

MINIX3 в QEMU

Цилюрник О.И.

Редакция 2.04

от 11.11.2009

Оглавление

Установка QEMU.....	1
Установка MINIX3.....	7
Загрузка с CD.....	7
Установка на диск.....	8
«Реальный» MINIX3 в QEMU.....	9
Оживляем сетку.....	11
Удалённый доступ.....	14
Дополнительные источники информации.....	16

Заглавие этого текста выглядит как-то не самым удачным образом, ... как «заяц в белом вине», но разговор пойдёт об установке операционной системы MINIX3 под виртуальной машиной QEMU, и не только об установке, но и настройке, и использовании некоторых не совсем очевидных особенностей. Все показанные действия будут выполняться в операционной системе Linux, для определённости — CentOS 5.2 (Fedora core 6), всё тоже самое будет сохраняться для дистрибутивов, использующих инсталлятор `yum` и пакетную систему `rpm` (Fedora, RedHat); в других дистрибутивах Linux всё будет практически тоже, с учётом отличий инсталляции пакетов применяемым в дистрибутиве системой инсталляций.

Но сначала краткие предварительные замечания относительно типографской разметки текста: что и как будет выглядеть. Некоторые действия (установка программ и др.) требуют выполнения команд с правами `root`. Далее по тексту, будут по возможности тщательно указываться необходимость прав `root` при выполнении команды — использованием значка приглашения в записи команды: `#` - только `root` полномочия, `$` - не требует `root` привилегий.

Примечание: Выполнение отдельной команды с правами `root` будет иногда показываться в записи:

```
$ su -c 'команда'
```

что то же самое (почти то же самое — отличие в переменных окружения, в путях поиска файлов), что и:

```
# команда
```

Все приводимые в тексте консольные команды проверялись дополнительно непосредственно перед написанием текста, записи выполнения консольных команд приводятся прямым (без редактирования) копированием с терминала. Такой текст (скопированный с терминала) будет показываться моноширинным шрифтом. Кроме того, в большинстве случаев (особо это касается ввода многострочных команд, где отдельные строки завершаются «\») пользовательский ввод в записи команды будет показан жирным шрифтом, а ответный вывод от системы — обычным. Короткие цитаты из различных источников информации будут показываться курсивом.

Вывод команд приведен, возможно, несколько с избыточными подробностями, но это сделано сознательно, для возможностей сравнения ожидаемого и получаемого результатов при воспроизведении, особенно при изменениях версий используемых команд.

Установка QEMU

Начнём с предположение, что виртуальная машина QEMU ещё не установлена в нашем Linux. Можно устанавливать её из исходных кодов, но я покажу инсталляцию бинарных пакетов из RPM-формата, а инсталляцию из исходных кодов мы увидим позже, при доустановке акселератора к QEMU (установку QEMU можно произвести так же и из того же источника). Ищем доступный репозиторий и нужные пакеты в нём:

```
# yum list qem*
```

```
...
```

```

Available Packages
qemu.i386                2:0.10.5-1.el5.2      epel
qemu-common.i386        2:0.10.5-1.el5.2      epel
qemu-img.i386           2:0.10.5-1.el5.2      epel
qemu-system-arm.i386    2:0.10.5-1.el5.2      epel
qemu-system-cris.i386   2:0.10.5-1.el5.2      epel
qemu-system-m68k.i386   2:0.10.5-1.el5.2      epel
qemu-system-mips.i386   2:0.10.5-1.el5.2      epel
qemu-system-ppc.i386    2:0.10.5-1.el5.2      epel
qemu-system-sh4.i386    2:0.10.5-1.el5.2      epel
qemu-system-sparc.i386  2:0.10.5-1.el5.2      epel
qemu-system-x86.i386    2:0.10.5-1.el5.2      epel
qemu-user.i386          2:0.10.5-1.el5.2      epel

```

Примечание: здесь показан результат на моём компьютере. Он может оказаться заметно шире, чем у вас, потому как использовался поиск в одном из дополнительно подключенных репозитариев yum, а yum показывает из найденных результатов самую последнюю найденную версию. Посмотреть текущие подключенные к yum репозитарии можно:

```

# yum repolist
...
repo id                repo name                status
addons                 CentOS-5 - Addons        enabled
adobe-linux-i386       Adobe Systems Incorporated  enabled
base                   CentOS-5 - Base          enabled
epel                   Extra Packages for Enterprise Linux 5 -  enabled
extras                 CentOS-5 - Extras        enabled
livna                  rpm.livna.org for 5 - i386  enabled
planetccrma           Planet CCRMA 5 - i386     enabled
planetcore             Planet CCRMA Core 5 - i386  enabled
updates               CentOS-5 - Updates        enabled

```

В принципе, стандартно определённого при установке Linux репозитария base достаточно для наших целей: из всех показанных пакетов **необходимым** будет только qemu.i386 (остальные будут **достаточными** для виртуализации не только OS, но и архитектуры процессора); отличия будут только в представленной версии, вот результат выполнения на другом компьютере:

```

# yum list qem*
...
Available Packages
qemu.i386                qemu.i386 0:0.9.0-4      base

```

Но и этого достаточно для наших целей! Если же вам захочется добавить репозитарий epel к списку репозитариев yum, то делается это так:

```

# rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-3.noarch.rpm

```

```

Загружается http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-3.noarch.rpm
предупреждение: /var/tmp/rpm-xfer.lBRu2J: Заголовок V3 DSA signature: NOKEY, key ID
217521f6

```

```

Подготовка... ##### [100%]
  1:epel-release          ##### [100%]

```

Продолжаем установку QEMU, устанавливаем **всё**, не жалея места:

```
# yum install qemu*
```

```
...
```

```
Dependencies Resolved
```

```
=====
```

Package	Arch	Version	Repository	Size
---------	------	---------	------------	------

```
=====
```

Installing:

qemu	i386	2:0.10.5-1.el5.2	epel	14 k
------	------	------------------	------	------

Installing for dependencies:

pulseaudio-libs	i386	0.9.10-1.el5.3	epel	166 k
qemu-common	i386	2:0.10.5-1.el5.2	epel	209 k
qemu-img	i386	2:0.10.5-1.el5.2	epel	103 k
qemu-system-arm	i386	2:0.10.5-1.el5.2	epel	803 k
qemu-system-cris	i386	2:0.10.5-1.el5.2	epel	438 k
qemu-system-m68k	i386	2:0.10.5-1.el5.2	epel	498 k
qemu-system-mips	i386	2:0.10.5-1.el5.2	epel	2.6 M
qemu-system-ppc	i386	2:0.10.5-1.el5.2	epel	2.4 M
qemu-system-sh4	i386	2:0.10.5-1.el5.2	epel	911 k
qemu-system-sparc	i386	2:0.10.5-1.el5.2	epel	747 k
qemu-system-x86	i386	2:0.10.5-1.el5.2	epel	1.4 M
qemu-user	i386	2:0.10.5-1.el5.2	epel	4.7 M

```
Transaction Summary
```

```
=====
```

Install	13 Package(s)
Update	0 Package(s)
Remove	0 Package(s)

```
Total download size: 15 M
```

```
Is this ok [y/N]: y
```

```
Downloading Packages:
```

(1/13): qemu-common-0.10.	100%	=====	209 kB	00:07
(2/13): qemu-system-x86-0	100%	=====	1.4 MB	00:49
(3/13): qemu-system-m68k-	100%	=====	498 kB	00:16
(4/13): qemu-system-cris-	100%	=====	438 kB	00:14
(5/13): qemu-img-0.10.5-1	100%	=====	103 kB	00:03
(6/13): qemu-system-ppc-0	100%	=====	2.4 MB	01:23
(7/13): qemu-system-mips-	100%	=====	2.6 MB	01:28
(8/13): qemu-system-sh4-0	100%	=====	911 kB	00:30
(9/13): qemu-0.10.5-1.el5	100%	=====	14 kB	00:00
(10/13): pulseaudio-libs-	100%	=====	166 kB	00:05
(11/13): qemu-system-spar	100%	=====	747 kB	00:24
(12/13): qemu-user-0.10.5	100%	=====	4.7 MB	02:49
(13/13): qemu-system-arm-	100%	=====	803 kB	00:26

```
Running rpm_check_debug
Running Transaction Test
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
```

```
Installing: qemu-common ##### [ 1/13]
Installing: pulseaudio-libs ##### [ 2/13]
Installing: qemu-system-arm ##### [ 3/13]
Installing: qemu-system-sparc ##### [ 4/13]
Installing: qemu-system-sh4 ##### [ 5/13]
Installing: qemu-system-mips ##### [ 6/13]
Installing: qemu-system-ppc ##### [ 7/13]
Installing: qemu-system-cris ##### [ 8/13]
Installing: qemu-system-m68k ##### [ 9/13]
Installing: qemu-system-x86 ##### [10/13]
Installing: qemu-user ##### [11/13]
Installing: qemu-img ##### [12/13]
Installing: qemu ##### [13/13]
```

```
Unable to look at what's on dbus
```

```
Installed: qemu.i386 2:0.10.5-1.el5.2
```

```
Dependency Installed: pulseaudio-libs.i386 0:0.9.10-1.el5.3 qemu-common.i386
2:0.10.5-1.el5.2 qemu-img.i386 2:0.10.5-1.el5.2 qemu-system-arm.i386 2:0.10.5-1.el5.2
qemu-system-cris.i386 2:0.10.5-1.el5.2 qemu-system-m68k.i386 2:0.10.5-1.el5.2 qemu-
system-mips.i386 2:0.10.5-1.el5.2 qemu-system-ppc.i386 2:0.10.5-1.el5.2 qemu-system-
sh4.i386 2:0.10.5-1.el5.2 qemu-system-sparc.i386 2:0.10.5-1.el5.2 qemu-system-x86.i386
2:0.10.5-1.el5.2 qemu-user.i386 2:0.10.5-1.el5.2
```

```
Complete!
```

Всё! Проверяем доступность установленного QEMU:

```
$ which qemu
```

```
/usr/bin/qemu
```

```
$ qemu
```

```
QEMU PC emulator version 0.10.5, Copyright (c) 2003-2008 Fabrice Bellard
```

```
usage: qemu [options] [disk_image]
```

```
...
```

- здесь вы получите обстоятельнейшую справку о деталях командной строки QEMU. Но установить — этого мало, хорошо бы ещё и проверить работоспособность и оценить качество QEMU прежде двигаться дальше. Отправляемся на сайт проета QEMU: <http://www.qemu.org/download.html> и на этой странице находим достаточно много образов различных OS собранных для загрузки в качестве образа, загрузим из них:

FreeDOS floppy disk image

<http://odin.fdos.org/odin2005/odin1440.img>

Выполняем команду (некоторые опции командной строки будут обсуждены позже, или см. внимательно ту справку, которую выводит qemu при запуске без параметров):

```
$ qemu -m 8M -fda odin1440.img -boot a
```

```
Could not open '/dev/kqemu' - QEMU acceleration layer not activated: No such file or directory
```

Может быть дополнительно ещё и такое сообщение:

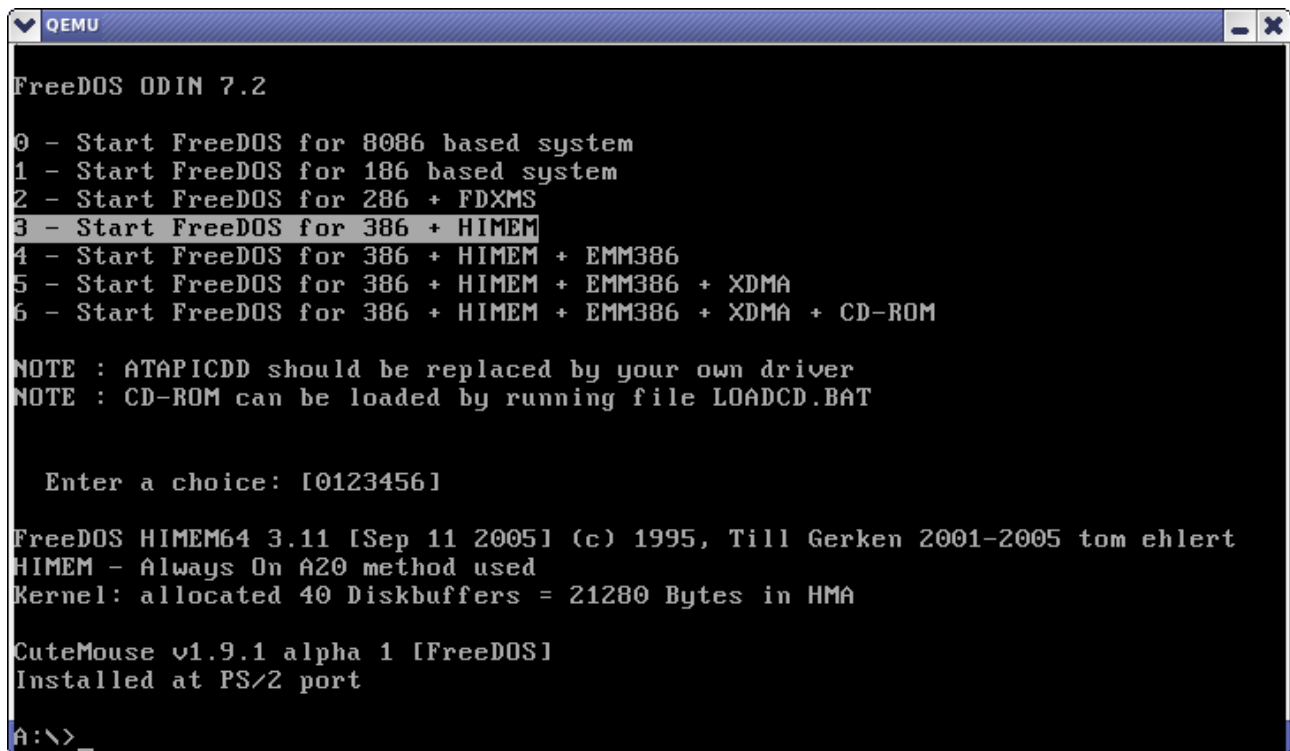
Could not configure '/dev/rtc' to have a 1024 Hz timer. This is not a fatal error, but for better emulation accuracy either use a 2.6 host Linux kernel or type 'echo 1024 > /proc/sys/dev/rtc/max-user-freq' as root.

Для устранения такого сообщения, как в нём и указано, сделаем:

```
# su -c 'echo 1024 > /proc/sys/dev/rtc/max-user-freq'
```

С 1-м сообщением мы станем разбираться чуть далее по тексту.

Но, тем не менее, запуск FreeDOS произошёл, как это показано на рис.1:



```
QEMU
FreeDOS ODIN 7.2

0 - Start FreeDOS for 8086 based system
1 - Start FreeDOS for 186 based system
2 - Start FreeDOS for 286 + FDXMS
3 - Start FreeDOS for 386 + HIMEM
4 - Start FreeDOS for 386 + HIMEM + EMM386
5 - Start FreeDOS for 386 + HIMEM + EMM386 + XDMA
6 - Start FreeDOS for 386 + HIMEM + EMM386 + XDMA + CD-ROM

NOTE : ATAPICDD should be replaced by your own driver
NOTE : CD-ROM can be loaded by running file LOADCD.BAT

Enter a choice: [0123456]

FreeDOS HIMEM64 3.11 [Sep 11 2005] (c) 1995, Till Gerken 2001-2005 tom ehler
HIMEM - Always On A20 method used
Kernel: allocated 40 Diskbuffers = 21280 Bytes in HMA

CuteMouse v1.9.1 alpha 1 [FreeDOS]
Installed at PS/2 port

A:\>
```

Рис.1

Примечание (в качестве напоминания): как делать снимки окон в Linux? Без этого любые описания или документация — голословные, и содержат, как правило, ошибки. Обычный способ: нажать PrintScreen и дождаться появления окна программы. Но часто (при большом числе инсталлированных средств, которые переопределяют и перехватывают клавиши) это может не работать. Тогда делаем прямой вызов программы из командной строки, например (в KDE):

```
$ ksnapshot &
```

```
[1] 10115
```

Теперь возвращаемся к первому из показанных выше сообщений QEMU. Это он «жалуется» на отсутствие акселератора `kqemu`. Вот что пишут относительно этого на <http://www.xakep.ru/magazine/xa/118/094/1.asp> (интересная статья о QEMU, с которой стоит ознакомиться):

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.

Пакет акселератора вы с помощью `uim` не найдёте, и мы поставим его из исходных кодов. Загрузим <http://www.qemu.org/kqemu-1.4.0pre1.tar.gz> с того же сайта проекта QEMU, которым мы уже пользовались. Далее (в удобном нам произвольном каталоге, результат `make` я показываю расширенно — здесь видно, что создаётся модуль ядра Linux):

```
$ tar -zxvf kqemu-1.4.0pre1.tar.gz
```

```
$ cd kqemu-1.4.0pre1
```

```
$ ./configure
```

```

Source path      /usr/src/kqemu-1.4.0pre1
C compiler       gcc
Host C compiler  gcc
make            make
host CPU        i386
kernel sources   /lib/modules/2.6.18-92.el5/build
kbuild type     2.6
$ make
make -C common all
...
gcc -D__KERNEL__ -nostdinc -iwithprefix include -I. -I.. -D__ASSEMBLY__ -c -o
i386/nexus_asm.o i386/nexus_asm.S
gcc -D__KERNEL__ -nostdinc -iwithprefix include -I. -I.. -D__ASSEMBLY__ -c -o
i386/monitor_asm.o i386/monitor_asm.S
...
gcc -D__KERNEL__ -nostdinc -iwithprefix include -I. -I.. -D__ASSEMBLY__ -c -o
i386/kernel_asm.o i386/kernel_asm.S
ld -r -o ../kqemu-mod-i386.o kernel.o i386/kernel_asm.o
make[1]: Leaving directory `/usr/src/kqemu-1.4.0pre1/common'
make -C /lib/modules/2.6.18-92.el5/build M=`pwd` modules
make[1]: Entering directory `/usr/src/kernels/2.6.18-92.el5-i686'
  CC [M]  /usr/src/kqemu-1.4.0pre1/kqemu-linux.o
cp /usr/src/kqemu-1.4.0pre1/kqemu-mod-i386.o /usr/src/kqemu-1.4.0pre1/kqemu-mod.o
  LD [M]  /usr/src/kqemu-1.4.0pre1/kqemu.o
Building modules, stage 2.
MODPOST
WARNING:      could      not      find      /usr/src/kqemu-1.4.0pre1/.kqemu-mod.o.cmd
for /usr/src/kqemu-1.4.0pre1/kqemu-mod.o
  CC      /usr/src/kqemu-1.4.0pre1/kqemu.mod.o
  LD [M]  /usr/src/kqemu-1.4.0pre1/kqemu.ko
make[1]: Leaving directory `/usr/src/kernels/2.6.18-92.el5-i686'
$ su -c'make install'
./install.sh

```

Этого достаточно для инсталляции модуля, но недостаточно для загрузки:

```

# lsmod | head -n3
Module                Size  Used by
mga                   62145  3
drm                   65493  4 mga
# modprobe kqemu
# lsmod | head -n3
Module                Size  Used by
kqemu                 131108  0
mga                   62145  3
# chmod a+rw /dev/kqemu
# ls -l /dev/kqemu

```

```
crw-rw-rw- 1 root root 10, 62 Ноя  7 15:57 /dev/kqemu
```

- загрузили модуль (плюс проделали ещё некоторые манипуляции с его правами, которые стационарно легко уладить с помощью udev). Теперь загрузка виртуальной машины должна происходить без единого предупреждения:

```
$ qemu -m 8M -fda odin1440.img -boot a
```

Всё! Теперь вы готовы виртуально выполнять любую OS на любой процессорной платформе.

Примечание: при последнем показанном запуске можно получить сообщение:

```
Version mismatch between kqemu module and qemu (00010300 00010400) - disabling kqemu use
```

И это очевидно не то, чего мы добивались! Это связано с версиями qemu и kqemu, как сказано на сайте проекта QEMU они должны соответствовать друг другу:

<http://www.qemu.org/kqemu-1.3.0pre11.tar.gz> Use with QEMU <= 0.9.1. Full source code available under the GPL license.

<http://www.qemu.org/kqemu-1.4.0pre1.tar.gz> Use with the QEMU development version in the Subversion repository

А как было показано ранее, в начале этого раздела, с помощью yum мы может установить как версию qemu 0.9.x, так и 0.10.x.

И последнее замечание относительно использования qemu. Любая гостевая система, работающая в окне qemu (например, рис.1), может захватывать клавиатуру, а, главное, мышь для своей работы. При этом может возникнуть ложное впечатление, что система (Linux) в которой выполняется qemu — зависла, но это не так: органы управления просто принадлежат в данный момент (захвачены) виртуальной операционной системой; чтобы их освободить нажмите Ctrl+Alt.

Установка MINIX3

Загрузка с CD

Теперь у нас всё готово для запуска MINIX3 под QEMU. Скачиваем последний по версии образ загрузочного CD, на время написания этого текст это был http://www.minix3.org/download/minix_R3.1.5-r5612.iso.bz2 . Если образ сжат bzip2 (некоторые версии сжаты, другие — нет), то разархивируем его, возможно, предварительно проверив для контроля целостность архива:

```
$ bzip2 -t minix_R3.1.5-r5612.iso.bz2
```

```
$ bzip2 -d minix_R3.1.5-r5612.iso.bz2
```

```
$ ls -l
```

```
-rw-rw-r-- 1 olej olej 640679936 Ноя  6 10:25 minix_R3.1.5-r5612.iso
```

Вот теперь перед нами ISO-образ последнего релиза OS MINIX3. Накатаем (любой программой для того предназначенной, например K3b в Linux) этот образ на CD. Это LiveCD, поэтому мы можем грузить OS непосредственно с него, или устанавливая на носитель. Сначала проделаем только загрузку:

```
$ qemu -m 100M -cdrom minix_R3.1.5-r5612.iso -boot d
```

Точно того же результата (внимание!) мы достигаем и используя записанный ранее LiveCD, командой:

```
$ qemu -m 100M -cdrom /dev/cdrom -boot d
```

или

```
$ qemu -m 100M -cdrom /dev/hda -boot d
```

Примечание: это у меня CD-ROM /dev/hda (именно так!) у вас он, скорее всего, будет под другой литерой.

```
QEMU
The system is now running and many commands work normally. To use MINIX
in a serious way, you need to install it to your hard disk, which you
can do by typing 'setup' while logged in as root. Then just follow the
on-screen directions.

After setup is complete, type 'shutdown' and when the boot monitor
starts, boot your new system by following the instructions at the end of
setup. Keep the CD-ROM in the drive, login as root and type 'packman'
to begin installing the many software packages available. After you
have installed the packages, type 'xdm' to start X Windows if you have
installed it.

Before you begin the installation process, it is strongly recommended
that you print and carefully read the installation instructions
available on the MINIX 3 website: http://www.minix3.org.

Minix Release 3 Version 1.5 (console)

minix login: root
#
#
# uname -a
Minix minix 3 1.5 i686
#
```

Рис.2

Установка на диск

Для установки на виртуальный диск нужно прежде создать файл, который будет имитировать этот виртуальный диск. Поскольку это будет рабочий диск MINIX3 на достаточно долгое время, то его размер должен быть весьма значительным (от нескольких десятков Mb до Gb), поэтому прежде есть смысл поинтересоваться наличием свободного места на дисковом пространстве Linux:

```
$ df
Файловая система      1К-блоков      Исп  Доступно  Исп% смонтирована на
/dev/hdf6              7640636      6842276   403968   95% /
tmpfs                  143956         0    143956    0% /dev/shm
/dev/hde1              3775880      2491336  1284544   66% /mnt/win_c
/dev/hdf4              7151004      3426472  3724532   48% /mnt/win_d
/dev/hde5              6221016        1116   6219900    1% /mnt/win_e
```

Я буду создавать виртуальный диск минимального размера, на который будет устанавливаться MINIX3, размером 90Mb (это близко к минимальному размеру 79Mb, который будет указывать этот релиз системы, остаток ~10Mb отдадим под /home), это годится для экспериментов, но вряд ли достаточно для рабочей системы:

```
$ qemu-img create minix3-disk 90M
Formatting 'minix3-disk', fmt=raw, size=92160 kB
$ ls -l minix3-*
-rw-r--r-- 1 olej olej 94371840 Ноя  7 18:54 minix3-disk
```

Хотя то же самое можно сделать и при помощи стандартной утилиты dd, для примерно того же размера, например, так:

```
$ dd of=minix3-disk1 bs=1024 seek=90000 count=0
...
$ ls -l minix3-*
```



```
-rw-r--r-- 1 olej olej 94371840 Ноя  8 20:01 minix3-disk
-rw-rw-r-- 1 olej olej 92160000 Ноя 11 10:36 minix3-disk1
```

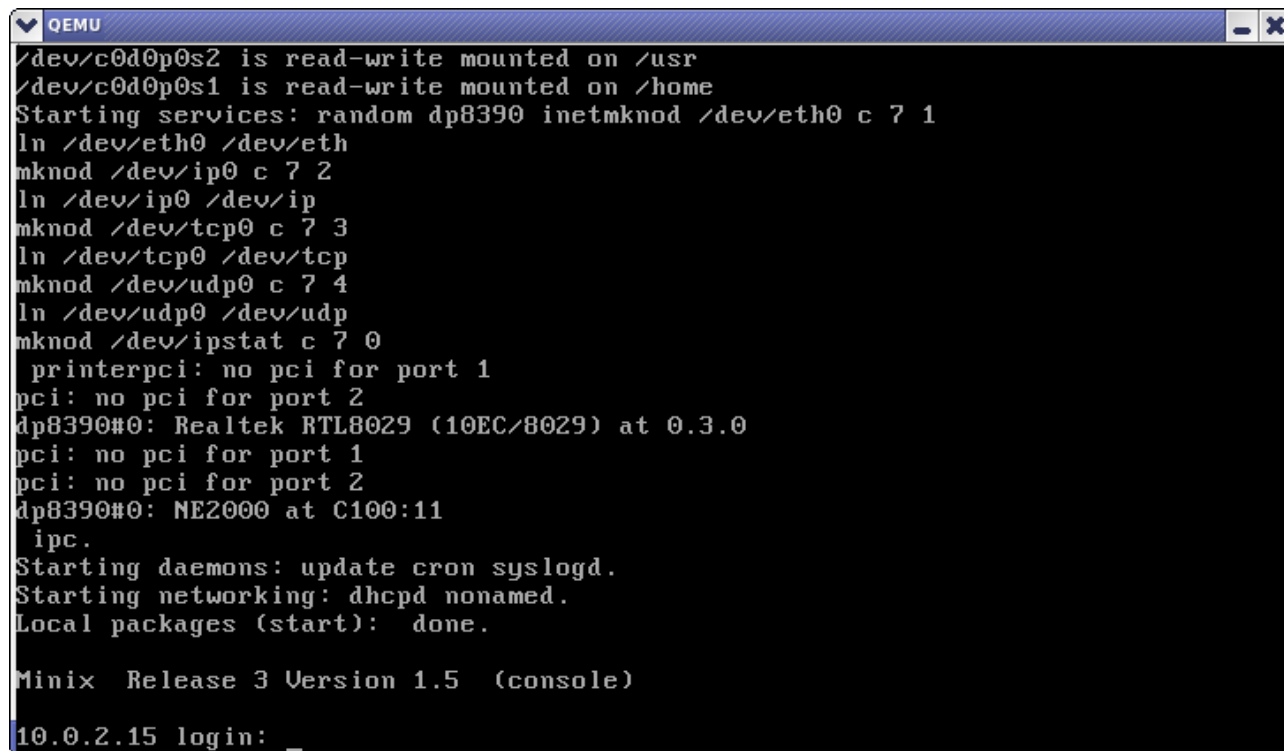
Примечание: есть отличие этих двух способов; утилита `dd` позволяет создать только `raw`-образ, который представляет собой файл, заполненный нулями. Утилита `qemu-img` поддерживает несколько различных форматов образа, указать на которые можно при помощи параметра `format`. По умолчанию создаются `qcow`-файлы (`qemu Copy On Write`). Этот формат поддерживает шифрование (AES, 128 бит) и компрессию, но возможны еще: `raw`-, `cow`- (`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 самим драйвером файловой системы. И у этого способа есть еще один несомненный плюс – можно монтировать в дерево ФС и работать как с обычным дисковым разделом. Утилита `qemu-img` поддерживает параметр `convert`, позволяющий преобразовывать образы из одного формата в другой.

Двигаемся дальше: загружаем CD (в точности так, как это делалось ранее) и приступаем к установке на заготовленный виртуальный диск:

```
$ qemu -m 100M -hda minix3-disk -cdrom /dev/cdrom -boot d -localtime
```

Дальше, вооружившись текстом по инсталляции: <http://www.minix3.ru/docs/setup-russian.pdf>, пошагово следуем этой инструкции установки на раздел реального диска, никаких принципиальных различий не встречаем: зарегистрировавшись в `MINIX3` как `root` (без пароля)... поехали:

```
# setup
```



```
QEMU
/dev/c0d0p0s2 is read-write mounted on /usr
/dev/c0d0p0s1 is read-write mounted on /home
Starting services: random dp8390 inetmknod /dev/eth0 c 7 1
ln /dev/eth0 /dev/eth
mknod /dev/ip0 c 7 2
ln /dev/ip0 /dev/ip
mknod /dev/tcp0 c 7 3
ln /dev/tcp0 /dev/tcp
mknod /dev/udp0 c 7 4
ln /dev/udp0 /dev/udp
mknod /dev/ipstat c 7 0
printerpci: no pci for port 1
pci: no pci for port 2
dp8390#0: Realtek RTL8029 (10EC/8029) at 0.3.0
pci: no pci for port 1
pci: no pci for port 2
dp8390#0: NE2000 at C100:11
ipc.
Starting daemons: update cron syslogd.
Starting networking: dhcpd nonamed.
Local packages (start): done.

Minix Release 3 Version 1.5 (console)
10.0.2.15 login:
```

Рис.3

После завершения установки производим загрузку только что установленной системы с виртуального диска (рис.3):

```
$ qemu -m 100M -hda minix3-disk -boot c -localtime
```

«Реальный» MINIX3 в QEMU

На другом компьютере (это специально подчеркнуто, чтобы не соотносить имена разделов дисков с предыдущим текстом), на котором был ранее установлен «реальный» `MINIX3` в раздел диска `/dev/hda2` :

```
# fdisk /dev/hda
```

```
Устр-во Загр      Начало           Конец           Блоки      Id      Система
```

```

/dev/hda1          1          511      4104576    b  W95 FAT32
/dev/hda2          512          956      3574462+   81  Minix / старый Linux
/dev/hda3    *      957          1723      6160927+   4f  QNX4.x 3-я часть
/dev/hda4          1724         2491      6168960    f  W95 расшир. (LBA)
/dev/hda5          1724         1787        514048+   82  Linux своп / Solaris
/dev/hda6          1788         2491      5654848+   83  Linux

```

- и запустим QEMU вот так:

```
$ qemu -m 50M -hda /dev/hda -boot a -localtime
```

```
qemu: could not open disk image /dev/hda2
```

Что, в общем, понятно (права доступа):

```
# ls -l /dev/hda
```

```
brw-r----- 1 root disk 3, 0 Ноя  8 08:54 /dev/hda
```

Но, поскольку очень уж не хочется выполнять qemu под правами root, то сделаем это так:

```
# chmod a+rw /dev/hda2
```

И ещё раз выполняем:

```
$ qemu -m 50M -hda /dev/hda -boot a -localtime
```

...

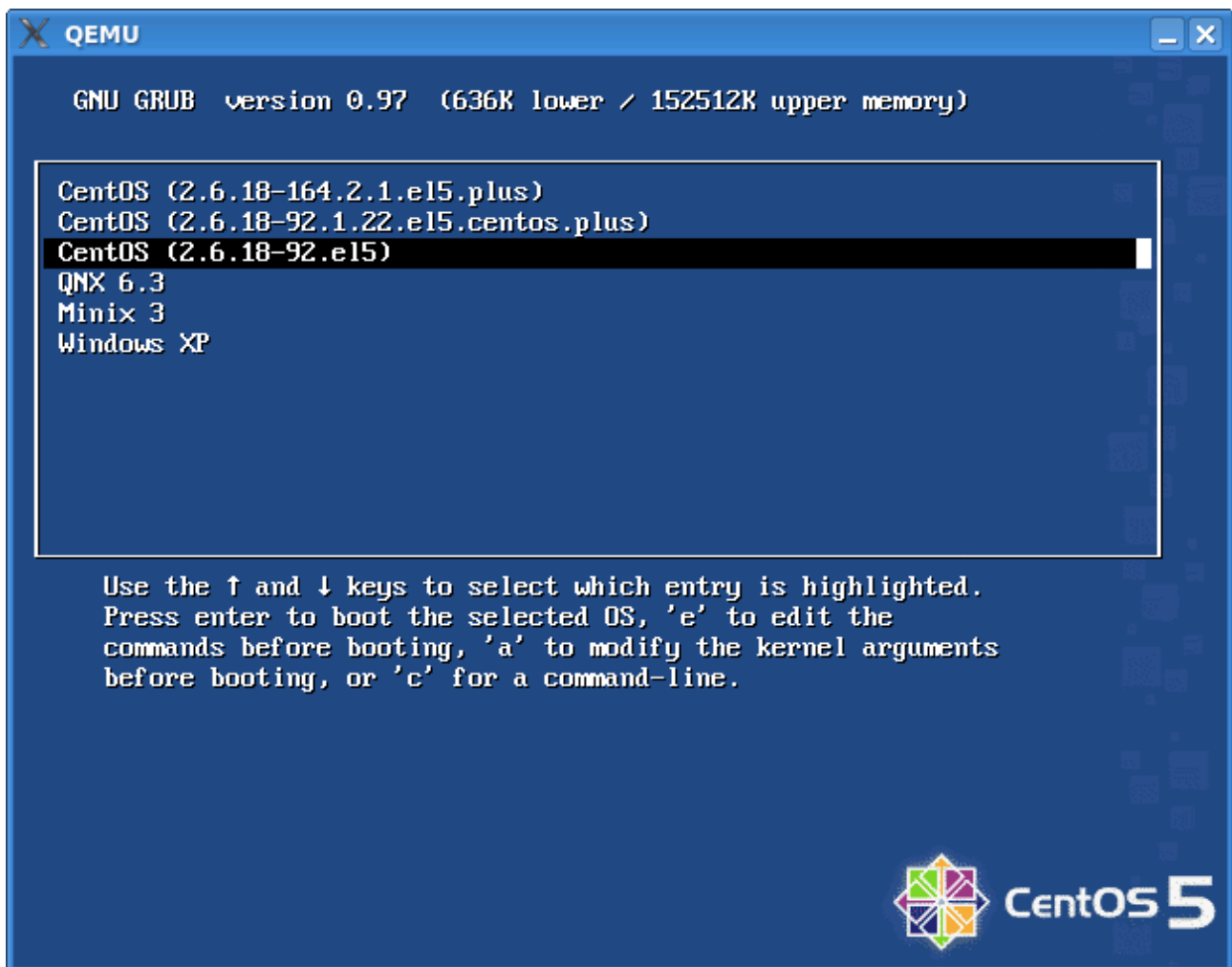


Рис. 4

В результате, мы проходим путь загрузки в qemu от загрузчика GRUB (рис.4) до загрузки (выбрав в меню «Minix 3») реально установленного (/dev/hda2) экземпляра MINIX3; на рис.5 показано выполнение

установленного в составе «реального» MINIX3 MidnightComander, выполняющийся в окне qemu.

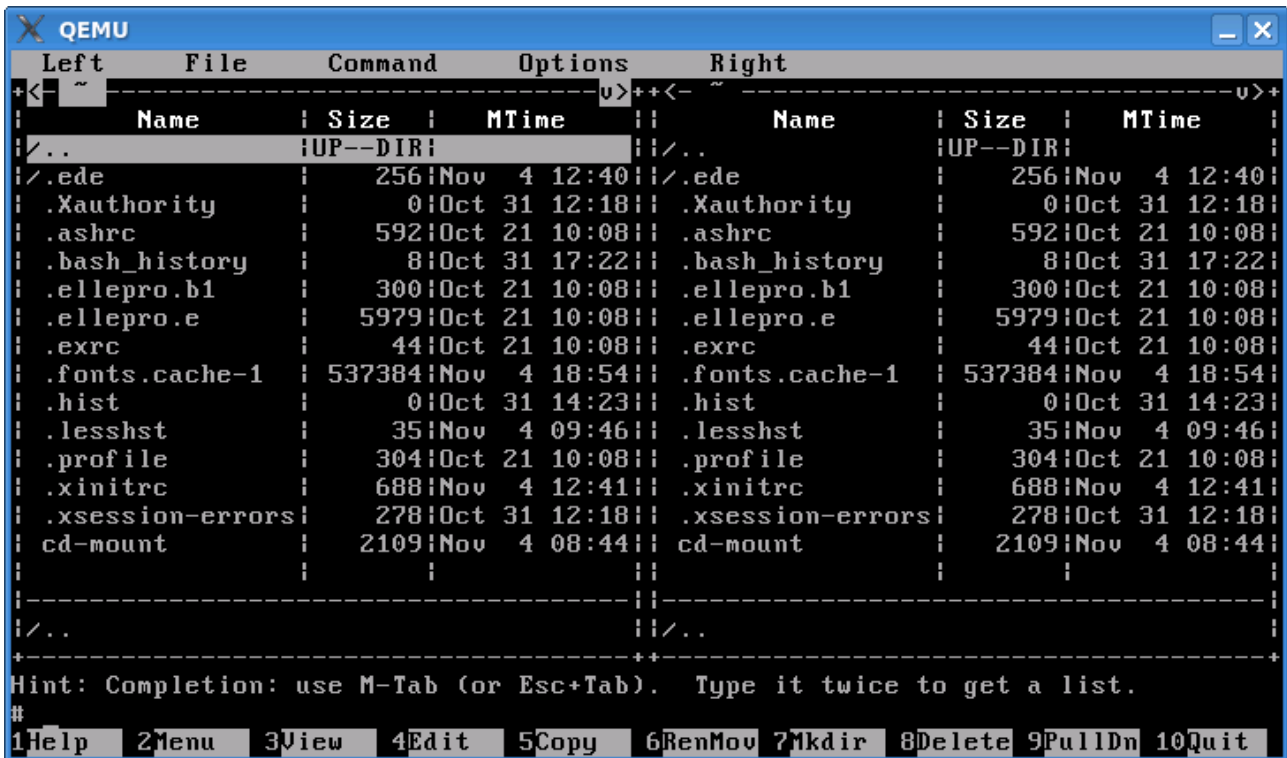


Рис. 5

Зачем нужно такое «извращение»? Таким путём мы имеем возможность запускать **один и тот же экземпляр** ОС:

- либо как реальную, для устранения любых эффектов виртуализации, адекватной проверки характеристик и возможностей системы без всяких привносимых особенностей;
- либо как виртуальную, но с дополнительными возможностями обмена данными с другими файловыми системами, например, MINIX3 очень капризен к устанавливаемым сетевым картам, но в таком варианте можно установить «виртуальное» сетевое соединение с базовой системой и по нему организовать обмен данными.

Оживляем сетку

Сеть к виртуальной ОС также предполагается как виртуальная, т.е. мы должны установить в этой виртуальной подсети соединение MINIX3 к базовому Linux, а оттуда уже, по необходимости, обеспечит роутинг в реальную LAN, или в наружу в Интернет. В моём случае в Linux уже установлена достаточно развитая сетевая подсистема:

```
# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:60:52:07:4F:4B
          inet addr:192.168.1.7  Bcast:192.168.1.7  Mask:255.255.255.248
          inet6 addr: fe80::260:52ff:fe07:4f4b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2337 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2631 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:820615 (801.3 KiB)  TX bytes:355058 (346.7 KiB)
          Interrupt:10 Base address:0x4000

eth1      Link encap:Ethernet  HWaddr 00:4F:49:00:B0:F9
```

```
BROADCAST MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)
Interrupt:10 Base address:0xec00
```

...

Здесь видно 2 сетевых интерфейса: (eth0) подключенный к LAN 192.168.1.7:255.255.255.248, и другой (eth1) в данный момент не активный (во 2-й строке нет UP). Для того, чтобы быть уверенным в неискажаемости результатов, виртуальную подсеть к MINIX3 будем строить в совсем другой подсети IP адресов: 192.168.2.0:255.255.255.0. Для этого...

Со стороны Linux:

1. создаём в /etc 2 скрипта (старта и останова) для тунельного интерфейса, они имеют права исполнения и установленный SUID бит:

```
# ls -l /etc/qem*
-rwsr-sr-x 1 root root 78 Ноя 8 16:32 /etc/qemu-ifdown
-rwsr-sr-x 1 root root 116 Ноя 8 16:32 /etc/qemu-ifup
# cat /etc/qemu-ifup
#!/bin/sh
echo ----- tap up -----
sudo /sbin/ifconfig $1 192.168.2.6
# cat /etc/qemu-ifdown
#!/bin/sh
echo ----- tap down -----
#/sbin/ifconfig $1 down
```

Скрипт остановки здесь реально ничего не делает ... но может быть, при необходимости, наполнен содержанием.

2. запускаем виртуальную машину так:

```
$ su -c 'qemu -m 100M -hda minix3-disk -boot c -localtime -net nic,vlan=0 -net tap,vlan=0'
...
```

- здесь запуск происходит от имени root, что связано с необходимостью инициализации сетевого интерфейса (это крайне нежелательно, но как от этого избавиться, и возможно ли — это вы разберётесь при желании сами):

```
# ifconfig -a
...
tap0      Link encap:Ethernet  HWaddr 00:FF:F7:C0:85:56
          inet addr:192.168.2.6  Bcast:192.168.2.255  Mask:255.255.255.0
          inet6 addr: fe80::2ff:f7ff:fec0:8556/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:82 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 b)  TX bytes:17858 (17.4 KiB)
```

Шлюз по умолчанию в LAN установлен через интерфейс eth0 (192.168.1.1):

```
# route
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.1.0	*	255.255.255.248	U	0	0	0	eth0
192.168.2.0	*	255.255.255.0	U	0	0	0	tap0
169.254.0.0	*	255.255.0.0	U	0	0	0	eth0
default	192.168.1.1	0.0.0.0	UG	0	0	0	eth0

Со стороны MINIX3:

1. В /etc/inet.conf пропишем:

```
eth0 dp8390 0 { default; };
```

- чему устанавливает до 4-х сетевых интерфейсов типа NE2000, а сетевая плата NE200 в MINIX3 находится в драйвере dp8390 (/usr/src/drivers/dp8390).

```

QEMU
supports multiple virtual terminals. Just use ALT+F1, F2, F3 and F4 to
navigate among them.

To get rid of this message, edit /etc/motd.

#
# ifconfig -a
# ls /dev/ip*
/dev/ip /dev/ip0 /dev/ipstat
# cat /dev/ipstat
cat: /dev/ipstat: I/O error
# ifconfig -h
option requires an argument -- h
Usage: ifconfig [-I ip-device] [-h ipaddr] [-n netmask] [-m mtu] [-iva]
# ifconfig -I /dev/ip0
ifconfig: /dev/ip0: Host address not set
# ifconfig -I /dev/eth0
ifconfig: Unable to get IP configuration: Illegal ioctl for device
# ifconfig -I /dev/ip0 -h 192.168.2.4
#
# ping 192.168.2.4
192.168.2.4 is alive
# ping 192.168.2.6
192.168.2.6 is alive
#

```

Рис. 6

2. Сетевому интерфейсу (умалчиваемому) присвоим IP адрес (после экспериментов это следует поместить куда-то в стартовый скрипт, например, /etc/profile):

```

# ifconfig -I /dev/ip
ifconfig: /dev/ip: Host address not set
# ifconfig -I /dev/ip -h 192.168.2.4
# ifconfig -av
/dev/ip0: address 192.168.2.4 mtu 1500

```

После этого мы можем пинговать достижимость хоста Linux, как показано на рис.6.

Или, напротив, достижимость MINIX3 с хоста Linux:

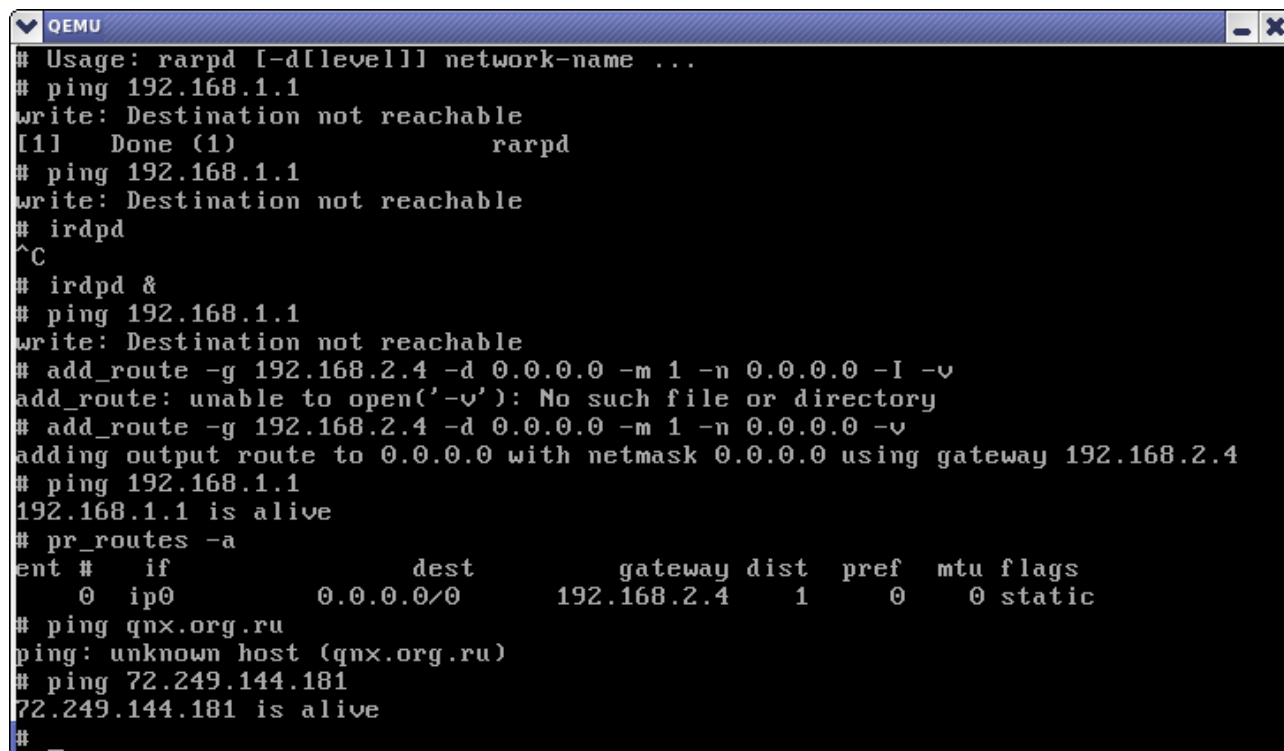
```

# ping 192.168.2.4
PING 192.168.2.4 (192.168.2.4) 56(84) bytes of data.
64 bytes from 192.168.2.4: icmp_seq=1 ttl=96 time=406 ms

```

```
64 bytes from 192.168.2.4: icmp_seq=2 ttl=96 time=81.6 ms
64 bytes from 192.168.2.4: icmp_seq=3 ttl=96 time=114 ms
--- 192.168.2.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 81.691/201.049/406.757/146.080 ms
```

Но это ещё не даёт нам возможность доступа к шлюзу (192.168.1.1) в LAN и далее:



```
QEMU
# Usage: rarpd [-d[level]] network-name ...
# ping 192.168.1.1
write: Destination not reachable
[1] Done (1) rarpd
# ping 192.168.1.1
write: Destination not reachable
# irdpd
^C
# irdpd &
# ping 192.168.1.1
write: Destination not reachable
# add_route -g 192.168.2.4 -d 0.0.0.0 -m 1 -n 0.0.0.0 -I -v
add_route: unable to open('-v'): No such file or directory
# add_route -g 192.168.2.4 -d 0.0.0.0 -m 1 -n 0.0.0.0 -v
adding output route to 0.0.0.0 with netmask 0.0.0.0 using gateway 192.168.2.4
# ping 192.168.1.1
192.168.1.1 is alive
# pr_routes -a
ent # if dest gateway dist pref mtu flags
0 ip0 0.0.0.0/0 192.168.2.4 1 0 0 static
# ping qnx.org.ru
ping: unknown host (qnx.org.ru)
# ping 72.249.144.181
72.249.144.181 is alive
#
```

Рис.7

Всё, что необходимо для установления маршрута, показано снимком экрана MINIX3 на рис.7.

Примечание: показанный для образца ping qnx.org.ru в Интернет — недостижим, поскольку в нашем MINIX3 не работает (не настроена?) служба разрешения имён (DNS и др.); но если мы определим IP адрес тестируемого хоста вручную:

```
$ nslookup qnx.org.ru
Server: 195.5.51.182
Address: 195.5.51.182#53
Non-authoritative answer:
Name: qnx.org.ru
Address: 72.249.144.181
```

- то непосредственно по IP хост в Интернет достижим, что и показывает последняя команда на рис.7.

Удалённый доступ

Работать с консоли MINIX3 занятие ... не комфортное и утомительное. Но, если у нас есть теперь сеть из «комфортного» Linux в «не комфортный» виртуальный MINIX3, то мы можем просто всю работу в ОС MINIX3 производить из терминалов (сколько угодно много по числу) Linux.

1. Доступ к текстовой консоли по SSH.

```
$ ssh -luser 192.168.2.4
```

Password: ...

Примечание: к этому месту как раз вовремя или назначить (изменить) пароль для root (при установке MINIX3 root остался с пустым паролем) командой `passwd`; либо добавить нового пользователя MINIX3 командой `adduser` — `ssh` не допускает обращений пользователей без паролей (хотя, возможно, этого и можно добиться настройками `sshd`?).

2. Симметричным образом можно и со стороны MINIX3 получить SSH-доступ к Linux, рис.8.

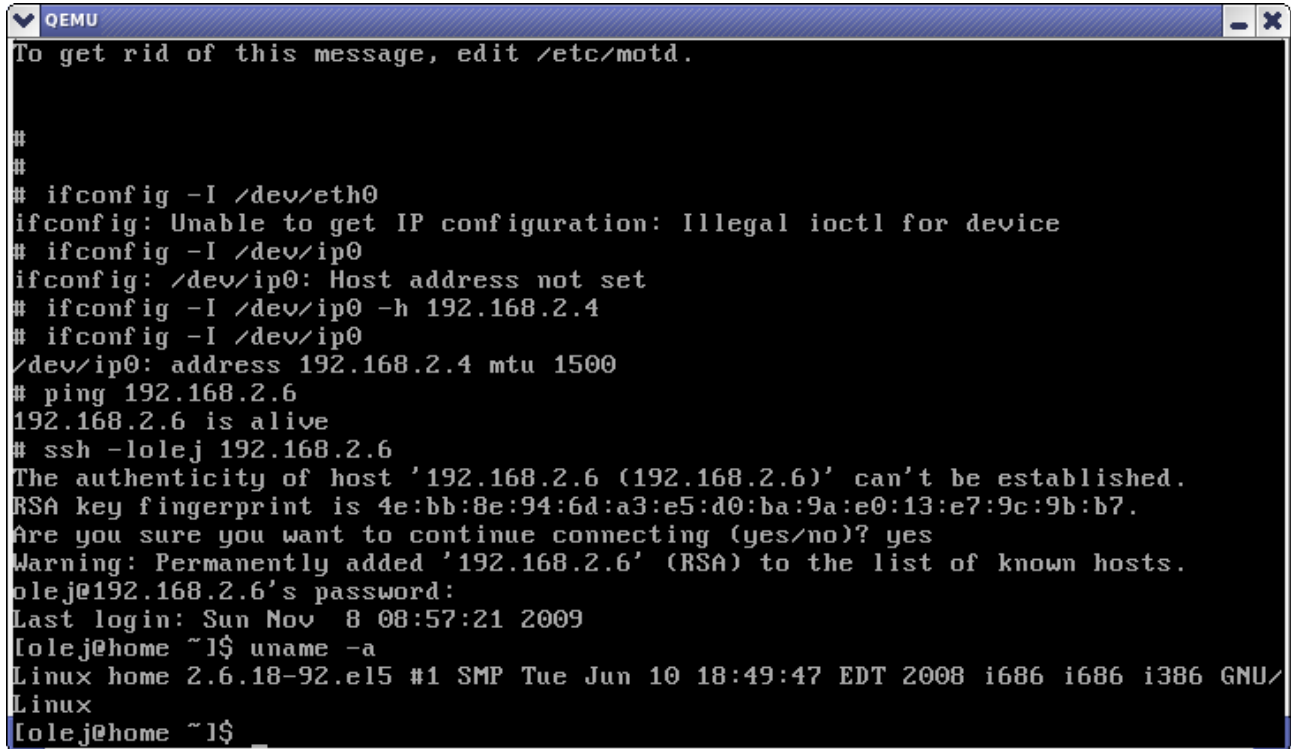


Рис. 8

3. Доступ из одной из панелей mc в Linux по SSH к хосту MINIX3.

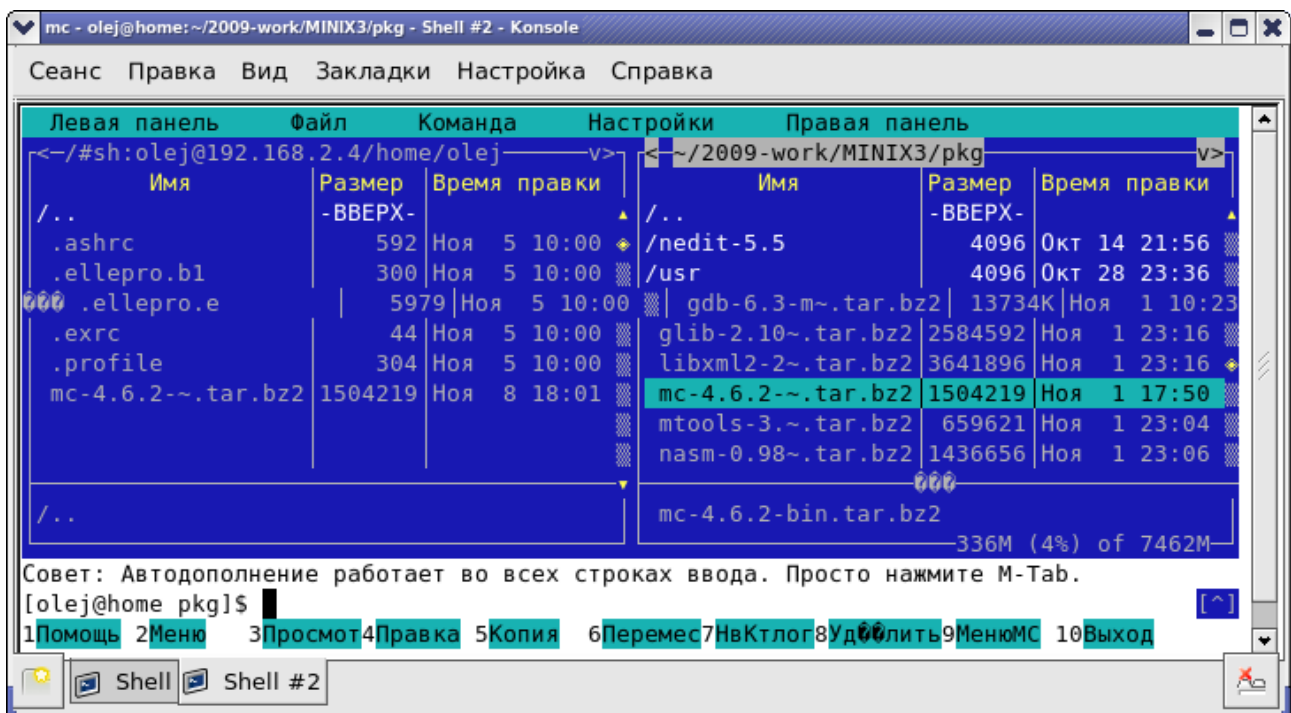


Рис. 9

Это достигается, например, следующей последовательностью действий-меню: F9 -> «Левая панель» -> «Shell соединение», а на предложение ввести имя компьютера вводим: `user@192.168.2.4` (это именно в точности соответствует SSH-команде, показанной выше — здесь другой синтаксис).

Теперь мы имеем то, что показано на рис.9, где в левой панели — хост MINIX3 (192.168.2.4), в правой — Linux (192.168.2.6), и мы привычными «менюшными» операциями `mc` по F3 / F4 / F5 / F6 / F7 / F8 - оперируем с файлами на любом из хостов и гоняем их туда-сюда.

Примечание: операция копирования с шифрованием потока по SSH — достаточно ресурсоёмкая операция (если ещё и учесть, что на одном конце она выполняется в виртуальной машине), показанная на рис.9 операция копирования дистрибутивного пакета `mc` на MINIX3 занимала у меня 1-2 минуты.

Дополнительные источники информации

1. Документация пользователя эмулятора процессора QEMU

Перевод: Павел Марьянов, март 2006

<http://jack.kiev.ua/docs/qemu-doc-ru.html>

2. Виртуальный полигон: Эмулируем аппаратное обеспечение различных платформ с помощью QEMU

Владимир Ляшко

<http://www.xakep.ru/magazine/xa/118/094/1.asp>

3. Установка MINIX3

Перевод: Роман Игнатов, Павел Макаров

<http://www.minix3.ru/docs/setup-russian.pdf>

4. Описания не документированных команд MINIX3 `add_route`, `pr_routes` и др.

<http://www.os-forum.com/minix/net/general-comment-display.php?commentid=171>