

uml# cp ipsec-tools-0.5.1-i486-1maew.tgz  /root

uml# cp mc-4.6.0-i386-1.tgz  /root

uml#cp tcpdump-3.8.3-i486-1jim.tgz  /root

uml# cp wget-1.8.2-i386-1.tgz  /root
```

Mediante los siguientes comandos se consigue una instalación simple de las aplicaciones requeridas.

```
uml# installpkg  binutils-2.14.90.0.6-148-1.tgz

uml# rm  binutils-2.14.90.0.6-148-1.tgz

uml#installpkg make-3.80-i386-1.tgz

uml#rm make-3.80-i386-1.tgz

uml# installpkg gcc-g++-3.2.3-i486-1.tgz

uml# rm gcc-g++-3.2.3-i486-1.tgz

uml# installpkg gcc-3.2.3-i486-1.tgz

uml# rm gcc-3.2.3-i486-1.tgz

uml# installpkg gdb-5.3-i386-1.tgz

uml# rm gdb-5.3-i386-1.tgz

uml# installpkg ipsec-tools-0.5.1-i486-1maew.tgz

uml# rm ipsec-tools-0.5.1-i486-1maew.tgz

uml# installkpg mc-4.6.0-i386-1.tgz

uml# rm mc-4.6.0-i386-1.tgz
```

uml#installpkg tcpdump-3.8.3-i486-1jim.tgz

uml#rm tcpdump-3.8.3-i486-1jim.tgz

uml# installpkg wget-1.8.2-i386-1.tgz

uml# rm wget-1.8.2-i386-1.tgz

Con los comandos anteriormente mencionados se instalaron los paquetes y el sistema de archivos queda en condiciones óptimas para ser utilizado. Accediendo desde una terminal virtual a */var/log/packages* se pueden ver los paquetes instalados en el sistema de archivos.

## 5.2 Ejecutando Múltiples Instancias de Linux

Si se intentara ejecutar más de una instancia UML utilizando un sólo *filesystem* la situación sería tal cual se ilustra en la Figura 5.2. La primera instancia virtual de UML se inicia sin problemas pero cuando se intenta ejecutar la segunda instancia mediante el comando ./linux, el proceso se detiene. Este comportamiento responde a que el sistema de archivos root_fs se encuentra ocupado con la primera instancia de UML. Una rápida solución sería disponer de múltiples sistemas de archivos (uno por cada instancia de UML) con nombres tales como: root_fs0, root_fs1, etc.

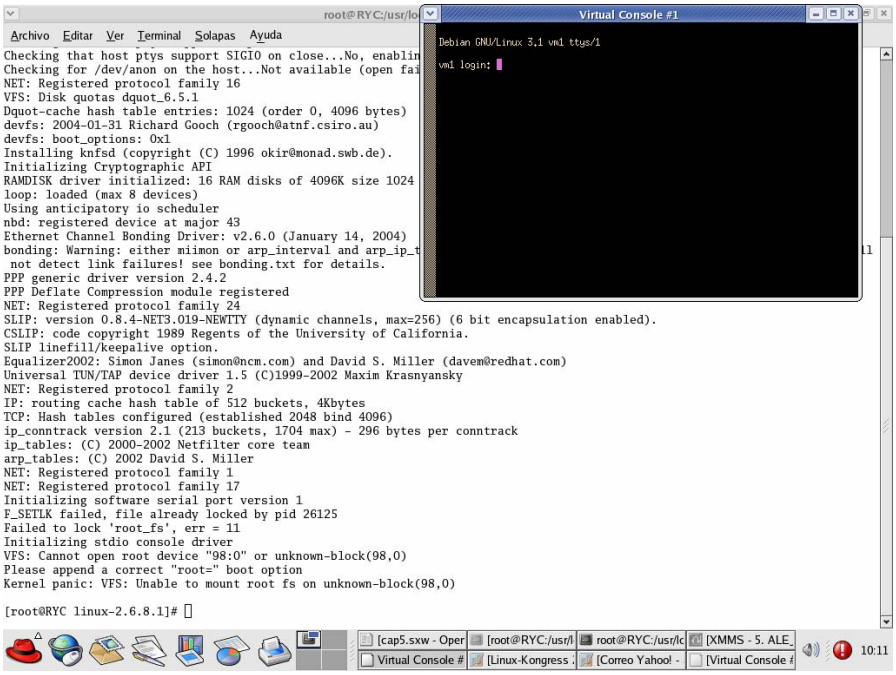


Figura 5. 2: *Kernel Panic* al ejecutar dos instancias de UML con el mismo root_fs

## 5.3 COPY ON WRITE

Para evitar el inconveniente señalado en el punto anterior los diseñadores de UML contemplaron el uso de la técnica COW *(Copy On Write).*

Mediante COW se dispone de un único *filesystem* (de sólo lectura) para todas las instancias UML y las modificaciones necesarias se escriben en un archivo adicional propio para cada instancia (archivo .COW). Ello implica un considerable ahorro de espacio en disco [1]. En la

Figura 5.3 se ilustra el resultado de la ejecución de dos instancias virtuales de UML.

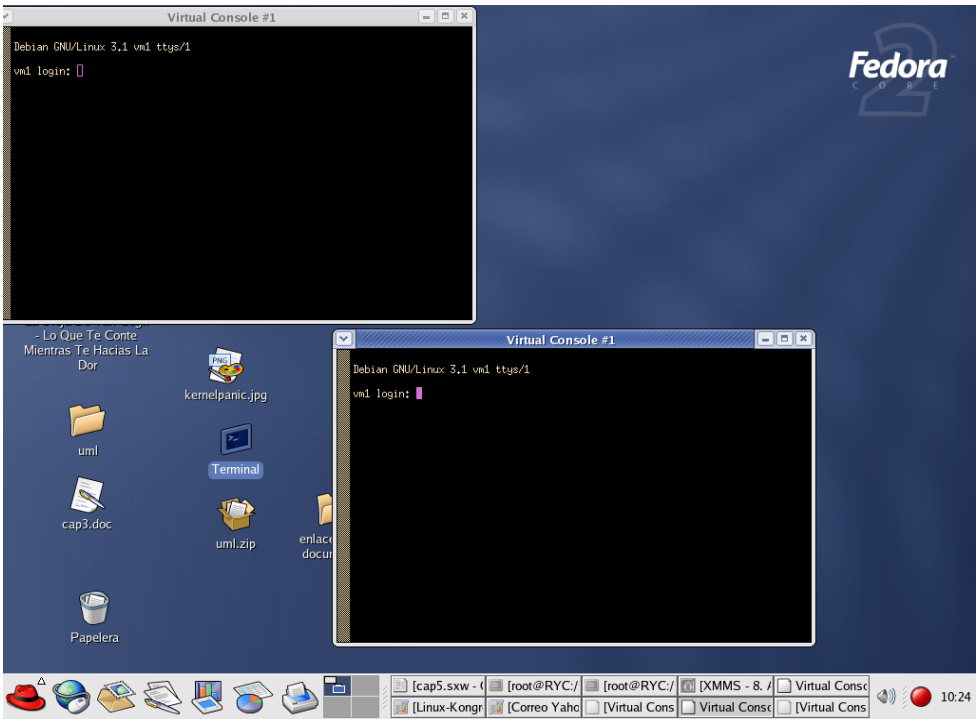


Figura 5. 3: Ejecución de dos instancias de UML utilizando la técnica *Copy On Write*

A continuación se muestran los comandos necesarios para la ejecución de dos instancias virtuales mediante la técnica descrita:

```
[root@Ryc2 linux-2.6.8.1]# ./linux ubd0=root_fs1.cow,root_fs

[root@Ryc2 linux-2.6.8.1]# ./linux ubd0=root_fs2.cow,root_fs
```

## 5.4 Conectando las Instancias UML

*User Mode Linux* presenta una manera conveniente de configurar entornos de redes virtuales, consistentes de múltiples instancias de máquinas virtuales UML ejecutándose sobre una simple máquina física. UML ofrece distintas facilidades de comunicación y dispositivos de red virtuales con los cuales se pueden construir complejos escenarios, aptos para experimentar con aplicaciones de red y protocolos sin perder visos de realidad.

El *daemon uml_switch* provee una manera fácil de crear una red UML conectada vía *switchs ethernet* virtuales. Mientras que TUN TAP es el mecanismo preferido para agregar conectividad desde las redes UML hacia la máquina host anfitrión. TUN TAP puede ser usado para poner las máquinas virtuales UML sobre la misma LAN *ethernet* en la que se encuentra el host. Topologías complejas de red virtuales pueden usar una combinación de esas facilidades de comunicación.

### 5.4.1 SWITCH DAEMON

Es un *switch ethernet* implementado en *software* (uml_*switch*). Internamente este *daemon* provee comunicación utilizando *sockets* de dominio *Unix* [1]. El *daemon* escucha las

conexiones a través de un par de *sockets* y envía paquetes entre las UML conectadas a él. Uml_*switch* puede operar en modo *switch* (los paquetes son enviados solo a la UML específica a la que fueron enviados) o modo *hub* (los paquetes son transmitidos a todas las UML, independientemente de su destino).

La red creada con uml_*switch* es aislada y no tiene conectividad con el mundo exterior (incluida la red del host). El *switch* virtual provisto por uml_*switch* puede ser conectado al *host* añadiéndole a este un dispositivo TAP sobre el host.

A pesar de sus limitaciones uml_*switch* provee una manera fácil de conectar máquinas virtuales, su inconveniente consiste en que si el *switch daemon* que se ejecuta sobre host muere, todas las UML deberán ser reiniciadas.

### 5.4.2 TUN TAP

En los *kernel*s linux TUN TAP provee interfaces de red virtual para conexiones punto a punto (TUN) y conexiones *ethernet* (TAP) [1].

UML utiliza las interfaces TAP para redes *ethernet* virtuales entre las UMLs y el host. TAP provee aplicaciones con dos interfaces:

* /dev/tapX – dispositivo de carácter.

* tapX – interface *ethernet* virtual.

Las aplicaciones pueden escribir frames *ethernet* en /dev/tapX y el *kernel* recibe estos *frames* desde la interface tapX. También, cada frame que se escribe en la interface tapX puede ser leído por las aplicaciones desde el dispositivo /dev/tapX. Hay básicamente dos maneras de establecer un dispositivo TAP con UML. El primero es usar uml_net para dinámicamente crear y configurar un dispositivo TAP sobre el host en tiempo de buteo de UML. El segundo método es crear (manualmente) un dispòsitivo TAP sobre el host de antemano.

Conectando las UML con interfaces TUN TAP y habilitando ip_forward sobre el host este actuará como *router* para las máquinas virtuales.

Para la conectividad entre las máquinas virtuales, todos los paquetes deberían ser ruteados a través del host. Esto puede volverse rápidamente pesado para una red virtual que cuente con un número elevado de máquinas virtuales. Afortunadamente, las UMLs pueden ser conectadas usando uml_*switch* y el dispositivo TAP puede ser conectado a este *switch daemon*.