ВЛАДИМИР «TURBINA» ЛЯШКО
(V.TURBINA@GMAIL.COM)

Виртуальный полигон

ЭМУЛИРУЕМ АППАРАТНОЕ ОБЕСПЕЧЕНИЕ РАЗЛИЧНЫХ ПЛАТФОРМ С ПОМОЩЬЮ QEMU

С ростом мощностей компьютеров тема виртуализации становится все популярнее. На одном компьютере можно без проблем создавать целые виртуальные сети, запуская несколько копий ОС. Это полезно не только для тестирования или обучения, но и в обычной работе. В случае сбоя или хакерской атаки вернуть виртуальную систему в исходное состояние очень просто.

ВОЗМОЖНОСТИ QEMU

QEMU относится к программам, эмулирующим аппаратную среду. Основные функции аналогичны именитым VMWare, VirtualBox, Bochs или Virtual PC, хотя некоторые возможности отличаются. Например, поддерживается два вида эмуляции:

• **Full system emulation** — создается полноценная виртуальная машина, имеющая «свой» процессор и различную периферию;

• **User mode emulation** — режим, поддерживаемый только в Linux. Он позволяет запускать на родном процессоре программы, откомпилированные под другую платформу.

Во втором варианте QEMU берет на себя всю заботу о переводе инструкций процессора и конвертации системных вызовов. Благодаря быстрому и компактному динамическому транслятору кода, достигается высокая скорость эмуляции. В этом режиме возможна эмуляция не только x86, но и процессоров других архитектур: ARM, SPARC, PowerPC, MIPS и m68k. К списку полной эмуляции добавим еще x86_64 и EM64T. Работает QEMU на Linux, FreeBSD, Mac OS X, FreeDOS и Windows (www.h7.dion.ne.jp/~qemu-win). В качестве основной платформы можно использовать компьютеры на базе x86, x86_64 и PowerPC. Впрочем, ограниченно поддерживаются и некоторые другие (DEC Alpha, SPARC32, ARM, S390). Виртуальная машина i386-архитектуры, созданная при помощи QEMU, получает в свое распоряжение следующий набор устройств:

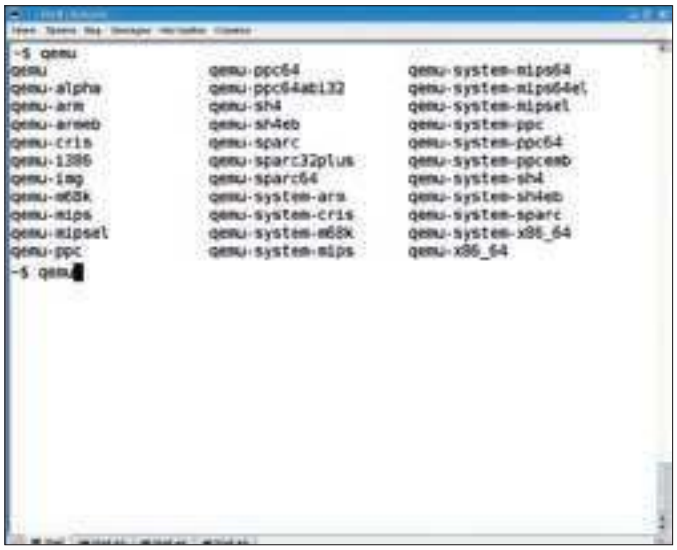
- процессор такой же частоты, как и на основной системе; в SMP-системах возможна работа до 255 CPU (по умолчанию 1 CPU);
- PC BIOS, используемый в проекте Bochs;
- материнская плата i440FX с PIIX3 PCI — ISA мостом;
- видеокарта Cirrus CLGD 5446 PCI VGA или VGA карта с Bochs VESA расширениями;
- мышь PS/2 и клавиатура;
- 2 PCI IDE интерфейса для жесткого диска и поддержку CD-ROM;
- два дисковода;
- до 6 NE2000 PCI сетевых карт;
- до 4 последовательных (COM) портов;
- вывод звука через Soundblaster 16 совместимую карту.

В последних версиях появилась долгожданная поддержка USB и улучшен звук. Также стало возможно сетевое соединение между эмулируемыми ОС.

По сравнению с другими известными виртуальными машинами, QEMU работает достаточно шустро. Но есть еще модуль QEMU Accelerator Module (KQEMU), позволяющий выполнять часть кода напрямую на реальном процессоре, минуя виртуальный. Это неплохо ускоряет работу гостевой системы. Без этого модуля запуск виртуальной ОС замедляется примерно в пять раз.

Сейчас KQEMU доступен для ядер Linux-версий 2.6.x и 2.4.x (работает только на x86 и x86_64). Есть порты под FreeBSD и Windows, но они недостаточно развиты. Ранее KQEMU распространялся по проприетарной лицензии с закрытым исходным кодом и только для Linux. Сегодня его код открыт под GNU GPL, как и прочие компоненты QEMU.

Кроме того, в рамках проекта Linux KVM (Kernel-based Virtual Machine) полным ходом идет разработка поддержки технологий аппаратной виртуализации (Intel VT и AMD SVM) для x86-процессоров Intel и AMD. Пока это патчи, позволяющие QEMU использовать возможности KVM. В будущем поддержка KVM будет реализована в основной ветке QEMU.



Для эмуляции других архитектур вызывается отдельная утилита

Виртуальная машина в работе

УСТАНОВКА QEMU

Всем хорош QEMU, но есть один недостаток — документация проекта больше рассчитана на разработчиков. Нормальных инструкций для пользователя так мало, что их можно пересчитать по пальцам одной руки. Особенно учитывая, что в последних версиях изменился процесс установки и работы (хотя и незначительно). На странице закачки предлагаются исходные тексты, бинарная сборка для Linux и ссылки на пакеты для RHEL/Fedora и Slackware, плюс порты Windows, OpenSolaris, Mac OS X и готовые образы систем. Для примера установим QEMU в Ubuntu. Но все сказанное будет актуально для Debian и в некоторой мере для других дистрибутивов. Проверяем, что доступно в репозитории:

```
$ sudo apt-get update
$ sudo apt-cache search qemu
```

Если подключен Universe, — в ответ получаем список приложений, в котором присутствует как сам эмулятор, так и некоторые утилиты для работы с ним. Смотрим, что за версию предлагают:

```
$ sudo apt-cache show qemu | grep -i version
Version: 0.9.1-1ubuntu1
```

На момент написания статьи это был самый последний релиз. Инсталлируем:

```
$ sudo apt-get install qemu
```

Появится QEMU и при установке пакета kvm. Список зависимостей можно просмотреть командой «`sudo apt-cache depends qemu`». Мы же будем собирать его самостоятельно. Установка ускорителя KQEMU вынесена в отдельную операцию — как при установке при помощи пакетов, так и при сборке из исходников. В первом случае вводим:

```
$ sudo apt-get install kqemu-common
```

При сборке с помощью пакетов проще не искать зависимости вручную:

```
$ sudo apt-get build-dep qemu
```

У меня было установлено 28 зависимостей. Кроме этого, эмулятор требует для работы еще несколько пакетов:

```
$ sudo apt-get install bochsbios libasound2 libc6 \
      libncurses5 libsdl1.2debian vgabios zlib1g
```

В списке рекомендуемых также значатся debootstrap, openbios-sparc, openhackware, proll, sharutils и vde2. Теперь качаем с сайта проекта последние версии эмулятора и ускорителя и распаковываем архив с qemu во временный каталог и в подкаталог архив с kqemu.

```
$ tar xzvf qemu-0.9.1.tar.gz
$ cd qemu-0.9.1/
$ tar xzvf ../kqemu-1.4.0pre1.tar.gz
```

Если в системе присутствует старая версия эмулятора с модулем kqemu, его желательно выгрузить:

```
$ sudo rmmod kqemu
```

Конфигурируем:

```
$ ./configure
```

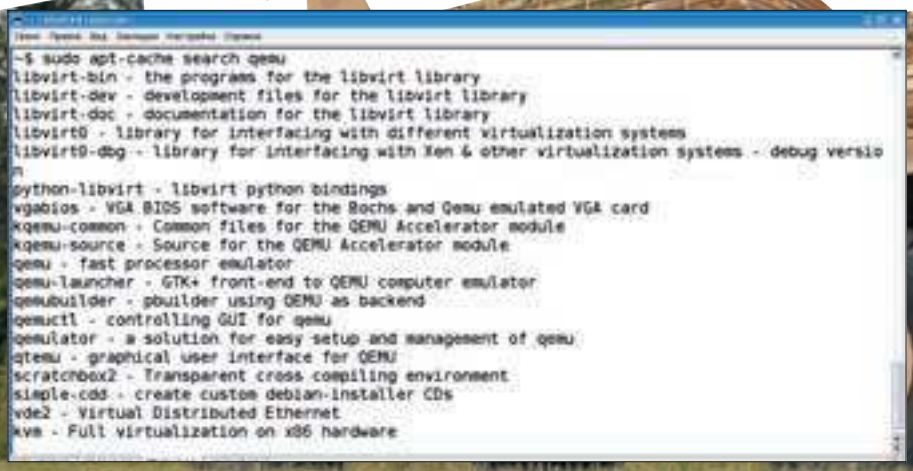
В результате получаем большой список параметров сборки. Среди них важны «`SDL support yes`», который означает возможность запуска в графическом режиме, и «`kqemu support yes`» — это поддержка ускорителя. Теперь вводим «`make`», а затем: «`sudo make install`». Ускоритель kqemu нужно собирать отдельно. Переходим в подкаталог с kqemu и повторяем все сначала:

```
$ cd kqemu-1.4.0pre1
$ ./configure
```

Если ошибок нет, то дальше выполняем обычную установку. Единственный момент, который способен вызвать проблему, — отсутствие заголовочных файлов действующего ядра. Поэтому если в ответ получаешь «`kqemu cannot be compiled on your system`», доустанови хэдеры. По какой-то причине команда «`sudo apt-get build-dep kqemu-common`» выдает запрос на установку только одного пакета dpatch. То есть, заголовочные файлы ядра в зависимостях не устанавливаются, поэтому вводим:

```
$ sudo apt-get install kernel-headers-$(uname -r)
```

И — повторяем процедуру установки. Загрузить модуль не со «своим» ядром не получится, в ответ будет выдано сообщение «`Invalid module format`».
Есть еще один нюанс. Если ранее по запросу был установлен gcc-3.4, то попытка выполнить «`sudo apt-get install kqemu-source`» при-



Ищем QEMU в репозитарии Ubuntu

В состав QtEmu входит простой и понятный мастер создания VM

links

• Сайт проекта находится по адресу www.qemu.org.

• В репозитариях дистрибутивов и на freshmeat.net можно найти несколько решений, упрощающих создание виртуальных машин.

• Запуск Linux для процессоров ARM в окружении QEMU: dreamcatcher.ru/docs/linux_arm_qemu.html.

ведет к тому, что в системе появится еще и gcc-4.1, который и будет использован для сборки модуля. Проблем это не вызывает, но все же.
Пробуем загрузить модуль:

```
$ sudo modprobe kqemu
```

Чтобы впредь не загружать его вручную, добавь «kqemu» в /etc/modules.
Возможно, в некоторых дистрибутивах, использующих UDEV, к команде для запуска модуля понадобится добавить параметр «major=0»:

```
$ sudo modprobe kqemu major=0
```

А чтобы изменить параметры доступа, используй скрипты настройки UDEV. Например, в Fedora заносим файл /etc/udev/permissions.d/50-udev.permissions строчку «kqemu:root:root:0666».

РАБОТА С QEMU

Запустить LiveCD-дистрибутив очень просто. Достаточно вставить диск в привод и ввести команду:

```
$ qemu -m 512 -cdrom /dev/cdrom
```

Для виртуальной машины я выделил 512 Мб (по умолчанию — 128 Мб, но этого не всегда хватает). Через некоторое время появится новое окно, в котором будет запущена ОС.

Чтобы управлять виртуальной системой, щелкаем мышкой внутри окна. Выйти можно, нажав <Ctrl+Alt>.
Если есть ISO-образ, то можно подключить и его:

```
$ qemu -m 512 -cdrom TinyMe-2008.0.i586.iso
```

Если при запуске эмулятора будет выдаваться сообщение о неактивности модуля kqemu: «Could not open /dev/kqemu — QEMU acceleration layer not activated: Permission denied», то потребуется изменение прав на файл устройства /dev/kqemu (по умолчанию 660):

```
$ sudo chmod 666 /dev/kqemu
```

Кстати, раньше нужно было создавать этот файл вручную при помощи команды «mknod /dev/kqemu c 250 0». Теперь в этом нет необходимости. В некоторых системах эмулятор при запуске может потребовать перестроить параметры таймера высокого разрешения — «Could not configure /dev/rtc to have a 1024 Hz timer...». Тогда выполняем команду:

```
$ sudo sh -c "echo 1024 > /proc/sys/dev/rtc/max-user-freq"
```

В Ubuntu 8.04 по умолчанию его значение равно 64, но QEMU никогда не жаловался. Идем дальше. Для установки гостевой ОС сначала нужно создать виртуальный диск:

```
$ qemu-img create test-disk 4G
```

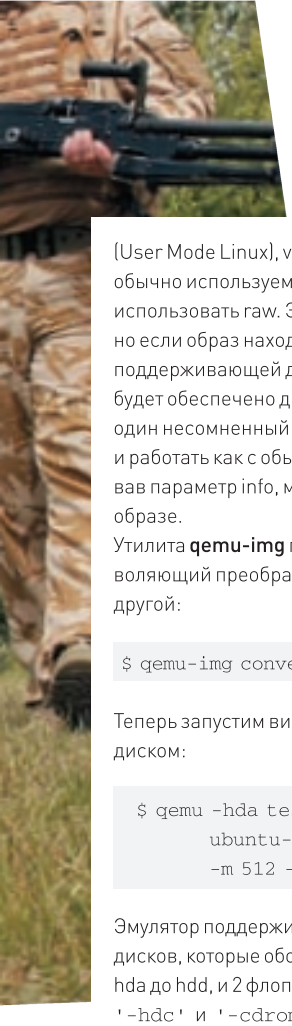
Хотя можно сделать это и при помощи dd:

```
$ dd of=test-disk bs=1024 seek=4194304 count=0
```

Правда, есть отличие: утилита dd позволяет создать только raw-образ, который представляет собой файл, заполненный нулями. Утилита qemu-img поддерживает несколько форматов, указать на которые можно при помощи параметра '-f'. По умолчанию создаются qcow-файлы (qemu Copy On Write). Этот формат поддерживает шифрование (AES, 128 бит) и компрессию, но возможны еще raw, cow

Консоль управления QEMU

QEMU предоставляет возможность управлять своей работой и получать информацию из специальной консоли. Чтобы ее вызвать, нажми комбинацию <Ctrl+Alt+2>. Теперь, используя разные команды (полный список которых доступен по help), можно добавить новое устройство, когда ОС уже загружена, сохранить (savevm имя_файла) или загрузить (loadvm) состояние виртуальной машины — и многое другое. Например, чтобы добавить CD-ROM, достаточно ввести «change cdrom /dev/cdrom». Использовав параметр info, мы узнаем о режиме или состоянии того или иного компонента. Например, «info kqemu» выведет режим, в котором находится модуль. Обратно в гостевую ОС можно вернуться, нажав <Ctrl+Alt+1>.



(User Mode Linux), vmdk (VMWare) или cloop (сжатый loop, обычно используемый на LiveCD). Многие предпочитают использовать raw. Этот формат не поддерживает сжатие, но если образ находится на разделе с файловой системой, поддерживающей дыры (holes), например ext2/3, то сжатие будет обеспечено драйвером ФС. И у этого способа есть еще один несомненный плюс — можно монтировать в дерево ФС и работать как с обычным дисковым разделом. Использовав параметр info, можно получить информацию о готовом образе.

Утилита **qemu-img** поддерживает параметр convert, позволяющий преобразовывать образы из одного формата в другой:

```
$ qemu-img convert – f cow cowimage.cow image.raw
```

Теперь запустим виртуальную машину уже с жестким диском:

```
$ qemu -hda test-disk -cdrom \
      ubuntu-8.04-desktop-i386.iso\
      -m 512 -boot d -localtime
```

Эмулятор поддерживает до четырех виртуальных жестких дисков, которые обозначаются аналогично линуксовым от hda до hdd, и 2 флоппи-диска — fda и fdb. Но использовать `'-hdc'` и `'-cdrom'` одновременно нельзя. Если используется только один образ диска, параметр hda можно опустить:

```
$ qemu test-disk
```

Параметр `'-boot'` так же, как и в реальной машине, позволяет указать приоритет загрузки. Доступно четыре варианта:

```
• boot a – загрузка с виртуального флоппи;
• boot c – загрузка с жесткого диска (по умолчанию);
• boot d – загрузка с CD-ROM;
• boot n – сетевая (Etherboot) загрузка.
```

Параметр `'-localtime'` позволяет указать на использование локального времени в виртуальной машине.

Теперь обычным образом устанавливаем операционную систему на жесткий диск и после перезагрузки используем уже:

```
$ qemu test-disk -m 512 -localtime
```

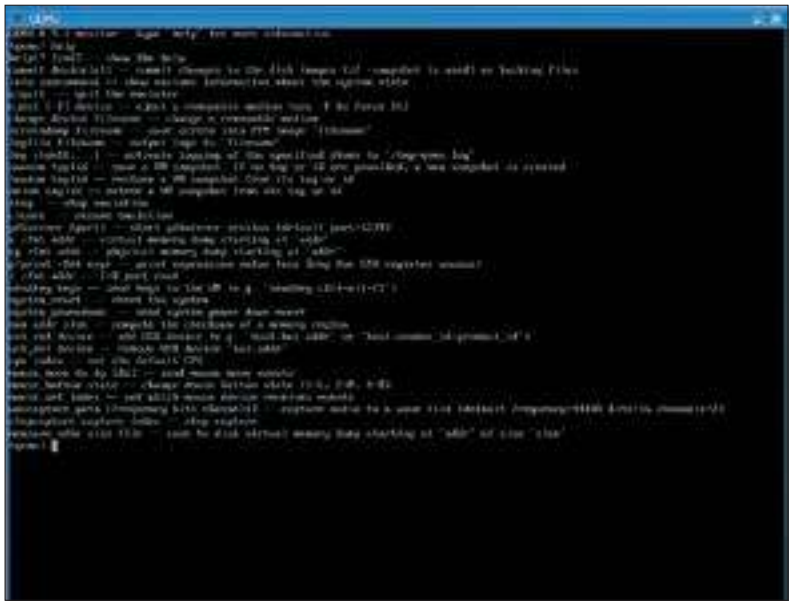
Чтобы виртуальная система стартовала сразу в полноэкранном режиме, добавь ключ `'-full-screen'`; переключение производится при помощи комбинации <Ctrl+Alt+F>. В некоторых гостевых ОС, возможно, потребуется отключить ACPI флагом `'-no-acpi'`.

В зависимости от установок родительской ОС, в процессе запуска иногда появляется сообщение о том, что эмулятор не может получить доменное имя. Самым простым выходом будет добавить опцию `'-dummy-net'`, активирующую поддельный сетевой стек. Но при этом гостевой системой не будут приниматься и отправляться пакеты:

```
$ qemu -dummy-net -cdrom /dev/cdrom
```

ПРОДВИНУТЫЕ ВОЗМОЖНОСТИ

По умолчанию эмулируется одно ядро. Чтобы увеличить число процессоров, используем параметр `'-smp'` с указанием их количества. Правда, в некоторых случаях это приводит к замедлению эмуляции, да и пытаться создать несколько виртуальных процессоров на компьютере с одним CPU бессмысленно.



Консоль управления QEMU

При запуске qemu эмулирует ту же аппаратную среду, в которой он запускается. При запуске на i386 будет «подражать» i386, на x86_64 — 64-битной системе. На PowerPC будет запущен еще один PowerPC-компьютер и т.д. Когда не обходима эмуляция системы отличной архитектуры, то запускаем специальную версию утилиты. Доступные варианты можно найти, набрав в консоли qemu и нажав табуляцию в bash (qemu-system-ppc, qemu-system-sparc и другие). Помимо стандартных PC и ISA PC (без шины PCI), QEMU может эмулировать и другие аппаратные платформы, несвязанные с персональным компьютером — такие, как APM Versatile или платы на основе MIPS. Вывести полный список поддерживаемых платформ можно при помощи ключа `'-M ?'`. Чтобы изменить платформу pc на ISA-only PC, достаточно набрать:

```
$ qemu -M isapc -hda test-disk -m 512
```

По умолчанию звуковая система не активируется. Придется это сделать самостоятельно, добавив `'-enable-audio'`. Получить список поддерживаемых аудиоподсистем можно при помощи параметра `'-audio-help'`, а список звуковых карт — `'-soundhw ?'`. Самое простое — это активировать все звуковые драйверы:

```
$ qemu -soundhw all -hda test-disk
```

Модуль kqemu может работать в двух режимах: «`for user code`» и «`for user and kernel code`». Первый режим устанавливается по умолчанию, и его использование проблем не вызывает (если при запуске ОС в этом режиме возникли проблемы — можно, чтобы не выгружать модуль kqemu, просто отказаться от его использования при помощи параметра `'-no-kqemu '`). Второй режим — более быстрый и активируется при помощи ключа `'-kernel-kqemu '`. Но учти, с некоторыми гостевыми ОС он не дружит. Кроме того, скорость работы зависит от версии ядра гостевой системы и нескольких других параметров.

Для поднятия виртуального сетевого tap/tun-интерфейса (в

info

Автор QEMU — французский программист Фабрис Беллар, создатель популярной библиотеки libavcodec, на базе которой были созданы такие программы, как FFmpeg, ffdshow, MPlayer, VideoLAN и др.

• QEMU может работать **в двух режимах**: как полноценная виртуальная машина или как машина, позволяющая запускать программы, откомпилированные под другую платформу.

• QEMU поддерживает управление по протоколу VNC.

ядре должен быть включен параметр CONFIG_TUN) qemu по умолчанию использует скрипт `/etc/qemu-ifup`. Если таковой не обнаруживается, то эмулятор самостоятельно выбирает параметры. В простейшем случае скрипт `/etc/qemu-ifup` выглядит так:

```
#!/bin/sh
sudo /sbin/ifconfig $1 192.168.0.100
```

Теперь делаем скрипт исполняемым (`chmod +x`) и запускаем эмулятор:

```
$ qemu test-disk -net nic,vlan=0 -net tap,vlan=0
```

Первая часть команды (`-net nic,vlan=0`) создаст сетевую карту в виртуальной машине, подключив ее к виртуальной сети 0, вторая (`-net tap,vlan=0`) поднимет tap-интерфейс на хост компьютера и подключит его к виртуальной сети 0. Адрес для tap-интерфейса будет взят из `/etc/qemu-ifup`. Адрес сетевой карты настраивается стандартными средствами гостевой ОС и должен находиться в той же подсети, что и tap (например, 192.168.0.101). Аналогичным образом можно добавить любое количество сетевых карт. Параметр `'-macaddr'` позволяет задать MAC-адрес первого сетевого интерфейса. MAC-адреса остальных будут инкрементированы автоматически.

Если на основной системе установлен Samba-сервер, то гостевая система может общаться с основной через расшаренные ресурсы. Для этого используется замечательная опция `'-smb'` с указанием каталога:

```
$ qemu test-disk — smb /mnt/qemu -net nic,vlan=0 -net
tap,vlan=0
```

Аналогично можно активировать и встроенный tftp-сервер, добавив при запуске команду «`-tftp каталог`». При этом все файлы, находящиеся в указанном каталоге, могут быть загружены на гостевую систему. Обменяться информацией между основной и гостевой системами можно и через перенаправление. Формат такой: «`-redir [tcp|udp]:host-port:[guest-host]:guest-port`». Запускаем эмуляцию с этой опцией:

```
$ qemu test-disk -redir tcp:1234::23
```

Теперь пробуем подключиться к telnet-порту на гостевой системе:

```
$ telnet localhost 1234
```

ПРОСТО И ФУНКЦИОНАЛЬНО

Как видишь, QEMU достаточно простая в использовании и многофункциональная система виртуализации, позволяющая эмулировать системы различных архитектур и запускать приложения, собранные под другие операционные системы. Она не требует предварительной настройки и подготовки, а сам процесс от компиляции до запуска занимает минимум времени.

Qemulator — удобная программа для работы с QEMU

Работа через KVM

QEMU может использовать для работы KVM. При этом не требуется kqemu и наблюдается хорошая производительность. Для начала проверим поддержку этой технологии ядром:

```
$ egrep '^flags.*(vmx|svm)' /proc/cpuinfo
```

В современных дистрибутивах она обычно присутствует, так что делать ничего не придется. Устанавливаем пакет kvm (qemu уже есть). Загружаем нужный драйвер. У меня AMD, поэтому:

```
$ sudo modprobe kvm-amd
```

Если проц от Intel, то:

```
$ sudo modprobe kvm-intel
```

А дальше уже создаем виртуальные машины обычным образом.

Графические тулзы

Управление QEMU производится исключительно из командной строки. Упростить задачу можно при помощи скриптов, записав все команды в файл. В Сети реально найти десяток проектов, предлагающих различные интерфейсы. Название одного из проектов совпадает с именем модуля — **KQEMU** (kqemu.sf.net). С его помощью можно легко настроить виртуальный ПК, просто выбирая нужное из меню, как это делается в VMWare. В KQEMU доступен выбор и монтирование дисков, настройка сети, создание скриптов для запуска настроенной виртуальной машины из командной строки. Еще одно решение — **QtEmu** (qtemu.org) — построено на Qt-библиотеках. Оно будет полезно тем, кто хочет, не рискуя, протестировать новую ОС. В **Qemulator** (qemulator.createweb.de) доступен удобный мастер создания виртуальных машин. Есть в этом списке и веб-интерфейс, предлагаемый проектом **Qemudo** (qemudo.sf.net). С его помощью можно создавать машины и удаленно управлять многочисленными VM.

Эти и некоторые другие приложения есть в репозитариях дистрибутивов. Например, можно установить несколько решений, введя в Ubuntu:

```
$ sudo apt-get install qemu-launcher qtemu qemulator qemuctl
```