

Лабораторная Работа 1

де Джофрой Мишель, Белунин Дмитрий Вадимович, Кирдин Даниил Игоревич

Часть 1. Linux (Debian 11)

```
#!/bin/bash

show_network_card_info() {
    printf "\n"
    echo "Network Card Information"
    echo "-----"

    # Get the first active interface with an IP address
    iface=$(ip -o -4 addr list | awk '{print $2}' | head -n 1)

    if [ -z "$iface" ]; then
        echo "No active network interface found!"
        return
    fi

    echo "Interface: $iface"

    ip addr show $iface | grep -E "inet |ether"

    printf "\n"

    lshw_output=$(lshw -class network | grep -A 15 "$iface")

    model=$(echo "$lshw_output" | grep "product:" | awk -F ':' '{print $2}')
    duplex=$(echo "$lshw_output" | grep -oP '(?<=duplex=)[^ ]+')
    speed=$(echo "$lshw_output" | grep -oP '(?<=speed=)[^ ]+')
    link=$(echo "$lshw_output" | grep -oP '(?<=link=)[^ ]+')

    echo "Model: $model"
    echo "Duplex: $duplex"
    echo "Speed: $speed"
    echo "Link: $link"
}

show_ipv4_config() {
    printf "\n"
    echo "IPv4 Configuration"
    echo "-----"

    iface=$(ifconfig | grep -o '^[^ ]*' | head -n 1)

    if [ -z "$iface" ]; then
        echo "No active network interface found!"
    fi
}
```

```
        return
    fi

    ifconfig $iface | grep 'inet ' | awk '{print "IP Address: " $2
"\nNetmask: " $4}'

    route -n | grep 'UG[ \t]' | awk '{print "Gateway: " $2}'

    grep "nameserver" /etc/resolv.conf | awk '{print "DNS: " $2}'
}

configure_scenario_1() {
    printf "\n"
    echo "Configuring network for static IP (Scenario #1)"

    iface=$(ifconfig | grep -o '^[^ ]*' | head -n 1)

    if [ -z "$iface" ]; then
        echo "No active network interface found!"
        return
    fi

    ifconfig $iface 10.100.0.2 netmask 255.255.255.0 up

    route add default gw 10.100.0.1

    echo "nameserver 8.8.8.8" | tee /etc/resolv.conf > /dev/null

    echo "Static network configuration applied for Scenario #1"
}

configure_scenario_2() {
    printf "\n"
    echo "Configuring network for dynamic IP (Scenario #2)"

    iface=$(ifconfig | grep -o '^[^ ]*' | head -n 1)

    if [ -z "$iface" ]; then
        echo "No active network interface found!"
        return
    fi

    ifconfig $iface down
    ifconfig $iface up
    dhclient $iface

    echo "Dynamic network configuration (DHCP) applied for Scenario #2"
}

# Menu template
while true; do
    echo ""
    echo "Select an option:"
    echo "1. Show network card details"
```

```
echo "2. Show current IPv4 configuration"
echo "3. Configure network (Scenario #1)"
echo "4. Configure network (Scenario #2)"
echo "5. Exit"
echo ""

read -p "Enter your choice: " choice

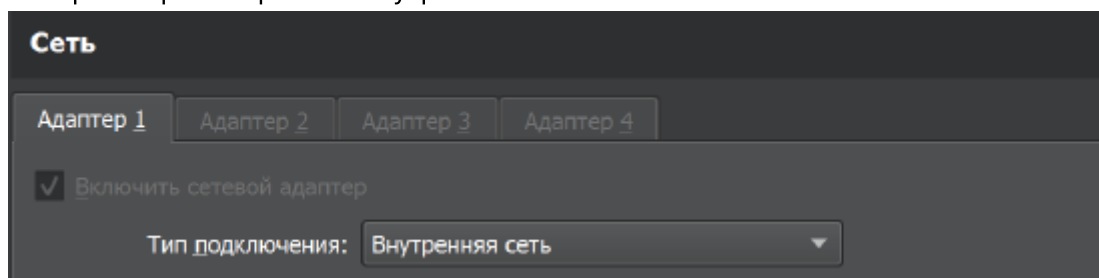
case $choice in
    1)
        show_network_card_info
        ;;
    2)
        show_ipv4_config
        ;;
    3)
        configure_scenario_1
        ;;
    4)
        configure_scenario_2
        ;;
    5)
        echo "Exiting..."
        exit 0
        ;;
    *)
        echo "Invalid choice. Please try again."
        ;;
esac
done
```

Часть 2. Работа с виртуальными интерфейсами Linux CentOS через Network Manager

Сценарии, используемые в работе:

- Статическая адресация **IP**=10.100.0.2, **MASK**=255.255.255.0, **GATE**=10.100.0.1, **DNS** = 8.8.8.8
- Динамическая – все параметры получаются автоматически с **dhcp** сервера.

1. Настроили режим работы внутренняя сеть в Virtual Box:




2. Настроили сетевой интерфейс с именем **enp0s3** по сценарию 1

```
nmcli connection add
type ethernet
con-name static-connection
ifname enp0s3
ipv4.address 10.100.0.2/24
ipv4.gateway 10.100.0.1
ipv4.dns 8.8.8.8
ipv4.method manual
```

Активировали по команде

```
nmcli connection up static-connection
```



```
CentOS9 [Pa66raer] - Oracle VM VirtualBox
Connection.auth-retries: -1
Connection.timestamp: 1725383112
Connection.permissions: --
Connection.zone: --
Connection.controller: --
Connection.master: --
Connection.slave-type: --
Connection.port-type: --
Connection.autoconnect-slaves: -1 (default)
Connection.autoconnect-ports: -1 (default)
Connection.down-on-poweroff: -1 (default)
Connection.secondaries: --
Connection.gateway-ping-timeout: 0
Connection.metered: неизвестно
Connection.lldp: default
Connection.mdns: -1 (default)
Connection.lldmr: -1 (default)
Connection.dns-over-tls: -1 (default)
Connection.mptcp-flags: 0x0 (default)
Connection.wait-device-timeout: -1
Connection.wait-activation-delay: -1
802-3-ethernet.port: --
802-3-ethernet.speed: 0
802-3-ethernet.duplex: --
802-3-ethernet.auto-negotiate: нет
802-3-ethernet.mac-address: --
802-3-ethernet.cloned-mac-address: --
802-3-ethernet.generate-mac-address-mask: --
802-3-ethernet.mac-address-denylist: --
802-3-ethernet.mtu: автоматически
802-3-ethernet.s390-subchannels: --
802-3-ethernet.s390-nettype: --
802-3-ethernet.s390-options: --
802-3-ethernet.wake-on-lan: default
802-3-ethernet.wake-on-lan-password: --
802-3-ethernet.accept-all-mac-addresses: -1 (default)
ipv4.method: auto
ipv4.dns: 8.8.8.8
ipv4.dns-search: --
ipv4.dns-options: --
ipv4.dns-priority: 0
ipv4.addresses: 10.100.0.2/24
ipv4.gateway: 10.100.0.1
ipv4.routes: --
ipv4.route-metric: -1
ipv4.route-table: 0 (unspec)
ipv4.routing-rules: --
ipv4.replace-local-rule: -1 (default)
ipv4.dhcp-send-release: -1 (default)
root@localhost ~#
```

(Что значит enp0s3:) enp0s3: The network interface name as a string. The "en" stands for ethernet, "p0" is the bus number of the ethernet card, and "s3" is the slot number. 3. Создали виртуальный сетевой интерфейс с именем **br0**

```
nmcli connection add
type dummy
con-name br0
ifname br0
ipv4.address 10.100.0.3/24
ipv4.method manual
```

(Что значит dummy interface:) As a Red Hat Enterprise Linux user, you can create and use dummy network interfaces for debugging and testing purposes. A dummy interface provides a device to route packets without actually transmitting them. It enables you to create additional loopback-like devices managed by NetworkManager and makes an inactive SLIP (Serial Line Internet Protocol) address look like a real address for local programs.

4. Активировали сетевой интерфейс по команде

```
nmcli connection up br0
```

5. С помощью команды ping проверили связь между реальным и виртуальным сетевым интерфейсом

```
[root@localhost ~]# ping 10.100.0.3
PING 10.100.0.3 (10.100.0.3) 56(84) bytes of data.
64 bytes from 10.100.0.3: icmp_seq=1 ttl=64 time=0.961 ms
64 bytes from 10.100.0.3: icmp_seq=2 ttl=64 time=2.07 ms
64 bytes from 10.100.0.3: icmp_seq=3 ttl=64 time=0.092 ms
64 bytes from 10.100.0.3: icmp_seq=4 ttl=64 time=0.118 ms
64 bytes from 10.100.0.3: icmp_seq=5 ttl=64 time=0.079 ms
64 bytes from 10.100.0.3: icmp_seq=6 ttl=64 time=0.075 ms
64 bytes from 10.100.0.3: icmp_seq=7 ttl=64 time=0.082 ms
64 bytes from 10.100.0.3: icmp_seq=8 ttl=64 time=0.180 ms
64 bytes from 10.100.0.3: icmp_seq=9 ttl=64 time=0.081 ms
64 bytes from 10.100.0.3: icmp_seq=10 ttl=64 time=0.101 ms
64 bytes from 10.100.0.3: icmp_seq=11 ttl=64 time=0.098 ms
64 bytes from 10.100.0.3: icmp_seq=12 ttl=64 time=0.051 ms
64 bytes from 10.100.0.3: icmp_seq=13 ttl=64 time=0.079 ms
64 bytes from 10.100.0.3: icmp_seq=14 ttl=64 time=0.104 ms
64 bytes from 10.100.0.3: icmp_seq=15 ttl=64 time=0.109 ms
64 bytes from 10.100.0.3: icmp_seq=16 ttl=64 time=0.154 ms
64 bytes from 10.100.0.3: icmp_seq=17 ttl=64 time=0.099 ms
64 bytes from 10.100.0.3: icmp_seq=18 ttl=64 time=0.100 ms
64 bytes from 10.100.0.3: icmp_seq=19 ttl=64 time=0.101 ms
64 bytes from 10.100.0.3: icmp_seq=20 ttl=64 time=0.079 ms
64 bytes from 10.100.0.3: icmp_seq=21 ttl=64 time=0.112 ms
64 bytes from 10.100.0.3: icmp_seq=22 ttl=64 time=0.077 ms
64 bytes from 10.100.0.3: icmp_seq=23 ttl=64 time=0.095 ms
64 bytes from 10.100.0.3: icmp_seq=24 ttl=64 time=0.147 ms
64 bytes from 10.100.0.3: icmp_seq=25 ttl=64 time=0.101 ms
64 bytes from 10.100.0.3: icmp_seq=26 ttl=64 time=0.136 ms
64 bytes from 10.100.0.3: icmp_seq=27 ttl=64 time=0.101 ms
64 bytes from 10.100.0.3: icmp_seq=28 ttl=64 time=0.091 ms
64 bytes from 10.100.0.3: icmp_seq=29 ttl=64 time=0.078 ms
64 bytes from 10.100.0.3: icmp_seq=30 ttl=64 time=0.092 ms
64 bytes from 10.100.0.3: icmp_seq=31 ttl=64 time=0.077 ms
^I64 bytes from 10.100.0.3: icmp_seq=32 ttl=64 time=0.079 ms
64 bytes from 10.100.0.3: icmp_seq=33 ttl=64 time=0.095 ms
64 bytes from 10.100.0.3: icmp_seq=34 ttl=64 time=0.141 ms
64 bytes from 10.100.0.3: icmp_seq=35 ttl=64 time=0.080 ms
^C
--- 10.100.0.3 ping statistics ---
35 packets transmitted, 35 received, 0% packet loss, time 34806ms
rtt min/avg/max/mdev = 0.051/0.180/2.069/0.355 ms
```

(Что значит проверить связь реального и виртуального сетевого интерфейса в контексте этой лабы): В скриншоте где мы добавляли enp0s3 девайс, мы в тот момент находились относительно его айпи адреса. Соответственно когда мы пинговали адрес 10.100.0.3 - мы из реального девайса пинговали виртуальный (пакеты проходили - значит связь есть)

6. С помощью команды `ip link show` узнали MAC-адрес виртуального интерфейса

```
root@localhost ~]# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether 08:00:27:9a:a4:ef brd ff:ff:ff:ff:ff:ff
3: br0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode DEFAULT group default qlen 1000
    link/ether da:40:a2:8b:2b:2b brd ff:ff:ff:ff:ff:ff
```

Часть 3. Работа с реальными интерфейсами Linux Debian 11 через netplan

1. Добавили YAML файл 01-network-manager-all.yaml в директорию /etc/netplan Содержимое файла:

```
network:
  version: 2
  ethernets:
    enp0s3:
      addresses:
        - 10.100.0.4/24
        - 10.100.0.5/24
      routes:
        - to: 0.0.0.0/0
          via: 10.100.0.3
```

Применили конфигурацию по команде `netplan apply`

2. Проверили связь ip адресов 10.100.0.2-10.100.0.5

```
PING 10.100.0.2 (10.100.0.2) 56(84) bytes of data.  
64 bytes from 10.100.0.2: icmp_seq=1 ttl=64 time=55.9 ms  
64 bytes from 10.100.0.2: icmp_seq=2 ttl=64 time=0.693 ms  
64 bytes from 10.100.0.2: icmp_seq=3 ttl=64 time=0.703 ms  
64 bytes from 10.100.0.2: icmp_seq=4 ttl=64 time=0.826 ms  
^Z  
[1]+  Остановлен    ping 10.100.0.2  
root@d12:/etc/netplan# ping 10.100.0.3  
PING 10.100.0.3 (10.100.0.3) 56(84) bytes of data.  
64 bytes from 10.100.0.3: icmp_seq=1 ttl=64 time=0.847 ms  
64 bytes from 10.100.0.3: icmp_seq=2 ttl=64 time=0.671 ms  
64 bytes from 10.100.0.3: icmp_seq=3 ttl=64 time=0.801 ms  
64 bytes from 10.100.0.3: icmp_seq=4 ttl=64 time=1.63 ms  
^Z  
[2]+  Остановлен    ping 10.100.0.3  
root@d12:/etc/netplan# ping 10.100.0.4  
PING 10.100.0.4 (10.100.0.4) 56(84) bytes of data.  
64 bytes from 10.100.0.4: icmp_seq=1 ttl=64 time=0.040 ms  
64 bytes from 10.100.0.4: icmp_seq=2 ttl=64 time=0.041 ms  
64 bytes from 10.100.0.4: icmp_seq=3 ttl=64 time=0.040 ms  
64 bytes from 10.100.0.4: icmp_seq=4 ttl=64 time=0.040 ms  
^Z  
[3]+  Остановлен    ping 10.100.0.4  
root@d12:/etc/netplan# ping 10.100.0.5  
PING 10.100.0.5 (10.100.0.5) 56(84) bytes of data.  
64 bytes from 10.100.0.5: icmp_seq=1 ttl=64 time=0.027 ms  
64 bytes from 10.100.0.5: icmp_seq=2 ttl=64 time=0.052 ms  
64 bytes from 10.100.0.5: icmp_seq=3 ttl=64 time=0.080 ms  
64 bytes from 10.100.0.5: icmp_seq=4 ttl=64 time=0.048 ms  
64 bytes from 10.100.0.5: icmp_seq=5 ttl=64 time=0.044 ms  
^Z  
[4]+  Остановлен    ping 10.100.0.5
```

3. Выводим arp кэш по команде `ip neigh`

```
[root@localhost ~]# ip neigh  
10.100.0.4 dev enp0s3 lladdr 08:00:27:c3:04:a5 STALE  
10.100.0.1 dev enp0s3 FAILED  
169.254.8.124 dev enp0s3 lladdr 08:00:27:c3:04:a5 STALE
```

Часть 4. Настройка объединения реальных сетевых интерфейсов в Linux

```
flsmode | grep bonding
```

```
modprobe bonding
```

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: no
    enp0s8:
      dhcp4: no
  bonds:
    bond007:
      dhcp4: yes
      interfaces:
        - enp0s3
        - enp0s8
      parameters:
        mode: balance-rr
        primary: enp0s3
```

```
netplan apply
```



```

root@d12:/etc/netplan# cat /proc/net/dev
Inter-|   Receive   |                               | Transmit
face |bytes    packets errs drop fifo frame compressed multicast|bytes    packets errs drop fifo colls carrier compressed
---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
lo:   32300      340    0    0    0    0      0      0    0   32300      340    0    0    0    0    0    0    0
enp0s3:    0        0    0    0    0    0      0      0   35080      505    0    0    0    0    0    0    0
enp0s8:   2480        6    0    0    0    0      0      0    4436      54    0    0    0    0    0    0    0
bond007:  2360        4    0    0    0    0      0      0    6026      61    0    9    0    0    0    0    0
root@d12:/etc/netplan#

```

```

root@d12:/etc/netplan# cat /proc/net/bonding/bond007
Ethernet Channel Bonding Driver: v6.1.0-25-amd64

```

```

Bonding Mode: load balancing (round-robin)

```

```

MII Status: up

```

```

MII Polling Interval (ms): 0

```

```

Up Delay (ms): 0

```

```

Down Delay (ms): 0

```

```

Peer Notification Delay (ms): 0

```

```

Slave Interface: enp0s8

```

```

MII Status: up

```

```

Speed: 1000 Mbps

```

```

Duplex: full

```

```

Link Failure Count: 0

```

```

Permanent HW addr: 08:00:27:90:c5:47

```

```

Slave queue ID: 0

```

```

Slave Interface: enp0s3

```

```

MII Status: up

```

```

Speed: 1000 Mbps

```

```

Duplex: full

```

```

Link Failure Count: 0

```

```

Permanent HW addr: 08:00:27:c3:04:a5

```

```

Slave queue ID: 0

```

```
#!/bin/bash
```

```
while true; do
```

```
    awk 'NR>2 {iface=$1; gsub(":", "", iface); rx=$2; tx=$10; print "
["strftime("%Y-%m-%d %H:%M:%S")] Interface: "    iface ", RX: " rx ", TX:
" tx}' /proc/net/dev
```

```
    sleep 5
```

```
done
```

```
root@d12:~# bash script2.sh
[2024-10-10 22:12:56] Interface: lo, RX:69294, TX:69294
[2024-10-10 22:12:56] Interface: enp0s3, RX:22888, TX:28272
[2024-10-10 22:12:56] Interface: enp0s8, RX:2480, TX:46654
[2024-10-10 22:12:56] Interface: bond007, RX:2360, TX:46654
[2024-10-10 22:13:01] Interface: lo, RX:69674, TX:69674
[2024-10-10 22:13:01] Interface: enp0s3, RX:23378, TX:28762
[2024-10-10 22:13:01] Interface: enp0s8, RX:2480, TX:46954
[2024-10-10 22:13:01] Interface: bond007, RX:2360, TX:46954
[2024-10-10 22:13:06] Interface: lo, RX:70244, TX:70244
[2024-10-10 22:13:06] Interface: enp0s3, RX:23868, TX:29252
[2024-10-10 22:13:06] Interface: enp0s8, RX:2480, TX:47254
[2024-10-10 22:13:06] Interface: bond007, RX:2360, TX:47254
[2024-10-10 22:13:11] Interface: lo, RX:70814, TX:70814
[2024-10-10 22:13:11] Interface: enp0s3, RX:24358, TX:29742
[2024-10-10 22:13:11] Interface: enp0s8, RX:2480, TX:47554
[2024-10-10 22:13:11] Interface: bond007, RX:2360, TX:47554
[2024-10-10 22:13:16] Interface: lo, RX:71004, TX:71004
[2024-10-10 22:13:16] Interface: enp0s3, RX:24908, TX:30292
[2024-10-10 22:13:16] Interface: enp0s8, RX:2480, TX:47794
[2024-10-10 22:13:16] Interface: bond007, RX:2360, TX:47794
[2024-10-10 22:13:21] Interface: lo, RX:71574, TX:71574
[2024-10-10 22:13:21] Interface: enp0s3, RX:25398, TX:30782
[2024-10-10 22:13:21] Interface: enp0s8, RX:2480, TX:48094
[2024-10-10 22:13:21] Interface: bond007, RX:2360, TX:48094
[2024-10-10 22:13:26] Interface: lo, RX:71954, TX:71954
[2024-10-10 22:13:26] Interface: enp0s3, RX:25888, TX:31272
[2024-10-10 22:13:26] Interface: enp0s8, RX:2480, TX:48394
[2024-10-10 22:13:26] Interface: bond007, RX:2360, TX:48394
```

QA

С помощью утилиты `ip`:

1. *назначить новый IPv4 адрес:* `ip addr add xx.xxx.x.x/yy dev enp0s3`
2. *назначить новый MAC адрес:* Сначала нужно временно остановить устройство, затем назначить новый MAC-адрес и потом снова поднять

```
sudo ip link set dev wlan1 down
sudo ip link set dev wlan1 address 12:e1:7d:f8:a3:e5
sudo ip link set dev wlan1 up
```

3. *назначить новый gateway:* Сначала удаляем текущий, затем добавляем новый

```
route del default
sudo route delete default gw IP-Address Adapter
sudo route add default gw IP-Address Adapter
```

В случае, если несколько дефолтных gateway:

```
sudo route delete default gw IP-Address Adapter
```

4. вывести информацию arp кэше: `ip neigh`
5. очистить arp кэш: `ip -s -s neigh flush all` Двойной флаг `-s` означает удаление и ARP и NDISC кэша
6. включить интерфейс: `nmcli connection up DEVICE_NAME`
7. выключить интерфейс: `nmcli connection down DEVICE_NAME`
8. Как с помощью `nmcli` назначить на интерфейс статический IP адрес, маску и настроить default gateway:
9. Как с помощью `netplan` назначить на интерфейс статический IP адрес, маску и настроить default gateway: `nmcli con add type ethernet ifname enp0s3 con-name CONNECTION_NAME ip4 192.168.1.100/24 gw4 192.168.1.1`
10. Какие режимы *bonding* стандартно существуют в Linux? Опишите их назначение, возможности по отказоустойчивости и необходимость поддержки со стороны оборудования: Обеспечение отказоустойчивости и увеличение пропускной способности в результате объединения сетевых интерфейсов и передачи/принятия пакетов по определённым стратегиям - либо "по кругу", при отказе 1 узла остальные продолжают функционировать, либо один работает, остальные просто на за замену (резервные), либо трафик идёт на все подряд - broadcast (теряем смысл увел пропускной способности), либо трафик распределяется по кэшу относительно мак адресов устройств.
11. Какие существуют и чем отличаются режимы работы адаптера (*duplex*): **Full-duplex mode** (full). Interfaces operating in this mode can send and receive packets simultaneously. **Half-duplex mode** (half). Interfaces operating in this mode cannot send and receive simultaneously. **Auto-negotiation mode** (auto). Interfaces operating in this mode negotiate a duplex mode with their peers.
12. Какой, по-вашему, практический смысл в возможности назначения нескольких IP адресов на один интерфейс: Для того, чтобы сервер мог принимать запросы от нескольких приложений
13. Какой, по-вашему, практический смысл в возможности создания виртуальных интерфейсов?: As a Red Hat Enterprise Linux user, you can create and use dummy network interfaces for debugging and testing purposes. A dummy interface provides a device to route packets without actually transmitting them. It enables you to create additional loopback-like devices managed by NetworkManager and makes an inactive SLIP (Serial Line Internet Protocol) address look like a real address for local programs.