

# Сеть MINIX3. Часть 2: инструменты

Циллорик О.И.

< [olej@front.ru](mailto:olej@front.ru) >

Редакция 4.18

от 25.02.2010

## Оглавление

Аннотация.....	1
Введение.....	1
Версии системы.....	2
Структура документа.....	2
Инструменты сети.....	2
Утилиты сети Linux.....	2
ping.....	2
ifconfig.....	3
route.....	3
ip.....	3
brctl.....	4
tunctl.....	5
iptables.....	5
Утилиты сети MINIX3.....	5
ping.....	5
ifconfig.....	5
host.....	6
hostaddr.....	7
tcpstat.....	8
udpstat.....	9
pr_routes.....	9
add_route.....	9
del_route.....	9
arp.....	9
irdpd.....	10
rarpd.....	10
tcpdump.....	10
Дополнительные источники информации.....	12

## Аннотация

В этой части рассматриваются сетевые утилиты под MINIX3 и работа с ними. В рассмотрение включены не только реализации, включённые в состав какого-то конкретного релиза официального дистрибутива, но и весь спектр существующих реализаций. Относительно каждого средства, если оно есть официальным компонентом MINIX3, освещены: использование и, где это необходимо, его настройка. Относительно сторонних продуктов, представленных программными пакетами, дополнительно рассматривается их установка.

## Введение

Сетевые инструменты (утилиты) MINIX3, несмотря на декларируемую совместимость MINIX3 с POSIX, значительно (по крайней мере, значительно больше, чем любая другая группа команд) отличаются от привычных, например, от присутствующих в Linux или OpenSolaris. Хотя, при внешнем отличии, все они основываются на тех же RFC и механизмах. На этой общности мы постараемся, по возможности, акцентироваться в дальнейшем.

Большая часть изложения и примеров будет построена на рассмотрении MINIX3, выполняющегося под управлением системы виртуализации QEMU. Тому есть целый ряд оснований:

- в MINIX3 крайне ограниченный набор поддерживаемых сетевых карт, и даже входящие в него образцы, это устаревшие модели, выходящие из употребления.
- виртуальные сети QEMU позволяют смоделировать весьма разветвлённые сетевые конфигурации, на них можно показать достаточно изощёренные примеры.
- сетевые процессы в виртуальных сетях под QEMU сложнее, происходящее в реальной сети в точности повторяет их; таким способом рассмотрения только повышается его степень общности.

Из-за такой установки, в рассмотрении инструментов будет вовлекаться и предметы, специфичные для Linux (в котором выполняется QEMU), не только MINIX3.

## Версии системы

Версии MINIX3 в очень большой мере «волатильны» - разработчики часто вносят существенные изменения, даже не считая должным отражать их даже в map-ах. Вся основная часть описания отработывалась на стабильной версии 3.1.6 (релиз 6084), в некоторых случаях, оговоренных особо, рассмотрение ведётся на стабильной версии 3.1.5 (релиз 5612). В используемой вами версии могут быть, порой, довольно существенные отличия, но основные принципы при этом сохраняются.

## Структура документа

Все показанные в тексте протоколы выполнения команд сохранены прямым копированием с экрана терминала, так же как и графические скриншоты; все действия, описываемые в тексте, могут быть повторно воспроизведены.

В самом тексте, все примеры команд (скопированные с терминала) будет показываться моноширинным шрифтом. Кроме того, в большинстве случаев пользовательский ввод в записи команды будет показан жирным шрифтом, а ответный вывод от системы — обычным. Короткие цитаты из различных источников информации будут показываться курсивом.

## Инструменты сети

Ниже приводятся параллельно сведения о сетевых утилитах и MINIX3 и, в весьма ограниченном объёме, Linux. В этом видится несколько смыслов:

- MINIX3 очень часто (наиболее часто) выполняется под управлением какого-то типа виртуальной машины, одной из которых есть QEMU;
- Linux хост (на котором выполняется QEMU) соединяется с гостевой MINIX3 системой посредством виртуальной сети, которая требует настройки её с обеих сторон;
- при настройках виртуальной сети со стороны Linux используются и не совсем традиционные утилиты (`tunctl`, `brctl`), недостаточно известные, некоторые требующие отдельной установки;
- утилиты 2-х систем, имеющих сходное предназначение (`ifconfig`, `ping`) существенно отличаются, и их удобно рассматривать сравнительно;

## Утилиты сети Linux

Приводится только краткая сводка команд Linux, которые активно используются при работе с сетью MINIX3, выполняющейся под управлением виртуальной машины QEMU в Linux. Показаны образцы использования утилит, но они не рассматриваются в деталях, потому как информация по сетевым утилитам Linux и так во множестве доступна — это скорее просто перечисление (с кратчайшими примерами использования) тех утилит Linux, которые крайне необходимы при работе с сетью MINIX3. В конце текста приведены источники, которые я считаю вполне достаточными для самого детального изучения этих средств.

## ping

Тестирование сетевой прозрачности от текущего хоста до указанного, посылка ICMP запроса и ожидание на

него ICMP ответа:

```
# 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=128 time=1.45 ms  
64 bytes from 192.168.2.4: icmp_seq=2 ttl=128 time=1.30 ms  
64 bytes from 192.168.2.4: icmp_seq=3 ttl=128 time=1.30 ms  
--- 192.168.2.4 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2000ms  
rtt min/avg/max/mdev = 1.301/1.352/1.453/0.082 ms
```

## ifconfig

Диагностика и управление сетевыми интерфейсами. Вот так в Linux выглядит интерфейс виртуальной сети к MINIX3 когда он выполняется в системе QEMU:

```
# ifconfig tap0
```

```
tap0      Link encap:Ethernet  HWaddr 5A:D1:C5:12:E6:C9  
          inet addr:192.168.3.6  Bcast:192.168.3.255  Mask:255.255.255.0  
          inet6 addr: fe80::58d1:c5ff:fe12:e6c9/64 Scope:Link  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
          RX packets:4625 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:12182 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:500  
          RX bytes:427912 (417.8 KiB)  TX bytes:1825693 (1.7 MiB)
```

## route

Диагностика таблицы маршрутизации:

```
[root@home qemu]# route -n
```

```
Kernel IP routing table  
Destination      Gateway          Genmask         Flags Metric Ref    Use Iface  
192.168.1.0      0.0.0.0         255.255.255.248 U        0      0      0 eth0  
192.168.3.0      0.0.0.0         255.255.255.0  U        0      0      0 tap0  
192.168.122.0    0.0.0.0         255.255.255.0  U        0      0      0 virbr0  
169.254.0.0     0.0.0.0         255.255.0.0    U        0      0      0 eth0  
0.0.0.0         192.168.1.1    0.0.0.0         UG       0      0      0 eth0
```

Или изменения в таблице маршрутизации:

```
[root@opos9 etc]# route add default gw 192.168.2.1 dev br0  
[root@home ~]# route add -host 192.168.1.1 dev br0  
[root@opos9 etc]# route del 192.168.3.0
```

## ip

Утилита с очень широкими возможностями для просмотра параметров и конфигурирования сетевых интерфейсов, сетевых адресов, таблиц маршрутизации, правил маршрутизации, arp-таблиц, IP-туннелей, адресов multicast рассылки, маршрутизацией multicast пакетов. Фактически, это «утилита нового стиля», которая интегрирует возможности нескольких традиционных сетевых утилит (ifconfig, route, ...) и может замещать их.

```
[root@home ~]# ip link show
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
```

```
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
link/ether 00:60:52:07:4f:4b brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop qlen 1000
link/ether 00:4f:49:00:b0:f9 brd ff:ff:ff:ff:ff:ff
4: sit0: <NOARP> mtu 1480 qdisc noop
link/sit 0.0.0.0 brd 0.0.0.0
5: virbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
```

**[root@home ~]# ip address show**

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
inet6 ::1/128 scope host
valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
link/ether 00:60:52:07:4f:4b brd ff:ff:ff:ff:ff:ff
inet 192.168.1.7/29 brd 192.168.1.7 scope global eth0
inet6 fe80::260:52ff:fe07:4f4b/64 scope link
valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop qlen 1000
link/ether 00:4f:49:00:b0:f9 brd ff:ff:ff:ff:ff:ff
4: sit0: <NOARP> mtu 1480 qdisc noop
link/sit 0.0.0.0 brd 0.0.0.0
5: virbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
inet6 fe80::200:ff:fe00:0/64 scope link
valid_lft forever preferred_lft forever
6: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 500
link/ether ee:34:22:47:67:2c brd ff:ff:ff:ff:ff:ff
inet 192.168.3.6/24 brd 192.168.3.255 scope global tap0
inet6 fe80::ec34:22ff:fe47:672c/64 scope link
valid_lft forever preferred_lft forever
```

**[root@home ~]# ip route show**

```
192.168.1.0/29 dev eth0 proto kernel scope link src 192.168.1.7
192.168.3.0/24 dev tap0 proto kernel scope link src 192.168.3.6
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1
169.254.0.0/16 dev eth0 scope link
default via 192.168.1.1 dev eth0
```

## brctl

Управление сетевыми мостами MAC уровня.

```
[root@opos9 etc]# brctl addbr br0
```

```
[root@opos9 ~]# brctl addif br0 eth0
```

```
[root@opos9 ~]# brctl show br0
bridge name      bridge id                STP enabled      interfaces
br0              8000.00d0b7169f0d       no               eth0
virbr0          8000.000000000000       yes
```

## tunctl

Управление тунельными (виртуальными) интерфейсами.

```
[root@home ~]# tunctl -u olej -t tap0
Set 'tap0' persistent and owned by uid 500
[root@home ~]# ifconfig tap0 192.168.3.6/24
[root@home ~]# ifconfig tap0 up
[root@home ~]# ip address show tap0
6: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 500
    link/ether 2a:2d:99:a8:a4:0c brd ff:ff:ff:ff:ff:ff
    inet 192.168.3.6/24 brd 192.168.3.255 scope global tap0
    inet6 fe80::282d:99ff:fea8:a40c/64 scope link
        valid_lft forever preferred_lft forever
```

## iptables

Сетевой фильтр, NAT (native address translation) фильтр, файрвол.

```
[root@home ~]# iptables -t nat -A POSTROUTING -s 192.168.3.0/24 -j MASQUERADE
[root@home ~]# iptables -t nat -A POSTROUTING -s 192.168.3.0/24 \
-j SNAT --to-source 192.168.1.7
```

## Утилиты сети MINIX3

Здесь собрана краткая сводка по сетевым утилитам, как они реализованы в MINIX3. Эти программы предназначены не для осуществления сетевых коммуникаций, но для настройки, диагностики и устранения неисправностей в сети. Набор сетевых утилит (и структура сетевого стека) MINIX3 очень значительно отличается от привычного сетевого стека, как он представлен в Linux, SunSolaris, QNX и других POSIX операционных системах.

## ping

ping, в общем, общеизвестная программа для проверки доступности хоста (прохождения протокола ICMP). В реализации MINIX3 уж совсем немногословный:

```
# ping 192.168.3.6
192.168.3.6 is alive
```

Дополнительная информация практически не представлена:

```
# man ping
man: no manual on ping
# ping
Usage: ping hostname [-l length] [-t timeout]
```

## ifconfig

В MINIX3 ifconfig выглядит совершенно иначе чем в Linux:

```
# ifconfig -av
/dev/ip0: address 192.168.3.7 netmask 255.255.255.0 mtu 1500
```

В обеих системах `ifconfig` используется (и мы его вынуждены использовать интенсивно) для присвоения адреса IP сетевому интерфейсу. При этом в MINIX3 `ifconfig` использует совершенно непривычный синтаксис:

```
# ifconfig -I /dev/ip1 -h 192.168.3.7
```

Доступна некоторая справочная информация:

```
# man ifconfig
```

```
...
```

```
SYNOPSIS
```

```
ifconfig [-I ip-device] [-h ipaddr] [-n netmask] [-m mtu] [-iva]
```

```
...
```

```
OPTIONS
```

```
-h The decimal TCP/IP address to set.
-n The netmask to set.
-m The mtu to set. (Minix-vmd only.)
-i Don't set the IP address or netmask if already set. This way
  ifconfig cannot interfere if the numbers have been found out by
  RARP.
-v Report IP address and netmask. This is the default action if there
  are no other options.
-a Report the IP addresses and netmasks of all configured interfaces.
```

## nonamed

Демон разрешения IP имён. Судя по всему, в самом MINIX3 пока не реализованы какие либо средства разрешения сетевых имен типа пакета BIND, с сервером DNS `named`. Единственный доступный механизм разрешения — через записи, прописанные в файле `/etc/hosts`, такое разрешение имён и осуществляет демон `nonamed`. Во всём остальном `nonamed` ведёт себя (старается) как настоящий DNS сервер.

Вот как резолвер имён MINIX3 разрешает доменное имя используя локальный демон `nonamed` (разрешаемое имя должно присутствовать в `/etc/hosts`):

```
# host -v qnx.org.ru 127.0.0.1
```

```
Using domain server 127.0.0.1:
```

```
Trying domain ""
```

```
rcode = 0 (Success), ancourt=1
```

```
qnx.org.ru      3600 IN A      72.249.144.181
```

А вот как он разрешает то же имя пользуясь DNS сервером Linux, находящимся на удалённом хосте в LAN:

```
# host -v qnx.org.ru 192.168.9.2
```

```
Using domain server 192.168.9.2:
```

```
Trying domain ""
```

```
rcode = 0 (Success), ancourt=1
```

```
The following answer is not authoritative:
```

```
qnx.org.ru      84350 IN      A      72.249.144.181
```

Достаточно хорошо описан:

```
# man nonamed
```

```
NAME
```

```
nonamed - not a name daemon, but acts like one
```

```
SYNOPSIS
```

```
nonamed [-qs] [-d[level]] [-p port]
```

```
DESCRIPTION
```

```
Nonamed is not a name daemon. It can answer simple queries from /etc/hosts, but anything else is relayed to a real name daemon.
```

```
...
```

Использует конфигурационный файл /etc/hosts, который достаточно описан:

```
# man hosts
```

```
NAME
```

```
hosts - hostname to IP address database
```

```
SYNOPSIS
```

```
/etc/hosts
```

```
...
```

## host

Резолвер: разрешение сетевых имён, как прямое (имя в IP), так и обратное (IP в имя). Отправляет запросы либо к демону nonamed, либо к внешним DNS серверам (в MINIX3 нет собственной реализации DNS сервера).

Образцы использования:

```
# host -v qnx.org.ru
```

```
Trying domain ""
```

```
rcode = 0 (Success), ancourt=1
```

```
qnx.org.ru      3600 IN A      72.249.144.181
```

```
# host yandex.ru
```

```
yandex.ru has address 213.180.204.11
```

```
yandex.ru has address 77.88.21.11
```

```
yandex.ru has address 87.250.251.11
```

```
yandex.ru has address 93.158.134.11
```

Может конкретизироваться адрес DNS от которого требуется разрешение:

```
# host -v qnx.org.ru 127.0.0.1
```

```
Using domain server 127.0.0.1:
```

```
Trying domain ""
```

```
rcode = 0 (Success), ancourt=1
```

```
qnx.org.ru      3600 IN A      72.249.144.181
```

```
# host -v qnxclub.net 192.168.9.254
```

```
Using domain server 192.168.9.254:
```

```
Trying domain ""
```

```
rcode = 0 (Success), ancourt=1
```

```
The following answer is not authoritative:
```

```
qnxclub.net     86394 IN      A      69.70.20.198
```

```
For authoritative answers, see:
```

```
gnxclub.net      86394 IN      NS      dns2.is47.com
gnxclub.net      86394 IN      NS      dns1.is47.com
Additional information:
dns1.is47.com    70100 IN      A       69.70.20.197
dns2.is47.com    70100 IN      A       69.70.20.196
```

Пример обратного разрешения:

```
# host 213.180.204.11
11.204.180.213.in-addr.arpa PTR yandex.ru
```

Имеется некоторая дополнительная информация:

```
# host
Usage: host [-w] [-v] [-r] [-d] [-V] [-t querytype] [-c class] [-a] host [server]
  -w to wait forever until reply
  -v for verbose output
  -r to disable recursive processing
  -d to turn on debugging output
  -t querytype to look for a specific type of information
  -c class to look for non-Internet data
  -a is equivalent to '-v -t *'
  -V to always use a virtual circuit
```

Достаточно обстоятельное описание:

```
# man host
...
SYNOPSIS
    host [-l] [-v] [-w] [-r] [-d] [-t querytype] [-a] host [ server ]
...
```

## hostaddr

Информация о сетевых параметрах хоста localhost:

```
# hostaddr
52:54:0:12:34:56 192.168.3.7 minix
# hostaddr -i
192.168.3.7
# hostaddr -I /dev/eth0 -i -e
hostaddr: Unable to fetch IP address: Illegal ioctl for device
52:54:0:12:34:56
# hostaddr -I /dev/ip0 -i -e
52:54:0:12:34:56 192.168.2.202
```

Команда достаточно детально описана:

```
# man hostaddr
...
SYNOPSIS
    hostaddr [-eiah] [-E eth-device] [-I ip-device]
DESCRIPTION
```

Without any of the `-eia` options, `hostaddr` shows the ethernet address, IP address and `hostname` of the local host on one line in the given order. With options only the wanted fields are shown, still in the same order, not in option order.

#### OPTIONS

- `-e` Show the ethernet address.
- `-i` Show the IP address.
- `-a` Show the fully qualified hostname. The IP address is shown if it can't be translated to a host name. This usually indicates that the DNS reverse address translation tables are incomplete or that the name daemon couldn't be contacted.
- `-h` Set the hostname of the machine if the caller is the superuser. (Used at boot time by the network initialization scripts.)

...

## tcpstat

Утилита визуализирует текущие активные TCP соединения:

```
# tcpstat -a -n
1 192.168.2.202:telnet <- 192.168.2.108:40850 ESTABLISHED
   RQ: 0, SQ: 3, RWnd: 32768, SWnd: 1460, SWThresh: 32768
2 192.168.2.202:ftp <- ::: LISTEN
3 192.168.2.202:login <- ::: LISTEN
4 192.168.2.202:ssh <- ::: LISTEN
5 192.168.2.202:telnet <- 192.168.2.108:46873 ESTABLISHED
   RQ: 0, SQ: 0, RWnd: 32768, SWnd: 1460, SWThresh: 32768
7 192.168.2.202:telnet <- 192.168.2.108:44587 ESTABLISHED
   RQ: 0, SQ: 0, RWnd: 32768, SWnd: 1460, SWThresh: 32768
8 192.168.2.202:telnet <- ::: LISTEN
```

Информация по утилите ограничена:

```
# man tcpstat
```

```
man: no manual on tcpstat
```

```
# tcpstat -h
```

```
illegal option -- h
```

```
Usage: tcpstat [-anv]
```

## udpstat

Утилита визуализирует текущие UDP соединения (?):

```
# udpstat -a -n
```

```
0 192.168.2.202:domain -> :::
1 192.168.2.202:syslog -> 127.0.0.1:syslog
2 192.168.2.202:32770 -> :::
```

Информация по утилите ограничена:

```
# man udpstat
```

man: no manual on udpstat

## pr\_routes

Вывод текущего содержимого таблицы маршрутизации, выходная таблица маршрутизации:

```
# pr_routes
ent #   if           dest           gateway dist  pref  mtu flags
    0  ip0       192.168.3.6/32  192.168.3.7   1    0    0 static
    1  ip0         0.0.0.0/0     192.168.3.6   1    0    0 static
```

Входная таблица маршрутизации:

```
# pr_routes -i
ent #   if           dest           gateway dist  pref  mtu flags
```

## add\_route

Ключевая команда в настройке сети MINIX3 — добавление записи маршрутизации:

```
# add_route -g 192.168.3.7 -d 192.168.3.6 -n 255.255.255.255 -I /dev/ip0
# add_route -g 192.168.3.6 -d 0.0.0.0 -n 0.0.0.0
# ping 192.168.3.6
```

192.168.3.6 is alive

- добавлена запись маршрутизации к хосту 192.168.3.6 через интерфейс /dev/ip0 , и адрес 192.168.3.6 добавлен как шлюз по умолчанию.

В терминологии MINIX3 существует 2 таблицы маршрутизации: входная (ключ -i) и выходная (ключ -o или его отсутствие). То, что у них называется входная маршрутизация, скорее соответствует понятию forwarding в Linux или OpenSolaris: непосредственная пересылка пакетов, поступающих с одного сетевого интерфейса, на другой. Вот как это может выглядеть:

```
# add_route -i -g 0.0.0.0 -d 192.168.3.0 -m 1 -n 255.255.255.0 -I /dev/ip0
```

- все пакеты, поступающие на адрес сети 192.168.3.0/24, независимо от интерфейса с которого они поступили (например 192.168.4.0/24 на интерфейсе /dev/ip1, или интерфейс /dev/psip0 на последовательном порту) будут ретранслироваться в интерфейс /dev/ip0 и ARP запросы на разрешение MAC адреса будут направляться в этот интерфейс.

Команды add\_route и del\_route описаны вместе:

```
# man add_route
NAME
    add_route, del_route - configure IP routing.
...
```

## del\_route

Команда, комплиментарная команде add\_route - удаление ранее установленного маршрута из таблицы маршрутизации:

```
# pr_routes
ent #   if           dest           gateway dist  pref  mtu flags
    0  ip0         0.0.0.0/0     192.168.3.7   1    0    0 static
# del_route -g 192.168.3.7
# pr_routes
ent #   if           dest           gateway dist  pref  mtu flags
```

## arp

Манипуляции с ARP таблицей хоста, прямое разрешение Ethernet адресов: IP (32 бит) в MAC (48 бит).

```
# arp -a
192.168.2.1 is at 0:f:34:61:aa:c0
ntc-server.altron.lan (192.168.2.2) is at 0:11:11:87:0:0
opos9.altron.lan (192.168.2.108) is at 0:d0:b7:16:9f:d
^C
```

Информация ограничена:

```
# man arp
man: no manual on arp
# arp
Usage:  arp [-I ip-dev] [-n] hostname
        arp [-I ip-dev] [-n] -a
        arp [-I ip-dev] -d hostname
        arp [-I ip-dev] -d -a
        arp [-I ip-dev] -s hostname ether-addr [temp] [pub]
        arp [-I ip-dev] -S hostname ether-addr [temp] [pub]
```

Исходный код утилиты: /usr/src/commands/simple/arp.c

## rarpd

Демон обратного разрешения адресов MAC (48 бит) в IP (32 бит).

```
# rarpd -d 2 ip0
Usage: rarpd [-d[level]] network-name ...
/dev/eth0: Ethernet address is 52:54:0:12:34:56
/dev/ip0: IP address is 192.168.2.202 / 255.255.255.0
...
^C
```

Описание:

```
# man rarpd
NAME
    rarpd - reverse address resolution protocol daemon
SYNOPSIS
    rarpd [-d[level]] network-name ...
...
```

Использует конфигурационный файл /etc/ethers , который достаточно описан:

```
# man ethers
NAME
    ethers - ethernet address to hostname database
SYNOPSIS
    /etc/ethers
DESCRIPTION
    The ethers database translates ethernet addresses to hostnames for use by
    the RARP daemon (see rarpd(8).) /etc/ethers may look like this:
```

```
0:0:c0:a:77:23      flotsam
0:0:c0:a:68:ce      jetsam
```

...

## irdpd

Демон RIP маршрутизации. При нескольких сетевых интерфейсах, ищет смежные RIP маршрутизаторы, и составляет таблицу оптимальных внешних маршрутов.

```
# irdpd -bd
```

```
Broadcasting router solicitation
```

```
Routing table after advert received from 192.168.2.99:
```

```
192.168.2.99          0 15:30:50
```

```
This router advert is valid until 16:00:50
```

```
Routing table after advert received from 192.168.2.98:
```

```
192.168.2.99          0 15:30:50
```

```
192.168.2.98          0 15:30:50
```

```
This router advert is valid until 16:00:50
```

...

```
^C
```

Достаточно полно описан:

```
# man irdpd
```

```
NAME
```

```
irdpd - internet router discovery protocol daemon
```

```
SYNOPSIS
```

```
irdpd [-bd] [-U udp-device] [-I ip-device] [-o priority-offset]
```

...

## dhcpcd

Клиент и сервер протокола динамической конфигурации DHCP.

Достаточно полно описан:

```
# man dhcpcd
```

```
NAME
```

```
dhcpcd - dynamic host configuration protocol daemon
```

```
SYNOPSIS
```

```
dhcpcd [-qar] [-t[level]] [-d[level]] [-f configfile] [-c cachefile]
[-p poolfile] [host ...]
```

```
DESCRIPTION
```

```
Dhcpcd is a client and a server for the Dynamic Host Configuration Protocol. As a client it collects DHCP data to configure the Ethernet networks with, and as a server it answers DHCP queries from other machines.
```

```
This manual page describes the operation of dhcpcd, the associated configuration file is described in dhcp.conf(5).
```

...

Дополнительная информация:

```
# man boot
```

```
...
```

Использует конфигурационный файл `/etc/dhcp.conf` :

```
# man -s 5 dhcp.conf
```

```
NAME
```

```
dhcp.conf - dynamic host configuration protocol configuration
```

```
...
```

## tcpdump

Эта известнейшая утилита из мира POSIX для анализа сетевого трафика портирована в MINIX3. Её нет (по непонятным причинам) на установочном CD версии 3.1.6, но установочный пакет может быть взят из набора утилит MINIX 3.1.6: <http://www.minix3.org/packages/i386/3.1.6/tcpdump-4.0.0.tar.bz2>

Пакет потребует установки. Скопировав полученный архив, скажем, в `/usr/src`, выполняем:

```
# cd /usr/src
```

```
# bzip2 -d tcpdump-4.0.0.tar.bz2
```

```
# tar -xvf tcpdump-4.0.0.tar
```

```
.minixpackage, 01 tape blocks
```

```
Can't make directory /usr/local/share/man/man1: File exists
```

```
/usr/local/share/man/man1/tcpdump.1, 0151 tape blocks
```

```
Can't make directory /usr/local/sbin: File exists
```

```
/usr/local/sbin/tcpdump, 06533 tape blocks
```

```
/usr/local/sbin/tcpdump.4.0.0, 06533 tape blocks
```

```
# ls -l /usr/local/sbin/
```

```
total 4575
```

```
-rwxr-xr-x 1 root operator 1182264 Nov  5 15:00 sshd
```

```
-rwxr-xr-x 1 root operator 1750090 Nov 11 15:29 tcpdump
```

```
-rwxr-xr-x 1 root operator 1750090 Nov 11 15:29 tcpdump.4.0.0
```

Бинарный пакет установлен:

```
# which tcpdump
```

```
No tcpdump in /root/bin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin
```

```
# man tcpdump
```

```
man: no manual on tcpdump
```

Для него не установлены пути поиска (`$PATH`, `$MANPATH`), можете восполнить это сами, или пользоваться полными путевыми именами при запуске:

```
# man /usr/local/share/man/man1/tcpdump.1
```

```
...
```

```
# /usr/local/sbin/tcpdump -D
```

```
1.eth0
```

**Примечание:** по `tcpdump` существует великое множество обстоятельных описаний в Интернет, в том числе и на русском языке, так что с информацией здесь вопроса не возникает.

Использование `tcpdump` настолько разнообразно, что трудно подобрать краткие примеры для иллюстрации.

Вот выполнение ping на гостевом MINIX3:

```
# ping 192.168.3.7
192.168.3.7 is alive
# ping 192.168.3.6
192.168.3.6 is alive
# ping 192.168.1.7
192.168.1.7 is alive
# ping 192.168.1.5
192.168.1.5 is alive
```

А вот что видит запущенный на этом же хосте tcpdump при каждом из показанных ping (разделительные строки между результатами для каждой команды ping добавлены мною):

```
# /usr/local/sbin/tcpdump -n -vv ip proto \icmp
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 68 bytes
07:47:55.700000 IP (tos 0x0, ttl 255, id 3458, offset 0, flags [none], proto ICMP (1), length 30)
    192.168.3.7 > 192.168.3.6: ICMP echo request, id 0, seq 0, length 10
07:47:56.016666 IP (tos 0x0, ttl 64, id 62455, offset 0, flags [none], proto ICMP (1), length 30)
    192.168.3.6 > 192.168.3.7: ICMP echo reply, id 0, seq 0, length 10

07:48:36.266666 IP (tos 0x0, ttl 255, id 3489, offset 0, flags [none], proto ICMP (1), length 30)
    192.168.3.7 > 192.168.1.7: ICMP echo request, id 0, seq 0, length 10
07:48:36.433333 IP (tos 0x0, ttl 64, id 62456, offset 0, flags [none], proto ICMP (1), length 30)
    192.168.1.7 > 192.168.3.7: ICMP echo reply, id 0, seq 0, length 10

07:48:51.766666 IP (tos 0x0, ttl 255, id 3505, offset 0, flags [none], proto ICMP (1), length 30)
    192.168.3.7 > 192.168.1.5: ICMP echo request, id 0, seq 0, length 10
07:48:52.983333 IP (tos 0x0, ttl 255, id 3508, offset 0, flags [none], proto ICMP (1), length 30)
    192.168.3.7 > 192.168.1.5: ICMP echo request, id 0, seq 0, length 10
07:48:54.166666 IP (tos 0x0, ttl 255, id 3511, offset 0, flags [none], proto ICMP (1), length 30)
    192.168.3.7 > 192.168.1.5: ICMP echo request, id 0, seq 0, length 10
07:48:54.866666 IP (tos 0xc0, ttl 64, id 62457, offset 0, flags [none], proto ICMP (1), length 58)
    192.168.3.6 > 192.168.3.7: ICMP host 192.168.1.5 unreachable, length 38
    IP (tos 0x0, ttl 254, id 3505, offset 0, flags [none], proto ICMP (1), length 30)
    192.168.3.7 > 192.168.1.5: [|icmp]
07:48:54.916666 IP (tos 0xc0, ttl 64, id 62458, offset 0, flags [none], proto ICMP (1), length 58)
    192.168.3.6 > 192.168.3.7: ICMP host 192.168.1.5 unreachable, length 38
    IP (tos 0x0, ttl 254, id 3508, offset 0, flags [none], proto ICMP (1), length 30)
    192.168.3.7 > 192.168.1.5: [|icmp]
07:48:54.950000 IP (tos 0xc0, ttl 64, id 62459, offset 0, flags [none], proto ICMP (1), length 58)
    192.168.3.6 > 192.168.3.7: ICMP host 192.168.1.5 unreachable, length 38
    IP (tos 0x0, ttl 254, id 3511, offset 0, flags [none], proto ICMP (1), length 30)
    192.168.3.7 > 192.168.1.5: [|icmp]
```

192.168.3.7 — запрос к собственному хосту (localhost, 127.0.0.1) — tcpdump не фиксирует какой либо активности на сетевом интерфейсе;

192.168.3.6 — запрос к интерфейсу tap0 базового хоста Linux, видно ICMP запрос и ICMP ответ;

192.168.1.7 — запрос к реальному интерфейсу eth0 базового хоста Linux, также видно ICMP запрос и ICMP ответ;

192.168.1.5 — запрос к другому хосту LAN, в которой находится базовый хост Linux; уходят ICMP запросы, но не возвращаются ICMP ответы: 192.168.1.5 не установлен роутинг в подсеть 192.168.3.0 на IP адрес 192.168.1.7, тем не менее ping MINIX3 сообщает «is alive» (управление роутингом детально рассмотрено в отдельной, 3-й части описания: «Сеть MINIX3. Часть 3: настройки и виртуализация»).

## Дополнительные источники информации

Ссылки на man я даю по размещённым в Интернет файлам для версии MINIX v.2.0.4. Они незначительно отличаются от версии 3; в работающей системе все man можно получить непосредственно в консоли системы (или в терминал удалённого доступа TELNET, RLOGIN, SSH) по имени команды.

1. man ip

<http://minix1.woodhull.com/current/2.0.4/wwwman/man4/ip.4html>

- описание на уровне программного кода, но очень проясняющее понятия сетевого интерфейса, и описывающее аварийные коды завершения, которые можно наблюдать при запуске сетевых программ в неправильных конфигурациях.

2. Описания не документированных команд MINIX3 `add_route`, `pr_routes` с примерами использования:

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

3. man страница по команде `add_route`:

[http://minix1.woodhull.com/current/2.0.4/wwwman/man8/add\\_route.8.html](http://minix1.woodhull.com/current/2.0.4/wwwman/man8/add_route.8.html)

- управление маршрутизацией IP.

4. man страница по команде `pr_routes`:

[http://minix1.woodhull.com/current/2.0.4/wwwman/man8/pr\\_routes.8.html](http://minix1.woodhull.com/current/2.0.4/wwwman/man8/pr_routes.8.html)

- индикация таблицы маршрутизации.

5. «tcpdump - фильтрация при сборе пакетов» :

<http://protocols.ru/modules.php?name=News&file=article&sid=125>

- обстоятельный справочник по параметрам tcpdump.

6. «Руководство по iptables (Iptables Tutorial 1.1.19)»

<http://www.opennet.ru/docs/RUS/iptables/>

Автор: Oskar Andreasson

Перевод: Андрей Киселев

7. «Пускаем QEMU в сеть»

[http://radist-elvin.blogspot.com/2008/07/qemu\\_23.html](http://radist-elvin.blogspot.com/2008/07/qemu_23.html)

- настройка iptables выхода в сеть из виртуальной сети QEMU.

8. «qemu - работа с сетью»

<http://sda00.blogspot.com/2009/05/qemu.html>

- несколько QEMU VM в одной сети.