

Iniciação ao User-mode Linux

Linux no Linux

O User-mode Linux (Linux no espaço do usuário) se parece com Linux porque é Linux. Existem mil maneiras de usar esse sistema Linux virtual rápido e racional; invente a sua.

POR FABRIZIO CIACCHI

O popular e versátil *User-mode Linux* (UML) [1] cria um sistema Linux virtual totalmente operacional numa máquina anfitriã com Linux. O UML tem muitos usos. Os desenvolvedores de programas “perigosos” o empregam para testar seus aplicativos apropriadamente sem pôr em risco o sistema anfitrião. Usuários comuns de Linux usam o UML para fazer experiências com versões do kernel sem ter de se preocupar com um patch novo ou não testado. Os administradores de sistema usam o UML para testar configurações alternativas sem tirar a rede da empresa do ar. É possível até mesmo rodar múltiplas versões do UML na mesma máquina para simular uma rede completa.

O que é User-mode Linux?

O User-mode Linux não é realmente um emulador nem uma API como o *Wine*. O melhor jeito de explicar o User-mode Linux é começar dando uma olhada no papel do kernel.

Cada aplicativo ou programa que você roda dispara um ou mais processos – pense num processo como sendo uma “tarefa” que o programa desempenha. O kernel tem a função

de executar e administrar os processos de todos os programas e “falar” com o hardware. Quando um processo quer se comunicar com um dispositivo (por exemplo, para exibir algo no monitor, enviar dados pela rede, imprimir um documento ou copiar um arquivo

Escolha de emulação

Talvez o melhor modo de compreender as vantagens do UML seja considerar que software desse tipo vem em três formas

- ⇒ Emulação de software
- ⇒ Emulação de hardware
- ⇒ Sem emulação

O Bochs [10] é um dos mais famosos programas emuladores por software. A principal atividade do Bochs é prover a emulação de uma arquitetura de hardware em particular (IA-32, também chamada x86) em cima de um sistema operacional em particular, como Windows®, Mac OS e, obviamente, Linux. Uma vez que o hardware é emulado, é possível instalar qualquer sistema operacional x86 nele (Linux, Windows®, DOS e assim por diante), mas a execução é muito lenta, já que cada instrução do computador precisa ser traduzida do sistema operacional convidado para o sistema anfitrião.

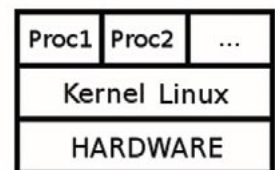


Figura 1: Estrutura de processos normal do Linux.

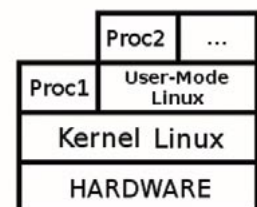


Figura 2: O UML roda como um processo. Nesta imagem, Proc1 está rodando no sistema Linux hospedeiro. Proc2 está rodando no sistema virtual User-mode Linux.

para um disquete), ele pede ao kernel Linux que administre a comunicação com o hardware (figura 1).

O kernel User-mode Linux roda no Linux como um processo do kernel anfitrião. A diferença entre um kernel UML e um kernel comum é que o kernel UML não se comunica diretamente com o hardware. Em vez disso, essa comunicação passa para o kernel “real” da máquina anfitriã, que gerencia a comunicação com o hardware (veja a figura 2).

Uma vez que o sistema virtual e o sistema anfitrião são, ambos, sistemas Linux com estrutura quase idêntica, a comunicação passa de forma muito eficiente do sistema virtual para o anfitrião, demandando pouquíssimo código extra para abstração ou tradução.

A emulação de hardware consiste na reprodução (ou virtualização) da arquitetura nativa de hardware. Esses emuladores são mais eficientes que os emuladores de

software porque usam a infra-estrutura do próprio hardware da máquina hospedeira, mas precisam interceptar todas as chamadas a ele vindas do sistema operacional convidado. Essa solução tem a grande desvantagem de que o código deve ser especializado para uma arquitetura de hardware em particular, que é a mesma para o ambiente do anfitrião e o do convidado. Um exemplo desse tipo de emulador é o *VMware* [2], um programa comercial bastante popular.

O User-mode Linux encaixa-se na última categoria. Ele não precisa emular nenhum hardware específico; em vez disso, fala diretamente com o hardware real. As instruções são passadas com eficácia do kernel UML para o kernel anfitrião. O UML pode executar código nativo e roda, na pior das hipóteses, com uma perda de desempenho de apenas 20% em comparação com o mesmo código sendo rodado no anfitrião.

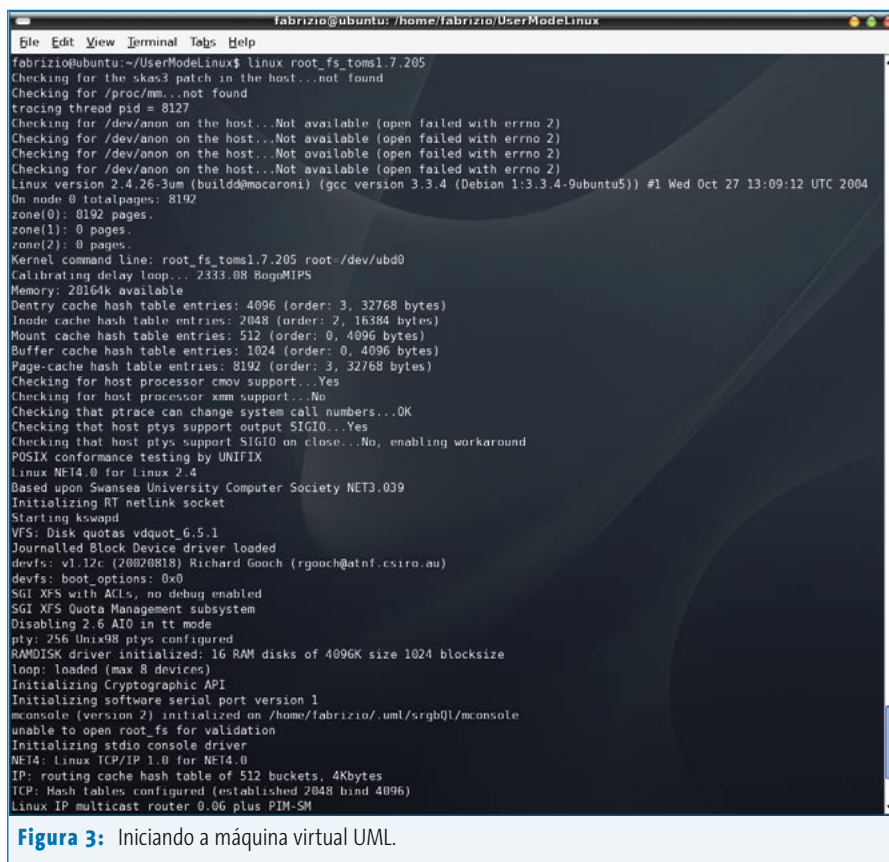


Figura 3: Iniciando a máquina virtual UML.

QEMU: Uma boa alternativa

Se o seu objetivo é usar UML para testar novas distribuições Linux, talvez seja melhor optar pelo emulador de sistema QEMU [10]. Baseado no Bochs [11], o QEMU é muito simples de instalar, configurar e usar. Para saber mais sobre ele, consulte o artigo “Máquinas virtuais: Emulação de sistemas com o QEMU” na edição 8 da Linux Magazine Brasil. Você pode baixar o artigo em PDF de nosso site [12]

Configuração do UML

Instale o User-mode Linux com seu gerenciador de pacotes. Por exemplo, no Debian é preciso digitar o seguinte comando, como root:

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

Esse comando instala o kernel do UML e outros utilitários. Outros gerenciadores de pacotes são igualmente simples, mas se você tiver algum problema ao instalar com um sistema de pacotes ou se tiver problemas de memória durante o boot [3], pode querer baixar um kernel normal (recomendamos a versão 2.4.27 [4]) e o patch para o kernel do UML [5]. Outros patches podem ser encontrados em [6]. Após baixar os arquivos (no mesmo diretório, é claro), abra um terminal e execute os seguintes comandos:

```
$ tar -jxvf linux-2.4.27.tar.bz2
$ 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
```

SELinux no UML

Há um documento que explica como configurar um sistema UML com o subsistema de segurança SELinux [13]. Um sistema UML com SELinux pode ser muito útil para criar servidores mais seguros e testar a política de segurança do SELinux sem pôr em risco um sistema “de verdade”.

Depois de completar esses comandos, você terá um arquivo chamado `linux` em seu diretório original. Esse arquivo é o kernel User-mode Linux, que será usado para iniciar o sistema Linux virtual.

Para fazer com que o UML funcione a contento, é necessário incluir duas outras peças do quebra-cabeças: um sistema de arquivos raiz (uma imagem comprimida de uma partição Linux que contenha todos os programas) e os utilitários do UML. Para o sistema de arquivos raiz, todas as imagens disponíveis podem ser encontradas em [7]. Você pode baixar os utilitários do UML em [8] e digitar os seguintes comandos:

```
$ tar -jxvf uml_utilities_XXXXXXX.tar.bz2
$ cd tools
$ make all
$ make install DESTDIR=/
```

XXXXXXX é número da versão das *UML utilities*. Agora você tem um diretório que contém o sistema de arquivos raiz. Lembre-se de pôr o programa `linux` num local que permita usá-lo (se você não o moveu, ele deve estar no diretório `linux-2.4.27`). Digite os seguintes comandos para ler o sistema de arquivos raiz:

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

O parâmetro `ubd0=` diz à máquina virtual que use o arquivo especificado como sistema de arquivos raiz.

Se tudo correr bem, você verá a máquina virtual iniciando (figura 3) e poderá iniciar uma “sessão virtual” com o nome de usuário `root` e a senha `root`.

Compartilhando o sistema de arquivos raiz.

É possível iniciar duas ou mais máquinas virtuais usando o mesmo sistema de arquivos raiz. O driver `udb0` usa um mecanismo chamado *Copy-On-Write* (COW),

que lê o sistema de arquivos raiz como um dispositivo somente para leitura e armazena as mudanças num arquivo privado de leitura/escrita (o arquivo COW). Se quiser iniciar duas máquinas virtuais (VM1 e VM2) com o mesmo sistema de arquivos, abra duas sessões de terminal e digite os seguintes comandos:

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

Todas as modificações nos dois sistemas virtuais serão escritas nos arquivos COW respectivos. Na verdade, o sistema de arquivos não é compartilhado, mas as duas execuções são independentes. A coisa mais importante a evitar quando os arquivos COW são criados é iniciar o sistema de arquivos diretamente (com `ubd0=root_fs_XXX`), porque todos os arquivos COW registram o tamanho e o *timestamp* (instante em que é montado) do sistema de arquivos raiz e qualquer modificação os tornará inutilizáveis. A sintaxe correta para o reinício seguinte, quando houver um arquivo COW, é esta:

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

Redes reais e virtuais

O UML oferece diversas opções interessantes para trabalhar com redes em sistemas virtuais Linux. Assim que você

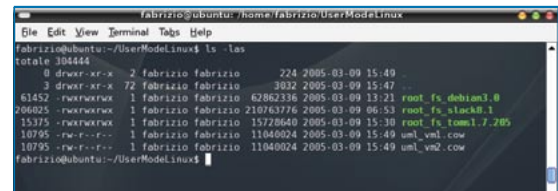


Figura 4: Os arquivos COW das duas máquinas virtuais UML.

tiver seu sistema virtual UML pronto e funcional, pode ter vontade de ligar em rede o sistema virtual com seu hospedeiro ou com outros sistemas virtuais. Há uma descrição detalhada das redes no UML em [9].

A idéia básica por trás das redes no UML é que diversos *transportes* opcionais são oferecidos para administrar a troca de pacotes entre o sistema virtual e o anfitrião. A tabela 1 mostra alguns dos tipos de transporte disponíveis para o UML.

Para habilitar um dispositivo de rede na máquina virtual, digite o texto a seguir na linha de comando do kernel:

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

onde `<n>` representa a interface real do anfitrião (ou seja, `eth0`) à qual a máquina virtual será conectada. A explicação teórica é que, na máquina virtual UML, há um dispositivo `eth0`. Esse dispositivo é, na verdade, um “apelido” para o dispositivo `tap0` no hospedeiro – trocando em miúdos, o `tap0` do hospedeiro e o `eth0` da máquina virtual UML são o mesmíssimo dispositivo. A interface `tap0` (que, lembre-se, está no sistema real) está diretamente conectada à interface `eth0` do anfitrião. Na prática, isso significa que a `eth0` do

Tabela1: Tipos de transportes UML

Etherap, TUN/TAP	Transportes usados para troca de pacotes entre o sistema virtual e o hospedeiro real.
Switch daemon	Transporte projetado para redes puramente virtuais com outros sistemas UML.
Multicast	Outro transporte projetado para redes virtuais.
Slip, slirp	Transportes usados principalmente quando Ethertap e TUN/TAP não estão disponíveis ou se você não tiver acesso de root à configuração de rede do anfitrião.
Pcap	Transporte que oferece uma interface de rede de somente leitura sendo, portanto, uma boa opção para monitoramento de rede.

sistema virtual “falará” com o mundo pela `eth0` do sistema real, embora ambas possam ter números IP diferentes.

Assim, podemos usar o comando:

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

para permitir que o UML configure a interface `eth0` da máquina virtual com seu próprio endereço IP. O endereço IP do dispositivo `tap0` real e do `eth0` virtual pode ser o mesmo para configurações mais simples (veja a referência [9] para saber mais sobre configurações de rede mais complexas).

Você precisa então configurar a interface de rede da máquina virtual, através dos arquivos `/etc/hosts`, `/etc/resolv.conf`, `/etc/network`, etc. (exatamente como numa máquina real) para ter acesso à Internet plenamente funcional dentro do ambiente UML.

Conclusão

O User-mode Linux oferece um modo rápido e conveniente de criar sistemas virtuais no Linux. O UML pode ser usado como uma ferramenta para planejar, modelar, testar e resolver problemas em sistemas Linux. O UML também é a base de muitos outros projetos e experimentos, assim como aplicações de negócios e serviços de hospedagem personalizados. Talvez o UML não seja lá muito fácil de instalar e configurar, mas se você conseguir fazer com que funcione, descobrirá muitos usos para ele. ■

SOBRE O AUTOR

Fabrizio Ciacchi (fabrizio.ciacchi.it) – fabrizio@ciacchi.it estuda Ciência da Computação na Universidade de Pisa. Suas atividades principais são estudar Linux, desenvolver sites em PHP e programar em Java. Trabalha também como consultor para diversas empresas e escreve artigos sobre Linux.

INFORMAÇÕES

- | |
|--|
| [1] Site oficial do User-mode Linux: user-mode-linux.sourceforge.net |
| [2] Página oficial do VMWare: www.vmware.com |
| [3] UML em máquinas 2G/2G: user-mode-linux.sourceforge.net/UserModelLinux-HOWTO-4.html#2G-2G |
| [4] Kernel oficial do Linux versão 2.4.27: ftp://ftp.ca.kernel.org/linux/kernel/v2.4/linux-2.4.27.tar.bz2 |
| [5] Patch do UML para o kernel 2.4.27: prdownloads.sourceforge.net/user-mode-linux/uml-patch-2.4.27-1.bz2 |
| [6] Downloads do UML: user-mode-linux.sourceforge.net/dl-sf.html |
| [7] Arquivos de imagem com sistemas raiz: user-mode-linux.sourceforge.net/dl-jails-sf.html |
| [8] Utilitários do UML: prdownloads.sourceforge.net/user-mode-linux/uml_utilities_20040406.tar.bz2 |
| [9] Configuração de rede no UML: user-mode-linux.sourceforge.net/networking.html |
| [10] Site oficial do Bochs: bochs.sourceforge.net |
| [11] Site oficial do QEMU: fabrice.bellard.free.fr/qemu/ |
| [12] Artigo “Benefícios Virtuais” sobre o QEMU (em inglês): www.linux-magazine.com/issue/52/QEMU_System_Emulation.pdf |
| [13] SELinux e UML: www.golden-gryphon.com/software/security/selinux-uml.xhtml |
| [14] Compilação do kernel: user-mode-linux.sourceforge.net/compile.html |
| [15] Depuração no kernel: user-mode-linux.sourceforge.net/debugging.html |
| [16] Depurando o UML: user-mode-linux.sourceforge.net/debug-session.html |