Сетевой мост для QEMU в Linux.

Часть 1. Создание конфигурационных файлов.

Во первых необходимо включить маршрутизацию между интерфейсами.

$ sudo sysctl net.ipv4.ip_forward=1

Для постоянного включения маршрутизации создайте файл.

$ echo "net.ipv4.ip_forward = 1" **>** /etc/sysctl.d/99-sysctl.conf

Теперь придумайте имя вашего будущего сетевого адаптера для режима **«Мост»**. Пусть у меня будет **«br0»**. Мы с вами будет пользоваться встроенными в систему средствами без каких-либо дополнительных утилит.

С одной стороны — это слегка усложняет настройку, с другой стороны убирает необходимость устанавливать что-либо. Усложняет в том, смысле что, например, графический интерфейс **Network-Manager** не поддерживает создание адаптера для режима мост, только консольные средства.

Возможно для последнего я просто не нашел необходимых настроек, однако, вам всё равно также придётся искать. Команды же намного проще, чем кажутся. Ну да хватит демагогии, переходим к делу.

Теперь необходимо создать файл **«/etc/qemu/bridge.conf»** и записать в него одну строку с названием придуманного вами адаптера режима **«Мост»**. Собственно говоря, адаптер может быть не один. Просто запишите на каждой строке свой адаптер.

$ sudo mkdir -p /etc/qemu/

$ sudo touch /etc/qemu/bridge.conf

$ sudo nano /etc/qemu/bridge.conf

allow br0

Если вам сетевой мост необходим на постоянной основе, то стоит также сделать ручную настройку адресов данного адаптера, **но только после его создания**. Однако, настройку всё равно приведу, чтобы не листать вверх-вниз.

$ sudo nano /etc/network/interfaces

allow-hotplug br0

auto br0

iface br0 inet static

address 192.168.0.120

netmask 255.255.255.0

broadcast 192.168.0.255

gateway 192.168.0.1

dns-nameservers 192.168.0.1 8.8.8.8

pre-up ifconfig br0 hw ether xx:xx:xx:xx:xx:xx

Последняя строка, которая изменяет MAK-адрес не обязательна. Широковещательный адрес рекомендую всё-таки указывать, хоть он и не обязателен. Шлюз указывать обязательно. А вот DNS сервер не обязателен.

Однако, именно эту часть настроек вы можете внести уже графически через **«Network-Manager»**, т.к. после

создания адаптера вручную с указанием его типа он автоматически появляется в списках адаптеров для всех менеджеров сетей. Так что не обязательно делать всё вручную. Вручную можно сделать только часть настроек.

Часть 2. Создаем адаптер и настраиваем. Машрутизация.

Создаем адаптер с типом **«Мост»**.

$ sudo ip link add name br0 type bridge

Включаем его.

$ sudo ip link set br0 up

Далее не обязательно вносить все настройки вручную, через терминал, т.к. после создания адаптера вручную с указанием его типа он автоматически появляется в списках адаптеров для всех менеджеров сетей.

Указываем главный адаптер. У меня это **«enp3s0»**, проводной адаптер. У вас может другой, например, **«eth0»**, или **«wlp4s0»** беспроводной. Просто наберите в терминале команду **«ip link show»**, чтобы узнать названия всех ваших адаптеров и активных соединений (если отсутствует ifconfig — из net-tools утилиты). Если используете WiFi — то название должно начинаться с **«wl»**, а у проводного с **«en»** или с **«et»**.

$ sudo ip link set enp3s0 master br0

Подключаем сетевой мост.

$ sudo bridge link

Теперь настраиваем адреса и не забываем про широковещательный адрес. Я для вычисления

широковещательного адреса использую самый простой калькулятор — консольная утилита **«ipcalc»**.

$ ipcalc 192.168.0.120/24

$ sudo ip addr add 192.168.0.120/24 dev br0

$ sudo ifconfig br0 broadcast 192.168.0.255

$ sudo ifconfig br0 mtu 1000

И самое главное маршрут для этого моста по умолчанию, иначе интернета в **Qemu** не будет. Кстати, эту команду прям непосредственно можно вставить в ручные настройки адресов **«/etc/network/interfaces»**, разумеется без **«sudo»**.

$ sudo ip route add 192.168.0.120 via 192.168.0.1

И не забудьте указать шлюз по умолчанию, а то интернет на основном адаптере пропадет.

$ sudo route add default gw 192.168.0.1 br0

Для просмотра маршрутов по умолчанию:

$ ip route

Часть 3. QEMU

Адаптер сетевого моста указывается так:

-netdev bridge,br=br0,id=net0 \

-device virtio-net-pci,netdev=net0

Чтобы меньше заморачиваться я для себя просто создал скрипт рядом с необходимым образом системы. И в этот скрипт вношу все необходимые команды. Некоторые

записываю в комментарий, чтобы не забыть — просто для удобства, чтобы ничего не искать.

Базовый файл запуска виртуальной машины, например, Archlinux выглядит так — **«qemu.sh»**:

```
#!/usr/bin/env bash

qemu-system-x86_64 \

-enable-kvm \

-cpu host \

-smp cores=1 \

-m 1024 \

-machine q35 \

-device intel-iommu \

-vga virtio \

-netdev bridge,br=br0,id=net0 \

-device virtio-net-pci,netdev=net0 \

-boot menu=on \

-cdrom archlinux-2021.11.01-x86_64.iso \

-hda arch.qcow
```

А далее меняйте в этом скрипте необходимые параметры по своему усмотрению.

Часть 4. Завершение работы с сетевым мостом.

По окончании работы с сетевым мостом необходимо его правильно удалить из системы.

Если сетевой мост в QEMU больше не нужен, то смело удаляйте конфигурацию.

$ sudo rm -rf /etc/qemu/bridge.conf

Отсоединяем основной адаптер от сетевого моста.

$ sudo ip link set enp3s0 nomaster

Отключаем сетевой мост.

$ sudo ip link set br0 down

На всякий случай очищаем маршруты, если таковые остались. Обычно **«Network-Manager»** делает всё за нас в автоматическом режиме. При проблемах — команды ниже.

$ sudo route del default gw 192.168.0.1 br0

$ sudo ip route del 192.168.0.120 via 192.168.0.1

И только после этого, можем его безвредно удалить.

$ sudo ip link delete br0 type bridge

P.S.:

Тест всех соединений в приложенных скриншотах.

Ну а сегодня на этом всё. Надеюсь я хоть немного вас заинтересовал.

Спасибо за внимание. Всем Удачи, до новых встреч, Пока-Пока!

#bash #QEMU #ipcalc #ifconfig #NetTools #ip #iplink #sysctl #bridge #nano

Дополнение статьи: https://vk.com/wall-153221588_7609

Сохранение настроек сетевого моста при перезагрузке.

Для автоматической настройки сетевого моста после перезагрузки нужна утилита bridge-utils.
Без неё сетевой мост необходимо будет каждый раз настраивать заново, иначе даже соединения с интернетом или роутером не будет.
А именно: указать главный адаптер, подключить сетевой мост, и указать маршруты по умолчанию.

```
$ sudo ip link set br0 up
$ sudo ip link set enp3s0 master br0
$ sudo bridge link
$ sudo route add 192.168.0.120 via 192.168.0.1
$ sudo route add default gw 192.168.0.1 br0

$ sudo route del default gw 192.168.0.1 br0
$ sudo route del 192.168.0.120 via 192.168.0.1
```

В Archlinux утилиту можно установить так.

```
$ sudo pacman -S bridge-utils --noconfirm
```

В Debian так.

```
$ sudo su
$ echo 'deb http://ftp.de.debian.org/debian sid main' > /etc/apt/sources.list.d/ftp.de.debian.org.list
$ sudo apt update
$ sudo apt install bridge-utils –y
```

Команды утилиты.

```
# Создать мост.
$ brctl addbr bridge_name
# Добавьте устройство к мосту, например eth0:
$ brctl addif bridge_name eth0
# Показать текущие мосты и к каким интерфейсам они подключены:
$ brctl show
# Настроить мостовое устройство:
$ ip link set dev bridge_name up
# Чтобы удалить мост, вам нужно сначала установить его в положение down:
$ ip link set dev bridge_name down
$ brctl delbr bridge_name
```

Сохранение настроек Network Manager.

Если у вас сетевой менеджер Network Manager, то с сохранением настроек моста при перезагрузке могут возникнуть проблемы. Чтобы это исправить будем вручную управлять настройками всех адаптеров и соединений сетевого менеджера.
Для этого отредактируйте файл конфигурации сетевого менеджера «/etc/NetworkManager/NetworkManager.conf» или создайте отдельную конфигурацию по пути «/etc/NetworkManager/conf.d/».

```
$ sudo nano /etc/NetworkManager/NetworkManager.conf
# или
$ sudo mkdir -p /etc/NetworkManager/conf.d/
$ sudo nano /etc/NetworkManager/conf.d/managed.conf
# Далее просто пропишите секцию и её настройку.
[ifupdown]
managed=true
```

Тем самым мы говорим, чтобы менеджер брал настройки из файла «/etc/network/interfaces».
Также, по умолчанию менеджер каждый раз по активации любого соединения переписывает настройки файла «/etc/resolv.conf». Чтобы это предотвратить и управлять dns-настройками вручную просто создайте конфигурацию по тому же пути: «/etc/NetworkManager/conf.d/dns.conf».

```
$ sudo nano /etc/NetworkManager/conf.d/dns.conf
[main]
dns=none
```

Не забудьте включить или отключить resolvconf.service в system:

```
$ systemctl start resolvconf.service
$ systemctl enable resolvconf.service
# или
$ systemctl stop resolvconf.service
$ systemctl disable resolvconf.service
```

Теперь настройте сетевые интерфейсы «/etc/network/interfaces». Например, адрес моста пусть будет 192.168.0.120, адрес роутера по умолчанию 192.168.0.1, маска 24.
Не забудьте про обязательный loopback интерефейс и автоматические или статические адреса всех ваших сетевых интерфейсов, иначе они даже не запустятся. Включая WiFi интерфейс, например, wlp0s3. У меня в виртуальной машине WiFi нет.

```
$ sudo nano /etc/network/interfaces

auto lo
iface lo inet loopback

allow-hotplug enp0s3
auto enp0s3
# iface enp0s3 inet dhcp

allow-hotplug br0
auto br0
iface br0 inet static
# iface br0 inet dhcp
```

```
address 192.168.0.120
netmask 255.255.255.0
broadcast 192.168.0.255
gateway 192.168.0.1
# dns-nameservers 192.168.0.1 8.8.8.8
# pre-up ifconfig br0 hw ether xx:xx:xx:xx:xx:xx
# pre-up ip link set br0 address xx:xx:xx:xx:xx:xx
bridge_ports eth0
bridge_stp off
bridge_fd 0
bridge_maxwait 0
post-up route add 192.168.0.120 via 192.168.0.1
post-up route add default gw 192.168.0.1 br0
post-down route del default gw 192.168.0.1 br0
post-down route del 192.168.0.120 via 192.168.0.1
```

Строка, которая изменяет MAK-адрес не обязательна, поэтому она закоментирована. Она также указана, как в варианте настройки при помощи утилиты ifconfig, так и при помощи стандартной утилиты ip. Широковещательный адрес рекомендую всё-таки указывать, хоть он и не обязателен. Шлюз указывать обязательно. DNS сервер не обязателен. команды маршрутизации обязательны.
Описание команд конфигурационного файла.

address address - адрес.
netmask netmask - маска сети.
broadcast - широковещательный_адрес.
network network_address - адрес сети.
metric metric - Метрика(целое число).
gateway address - Шлюз по умолчанию.
pointopoint адрес - Адрес удалённой точки.
media type - Тип носителя, зависящий от драйвера.
hwaddress class address - Аппаратный адрес. Класс - это одно из следующих значений: ether, ax25, ARCnet или netrom. Адрес зависит от выбранного класса.
mtu size - Размер MTU.

Описание команд pre-up, post-up, pre-up, post-down.

• pre-up Запустить команду до поднятия интерфейса.
• post-up Запустить команду после поднятия интерфейса.
• pre-down Запустить команду перед отключением интерфейса.
• post-down Запустить команду после отключения интерфейса.

Сохранение настроек Connman.

Если у вас сетевой менеджер connman, то сохранить настройки при перезагрузке здесь будет немного посложнее.
Для установки в Debian воспользуйте следующими командами.

```
$ sudo su
$ echo "deb http://ftp.de.debian.org/debian sid main" >>
/etc/apt/sources.list.d/ftp.de.debian.org.list
$ apt update
$ sudo apt install connman connman-gtk connman-ui -y
```

Для установки в Archlinux:
$ sudo pacman –S connman --noconfirm

Затем внесите изменения в агент и активируйте сервис.

$ sudo connmanctl
agent on
quit
$ sudo systemctl enable connman

Чтобы понять какие настройки примутся после перезагрузки я вносил изменения при помощи графической утилиты «connman-gtk». Настройки находятся примерно здесь: «/var/lib/connman/ethernet_080027a0c313_cable/settings». На каждом ПК путь будет меняться.
Различные варианты настроек и команд можете посмотреть здесь. Самая первая конфигурация – это копия после сохранения из графической утилиты.

Сохранение настроек netctl.

А вот здесь всё гораздо проще и сложнее одновременно.
Сложнее в том, что утилита нормально устанавливается только в Archlinux. Впихнуть её в Debian и другие системы, вроде Fedora мне так и не удалось. Однако, на основе кода из Archlinux я написал свой собственный Makefile для скачивания её из реп-я arch-а и установки в систему. Предусмотрел также и сборку утилиты для упаковки в тот или иной установщик. Имеется возможность установки в любой каталог, а также полного контроля над путями всех скриптов и шаблонов утилиты.
Работу утилиты не тестировал, но установки и де-инсталяции (удаления) проходят корректно в любой ОС Linux.
Настройки и команды утилиты можно посмотреть здесь.

Ну а сегодня на этом всё. Надеюсь я хоть немного вас заинтересовал.
Спасибо за внимание. Всем Удачи, до новых встреч, Пока-Пока!