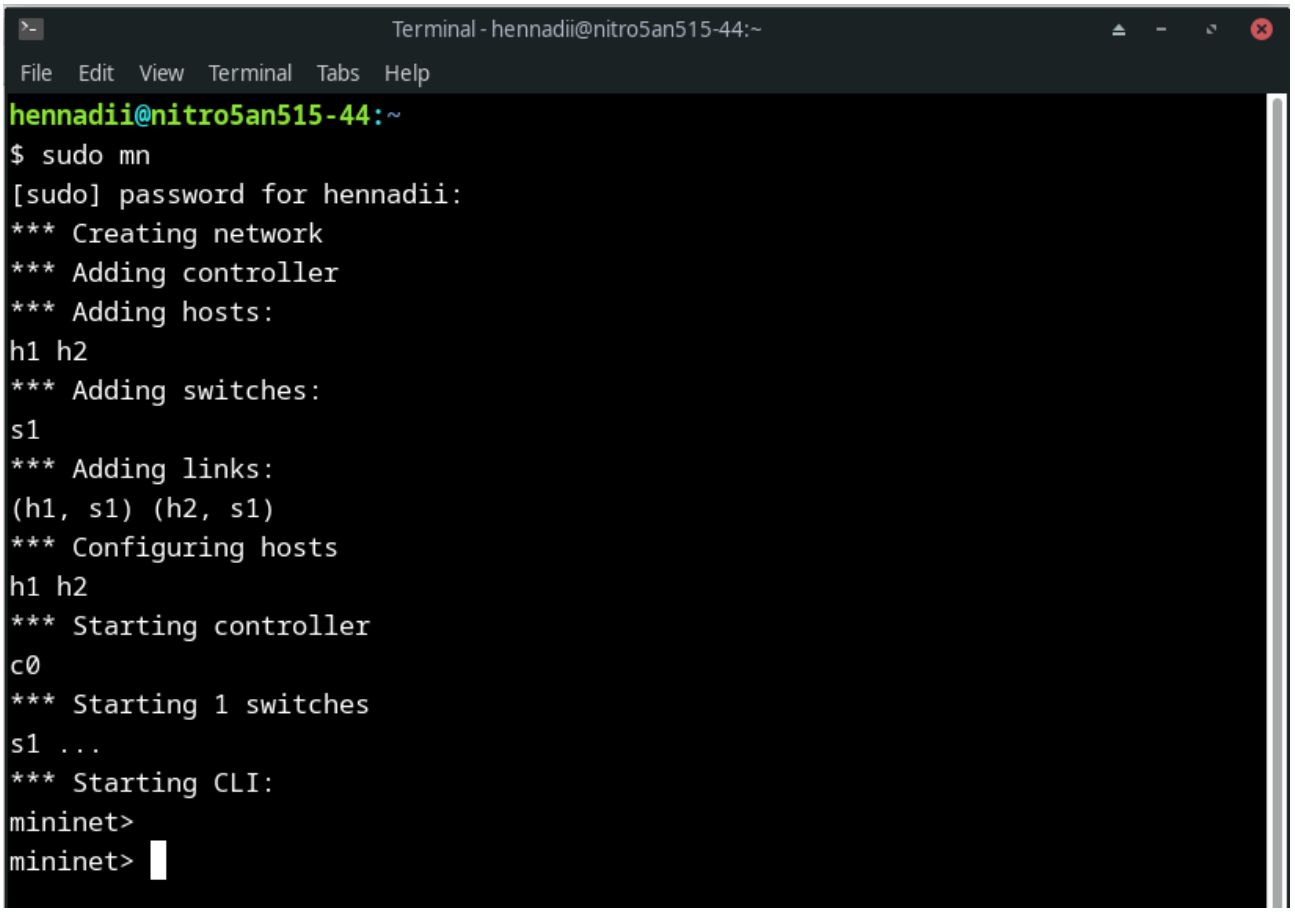
Виконав: Кочев Геннадій, ІП-91, https://t.me/vimacs

Розділ 1 - Вступ до Mininet

Запустимо mininet з конфігурацією за замовчуванням:



Команда help:

```
mininet>
mininet> help

Documented commands (type help <topic>):
========================================
EOF    gterm  iperfudp  nodes        pingpair      py      switch  xterm
dpctl  help   link      noecho       pingpairfull  quit    time
dump   intfs  links     pingall      ports         sh      wait
exit   iperf  net       pingallfull  px            source  x

You may also send a command to a node using:
  <node> command {args}
For example:
  mininet> h1 ifconfig

The interpreter automatically substitutes IP addresses
for node names when a node is the first arg, so commands
like
  mininet> h2 ping h3
should work.

Some character-oriented interactive commands require
noecho:
  mininet> noecho h2 vi foo.py
However, starting up an xterm/gterm is generally better:
  mininet> xterm h2

mininet>
```

Доступні вузли:

```
mininet> nodes
available nodes are:
c0 h1 h2 s1
mininet>
```

Доступні посилання:

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
mininet>
```

Виконаємо команду Linux на h1 для того щоб побачити інтерфейси хоста h1:

```
mininet> h1 ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.1  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::e469:95ff:fe0d:1621  prefixlen 64  scopeid 0x20<link>
        ether e6:69:95:0d:16:21  txqueuelen 1000  (Ethernet)
        RX packets 33  bytes 3406 (3.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 13  bytes 1006 (1006.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

mininet>
```

adii@nitro... Mon 3 Oct, 20:38

Перевіряємо підключення h1 до h2:

```
mininet> h1 ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=2.72 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.826 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.126 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.106 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.133 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.174 ms
^C
--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5043ms
rtt min/avg/max/mdev = 0.106/0.680/2.715/0.944 ms
mininet>
```

nadii@nitro... Mon 3 Oct, 20:40

Зупинимо емуляцію:

```
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 1279.288 seconds
hennadii@nitro5an515-44:~
$
```

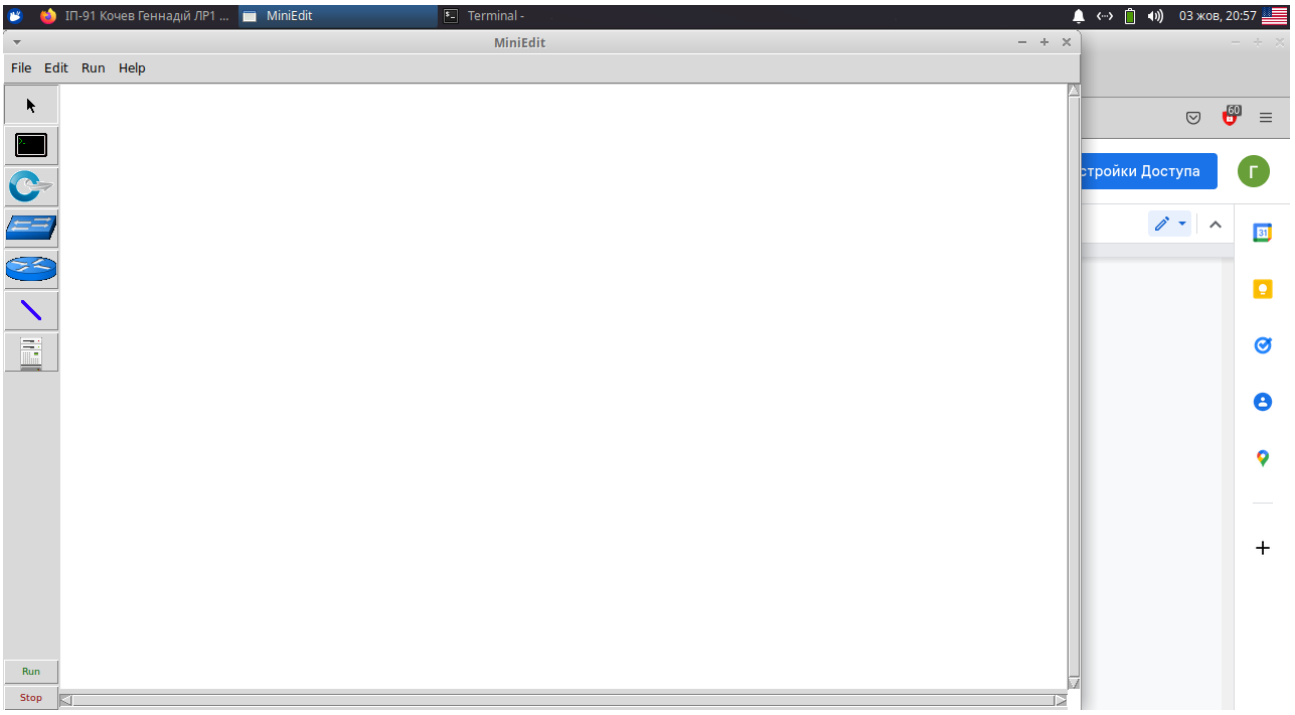
nadii@nitro... Mon 3 Oct, 20:44

Запустимо Miniedit на віртуальній машині Xubuntu.

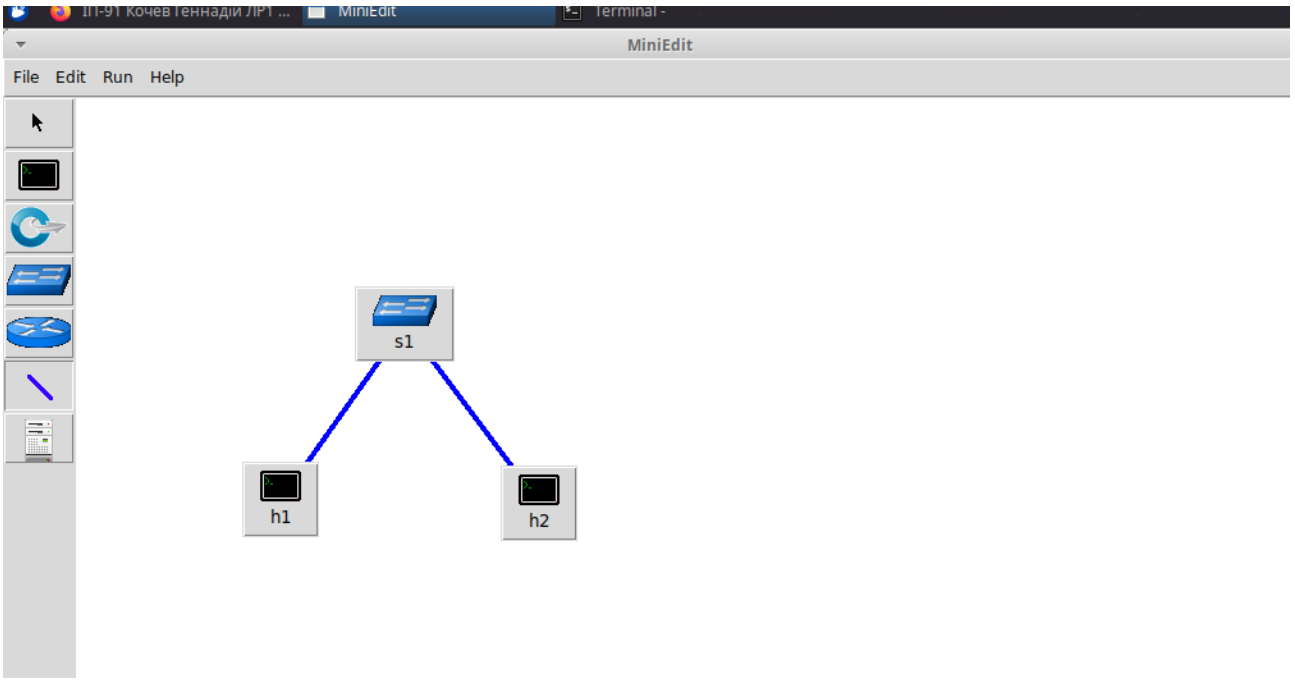
Запустимо Miniedit командою:

```
hennadii@asus:~
$ cd /usr/lib/python2.7/dist-packages/mininet/examples/
hennadii@asus:/usr/lib/python2.7/dist-packages/mininet/examples
$ ls
baresshd.py        consoles.py       hwintf.py           miniedit.py
baresshd.pyc       consoles.pyc      hwintf.pyc          miniedit.pyc
bind.py            controllers2.py   __init__.py         mobility.py
bind.pyc           controllers2.pyc  __init__.pyc        mobility.pyc
clustercli.py      controllers.py    intfoptions.py      multilink.py
clustercli.pyc     controllers.pyc   intfoptions.pyc     multilink.pyc
clusterdemo.py     controlnet.py     limit.py            multiping.py
clusterdemo.pyc    controlnet.pyc    limit.pyc           multiping.pyc
cluster.py         cpu.py            linearbandwidth.py  multipoll.py
cluster.pyc        cpu.pyc           linearbandwidth.pyc multipoll.pyc
clusterSanity.py   emptynet.py       linuxrouter.py      multitest.py
clusterSanity.pyc  emptynet.pyc      linuxrouter.pyc     multitest.pyc
hennadii@asus:/usr/lib/python2.7/dist-packages/mininet/examples
$ python miniedit.py
MiniEdit running against Mininet 2.2.2
topo=none
```

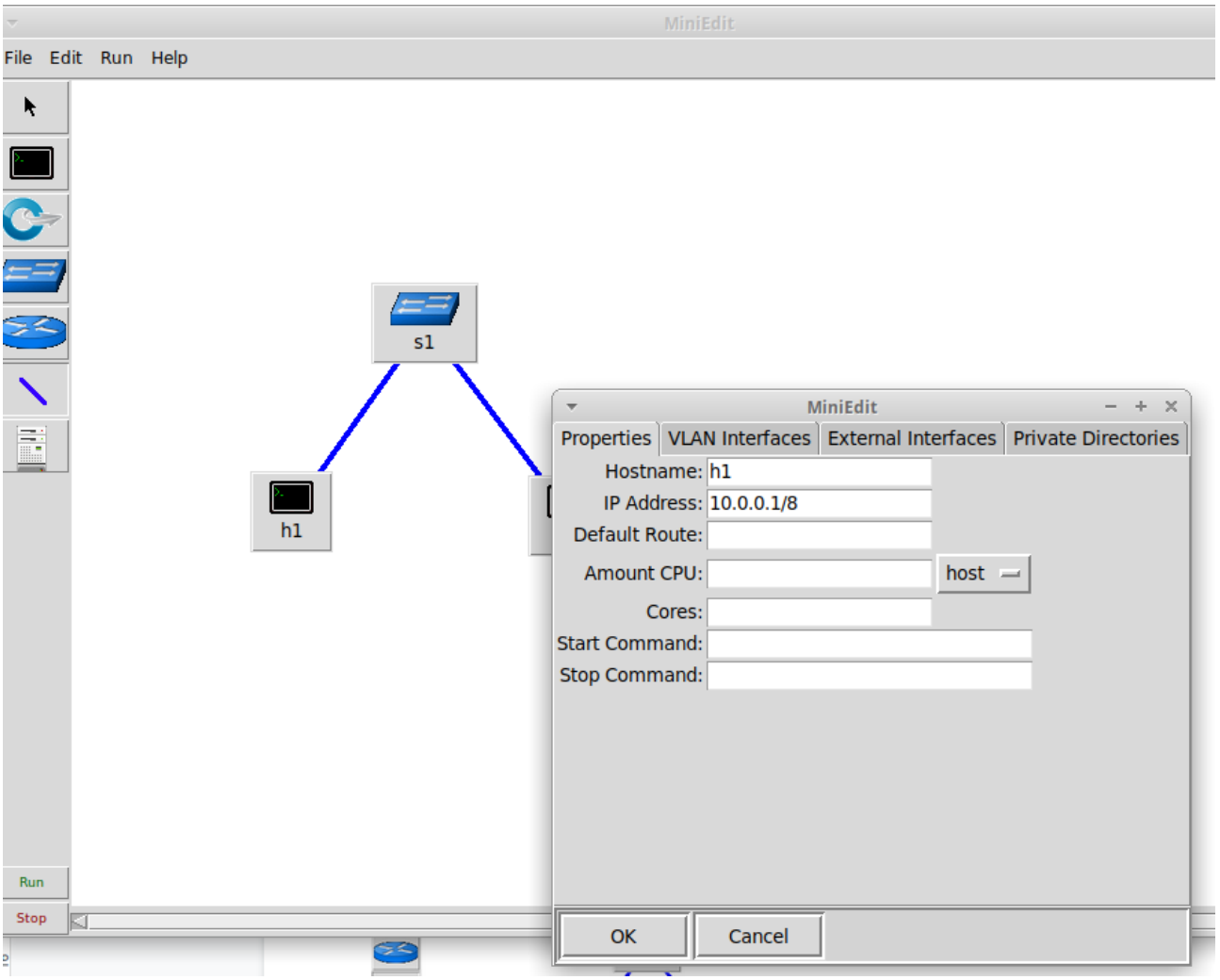
Miniedit екран:

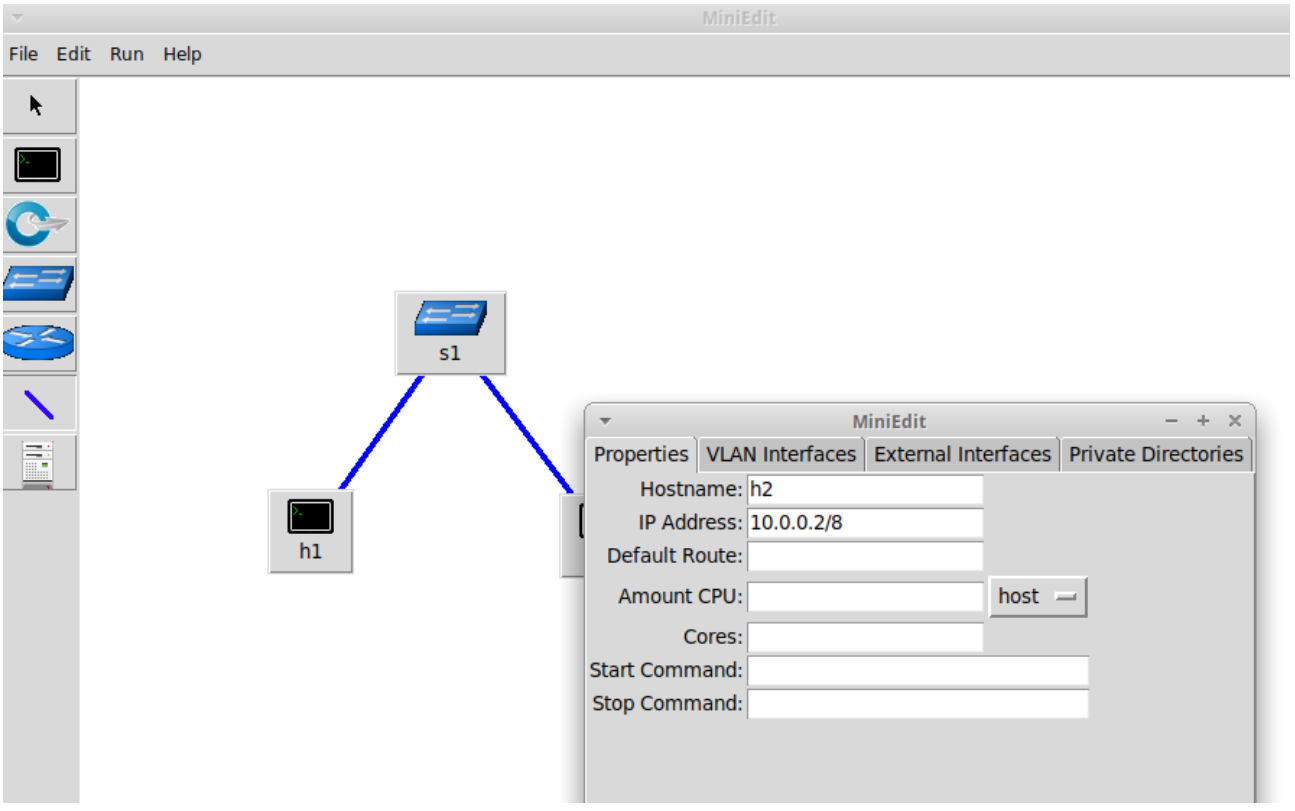


Створимо базову топологію:

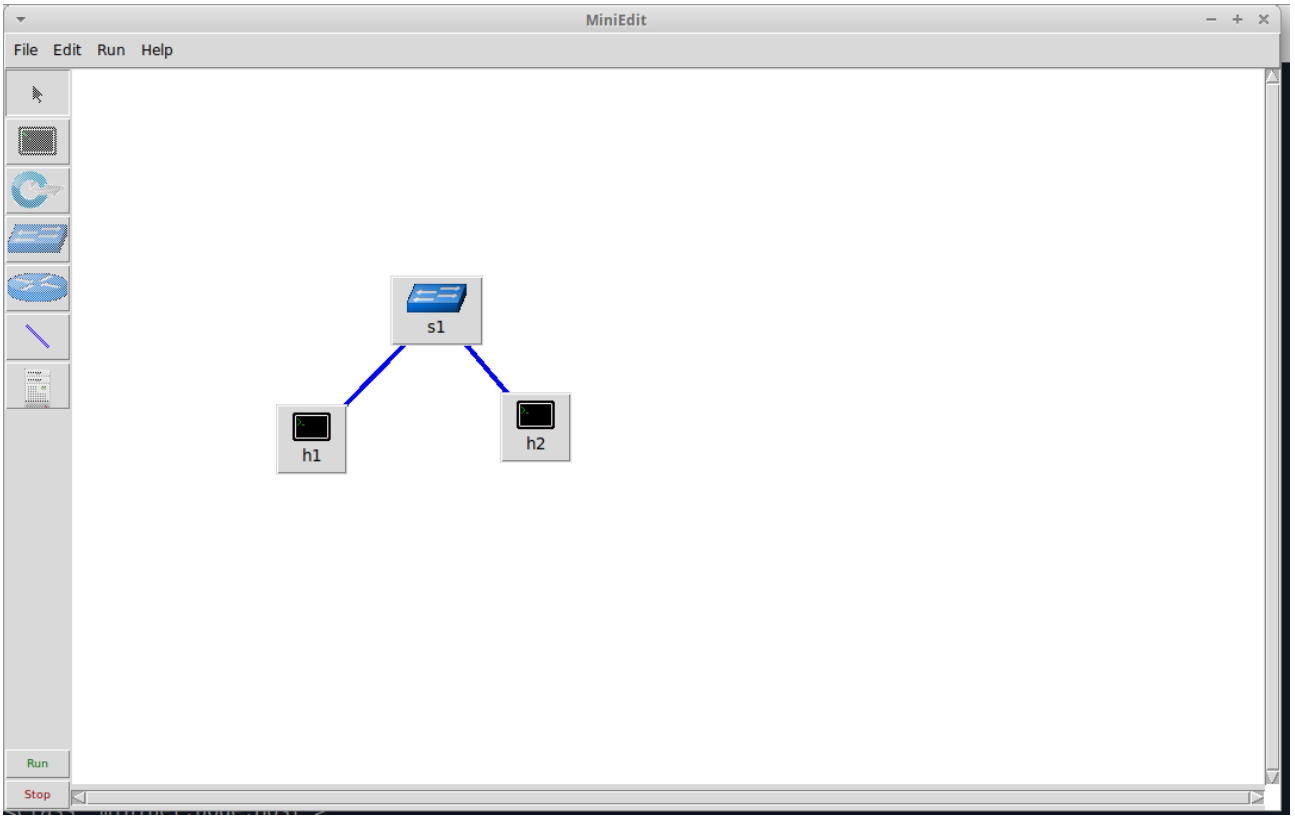


Встановимо IP-адреси хостів

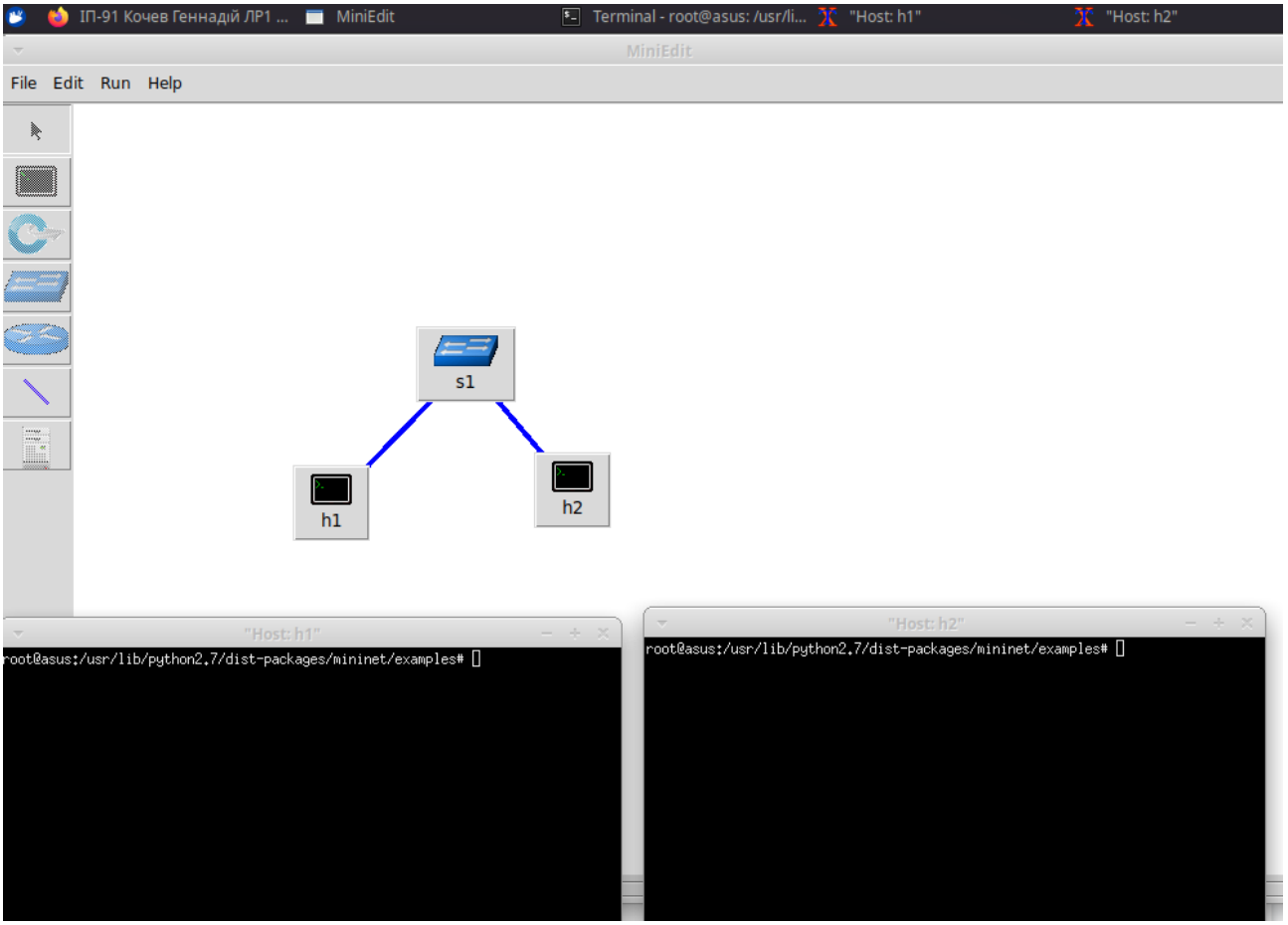

MiniEdit
File Edit Run Help
s1
h1
MiniEdit
Properties
VLAN Interfaces
External Interfaces
Private Directories
Hostname: h1
IP Address: 10.0.0.1/8
Default Route:
Amount CPU:
host
Cores:
Start Command:
Stop Command:
Run
Stop
OK
Cancel


MiniEdit
File Edit Run Help
s1
h1
MiniEdit
Properties
VLAN Interfaces
External Interfaces
Private Directories
Hostname: h2
IP Address: 10.0.0.2/8
Default Route:
Amount CPU:
host
Cores:
Start Command:
Stop Command:

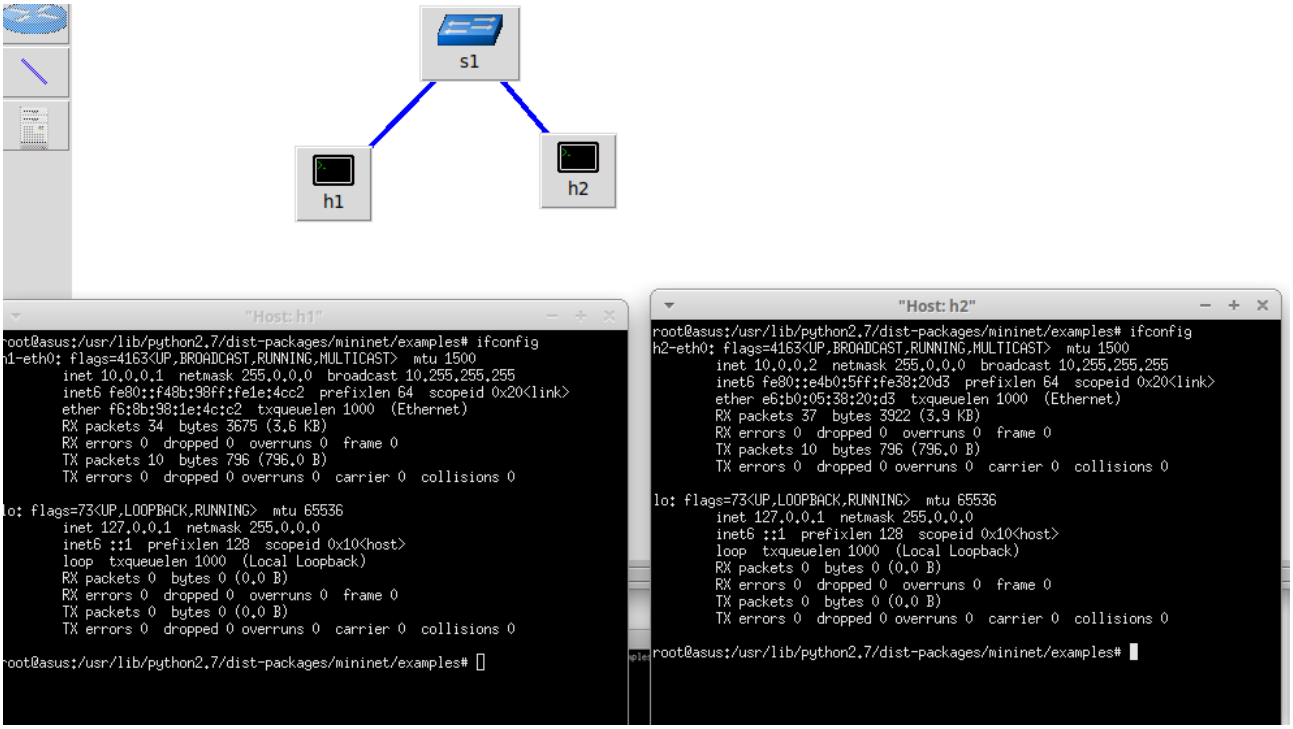
Для запуску поточної конфігурації, запустимо mininet як адміністратор (root)



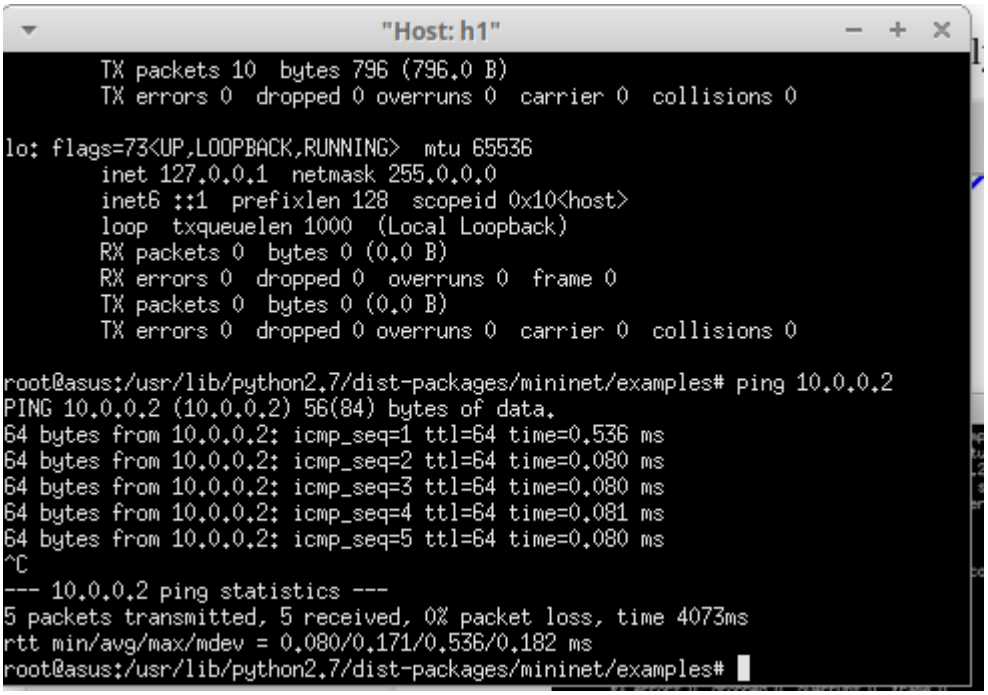
Відкриємо два термінали хостів h1, h2



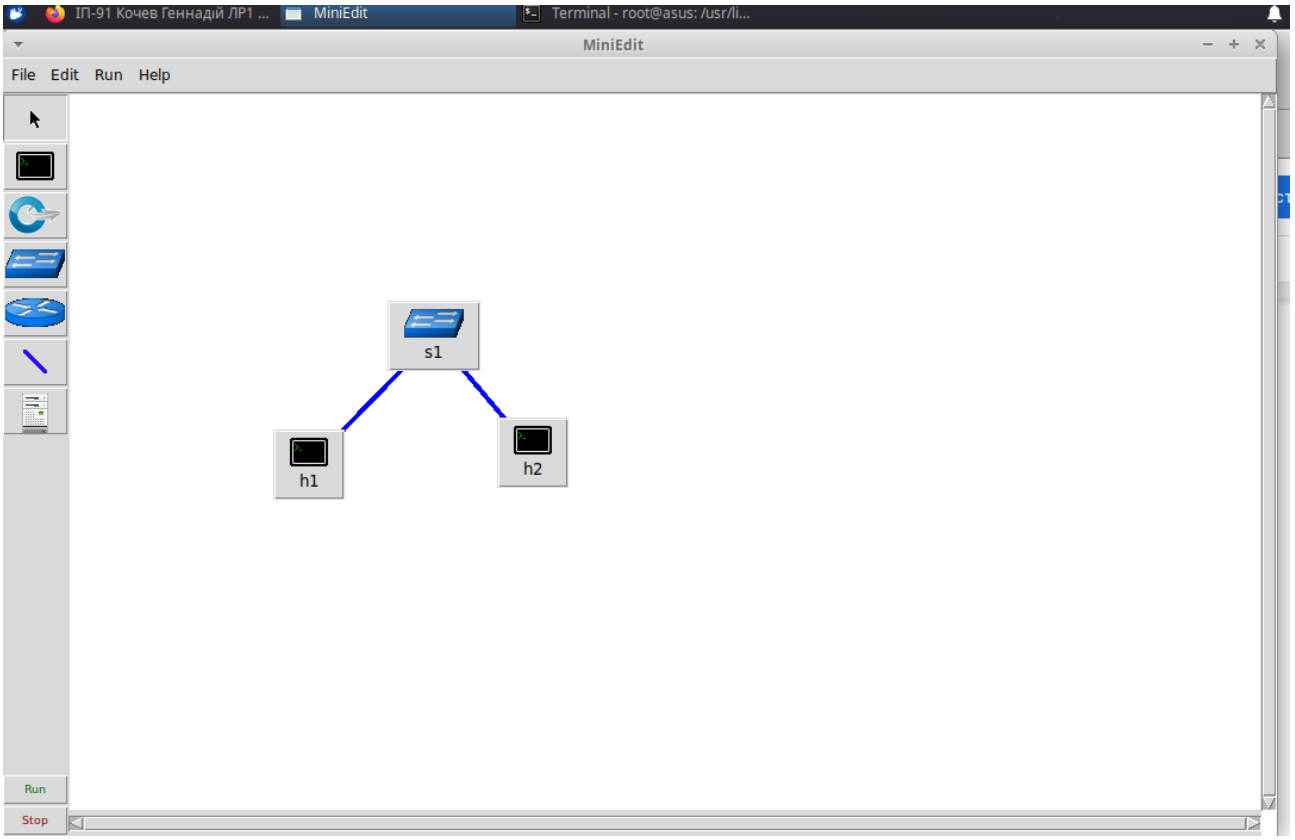
Перевіримо IP адреси що були встановлені



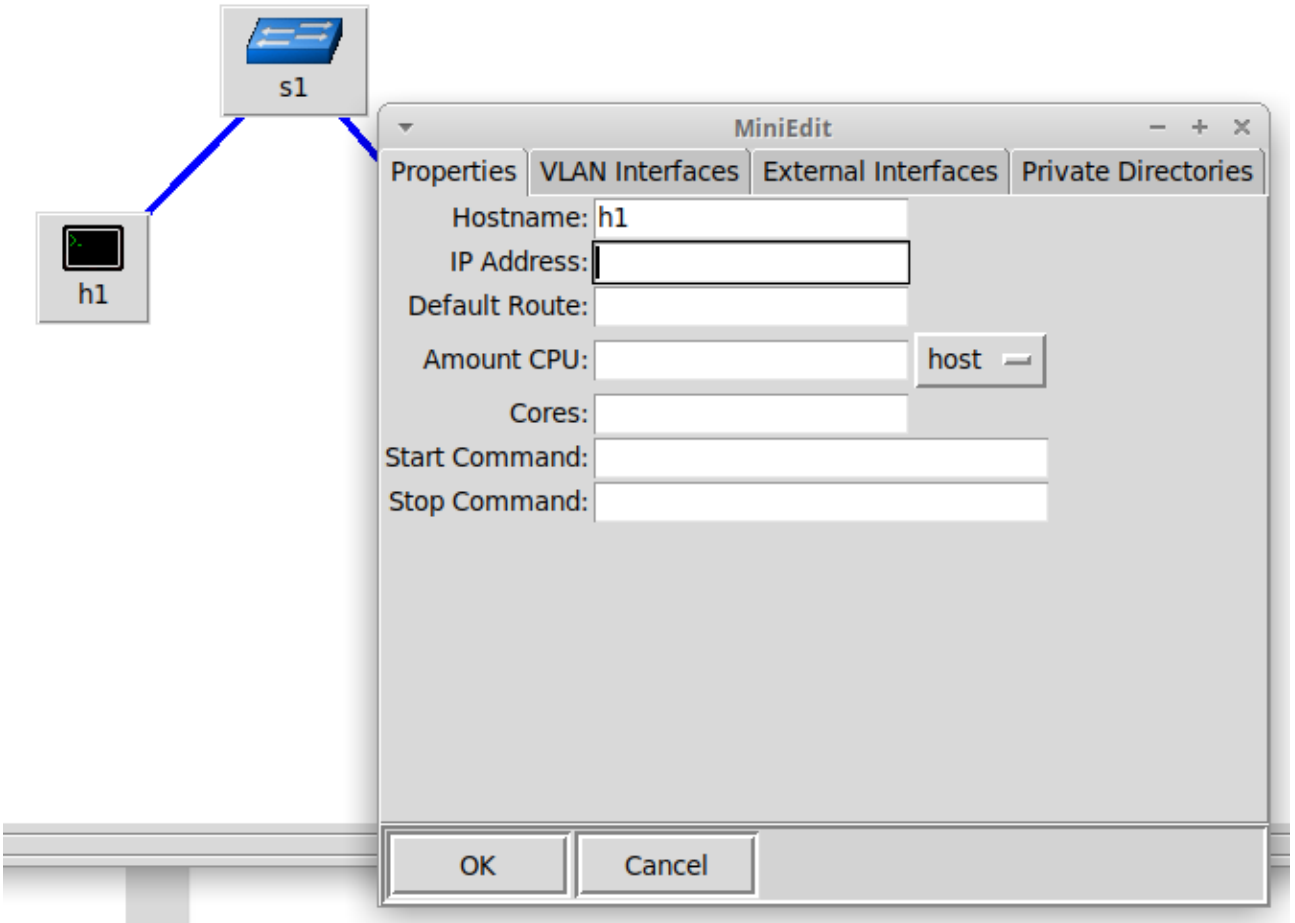
Пінгуємо h2 з хосту h1

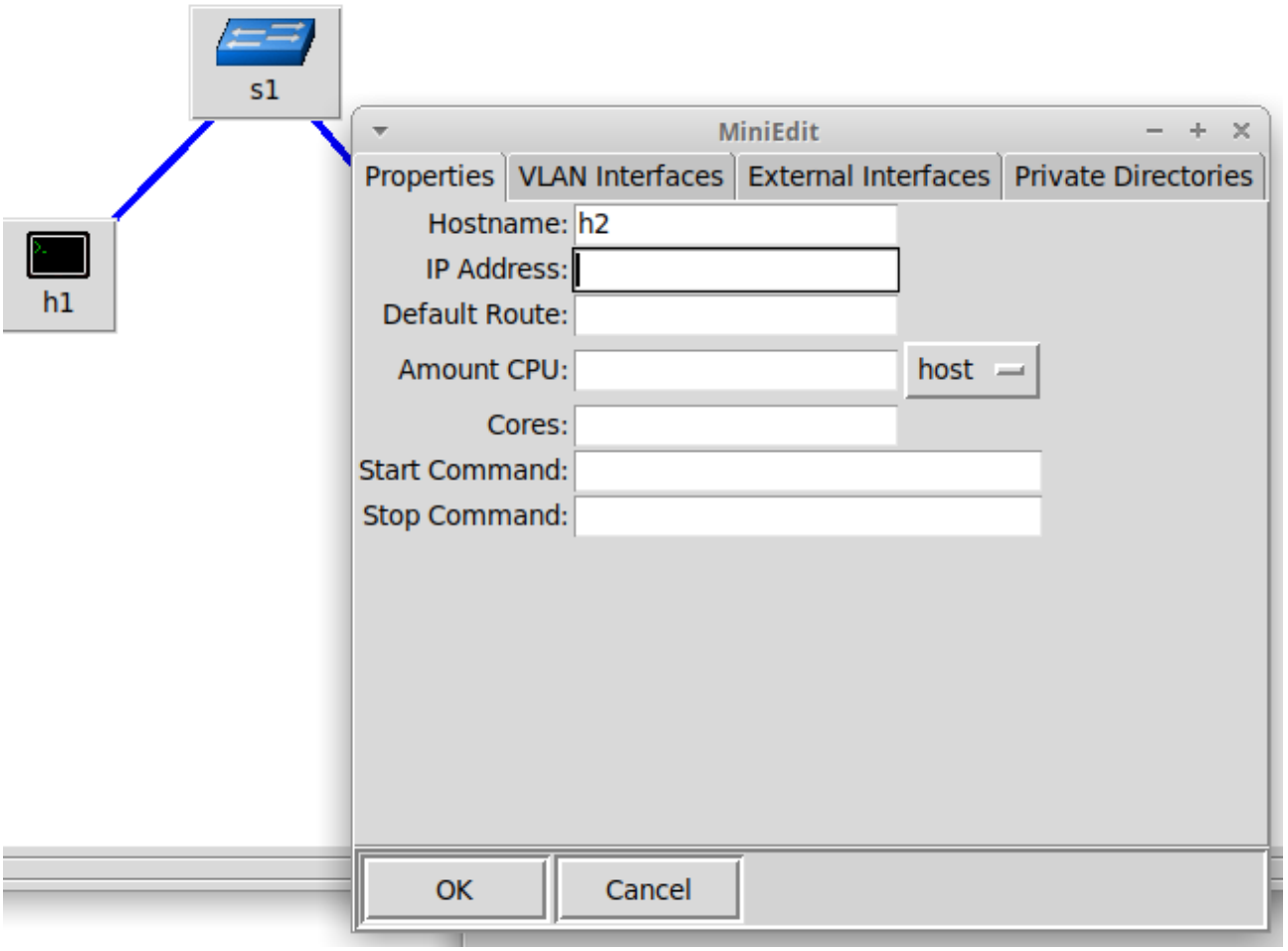


Зупинимо емуляцію

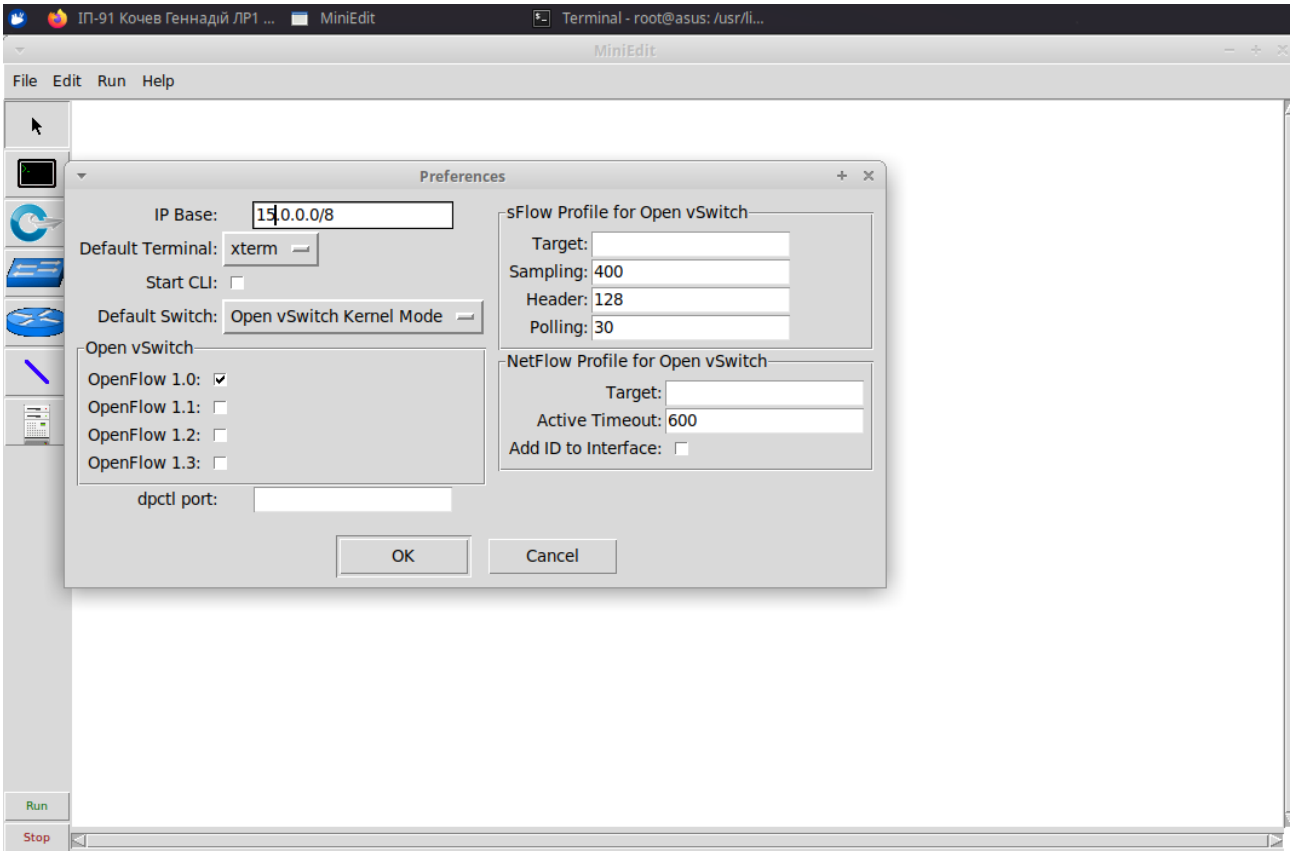


Стираємо старі IP адреси

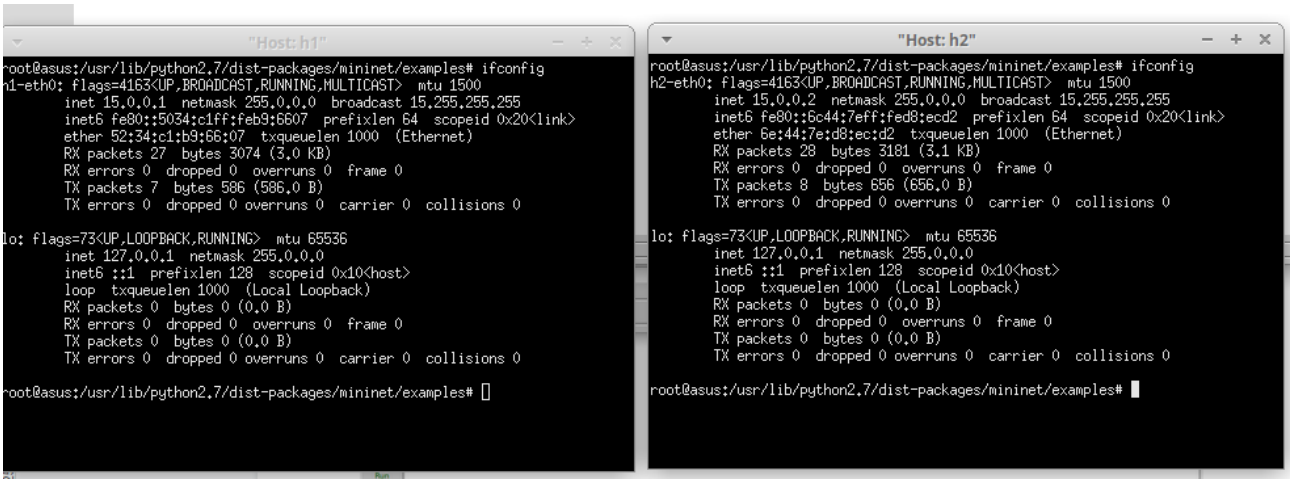




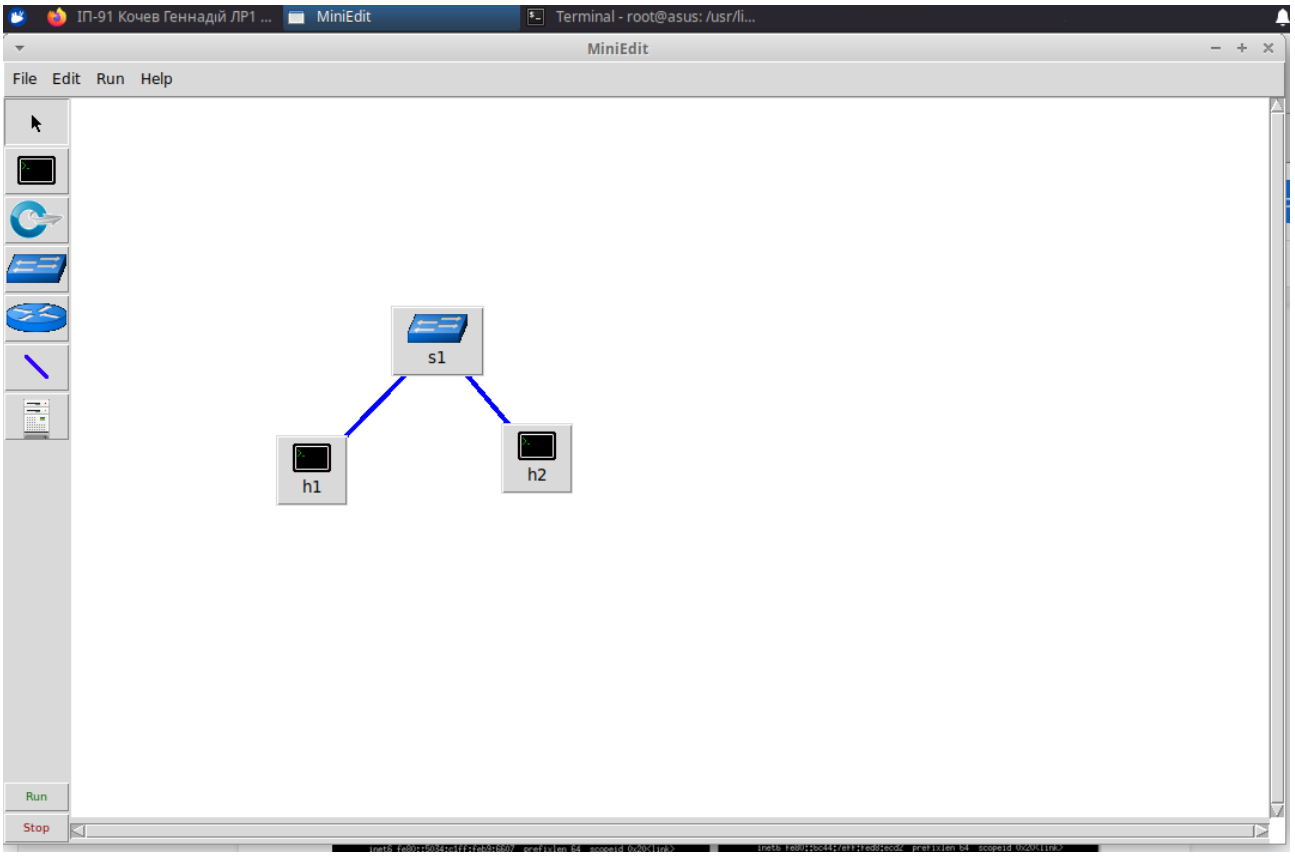
Змінимо IP базу, залишимо IP адресний маппінг за замовченням



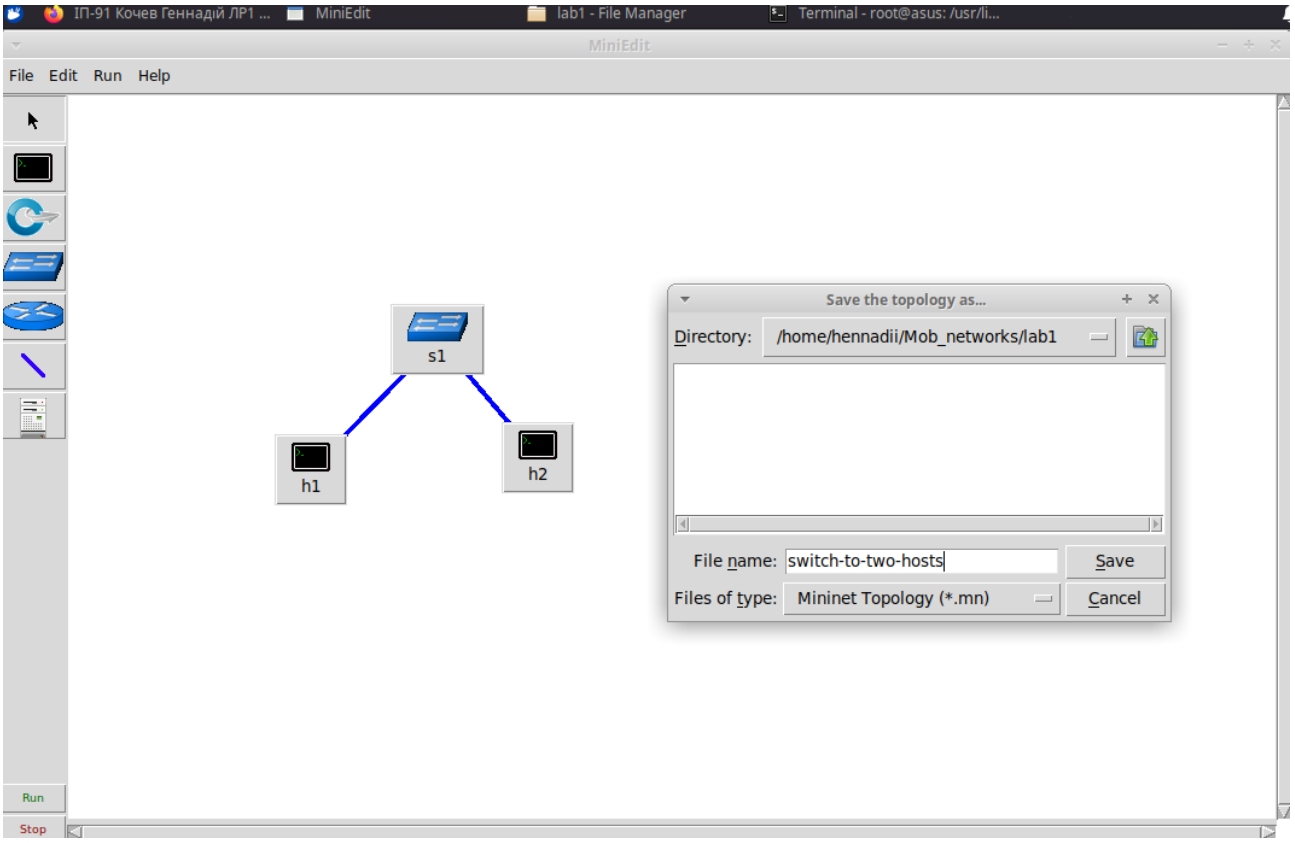
Перевіримо нові адреси за замовчуванням з новою базою



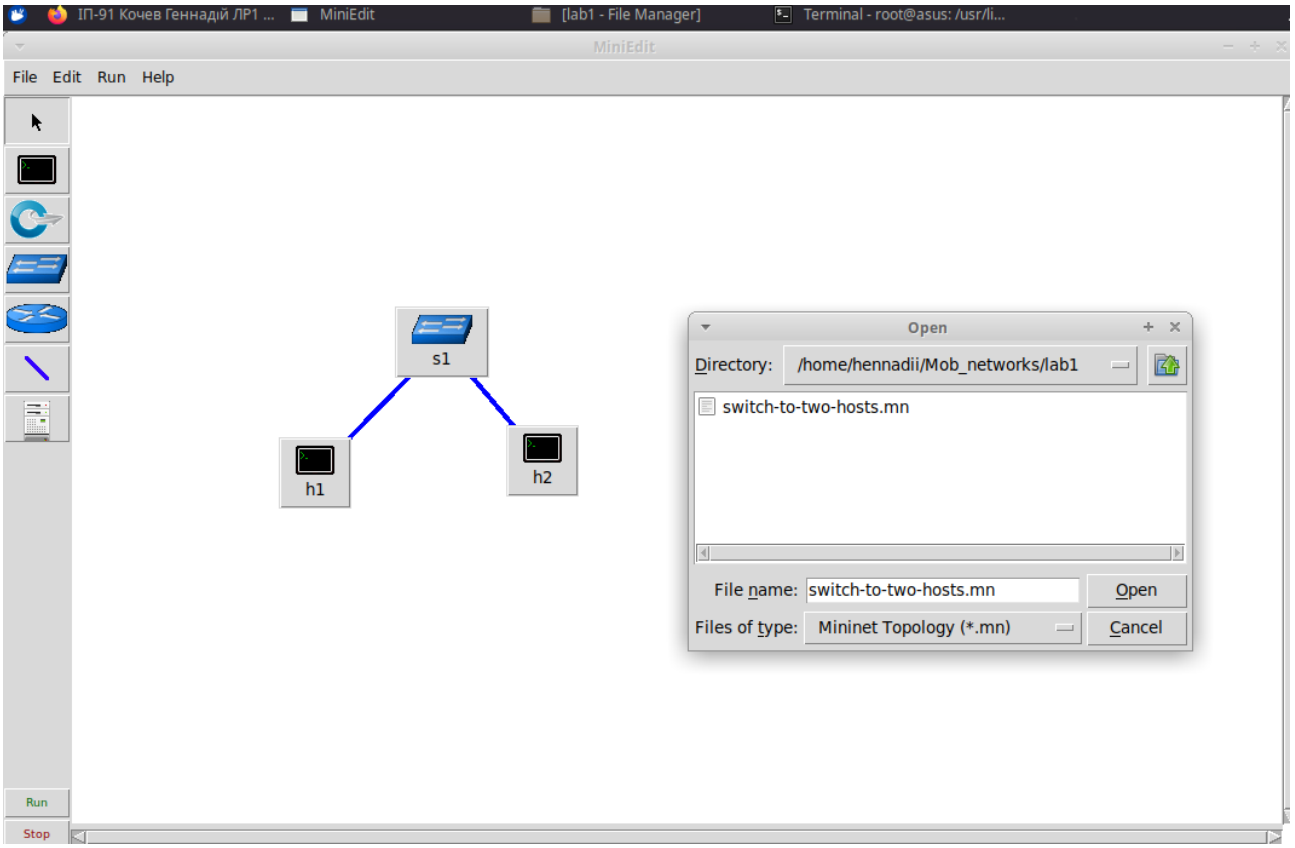
Зупинимо емуляцію



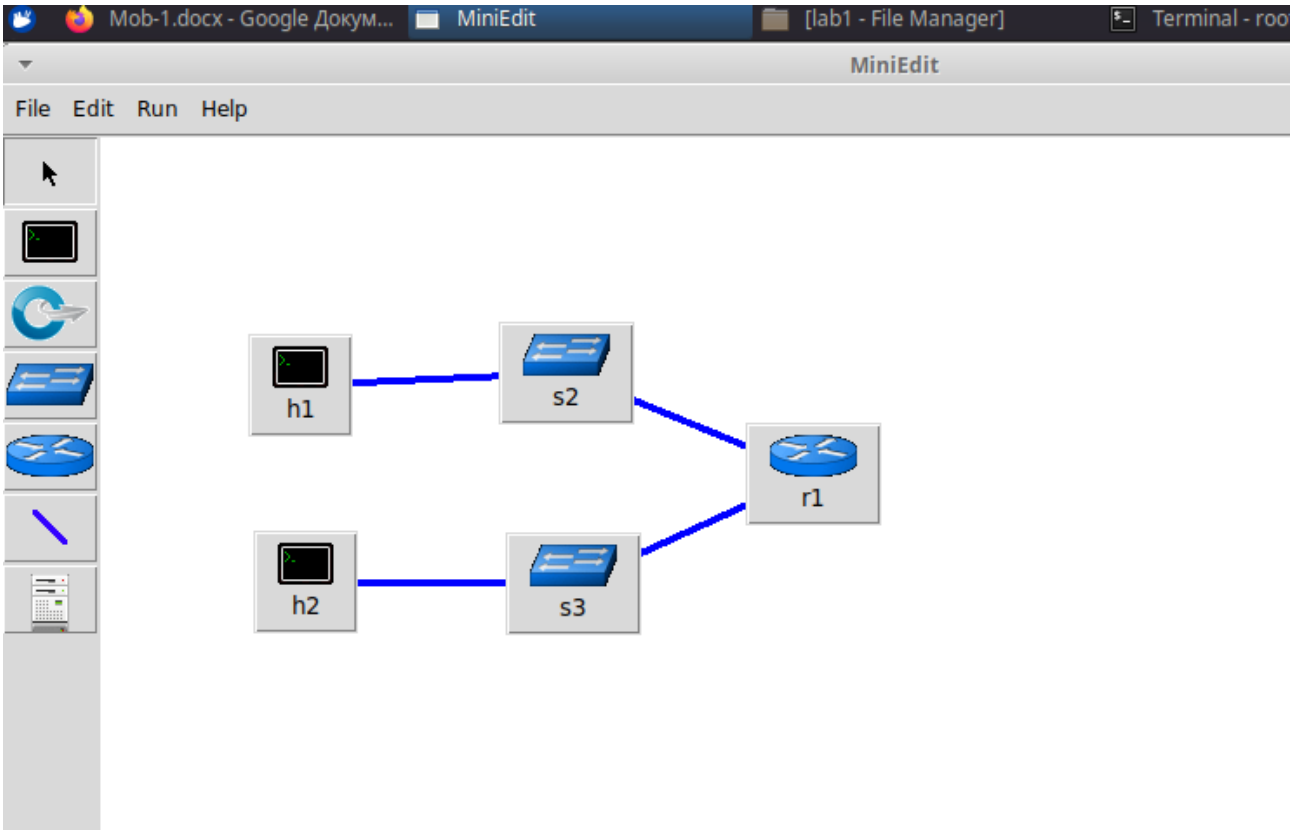
Збережемо топологію



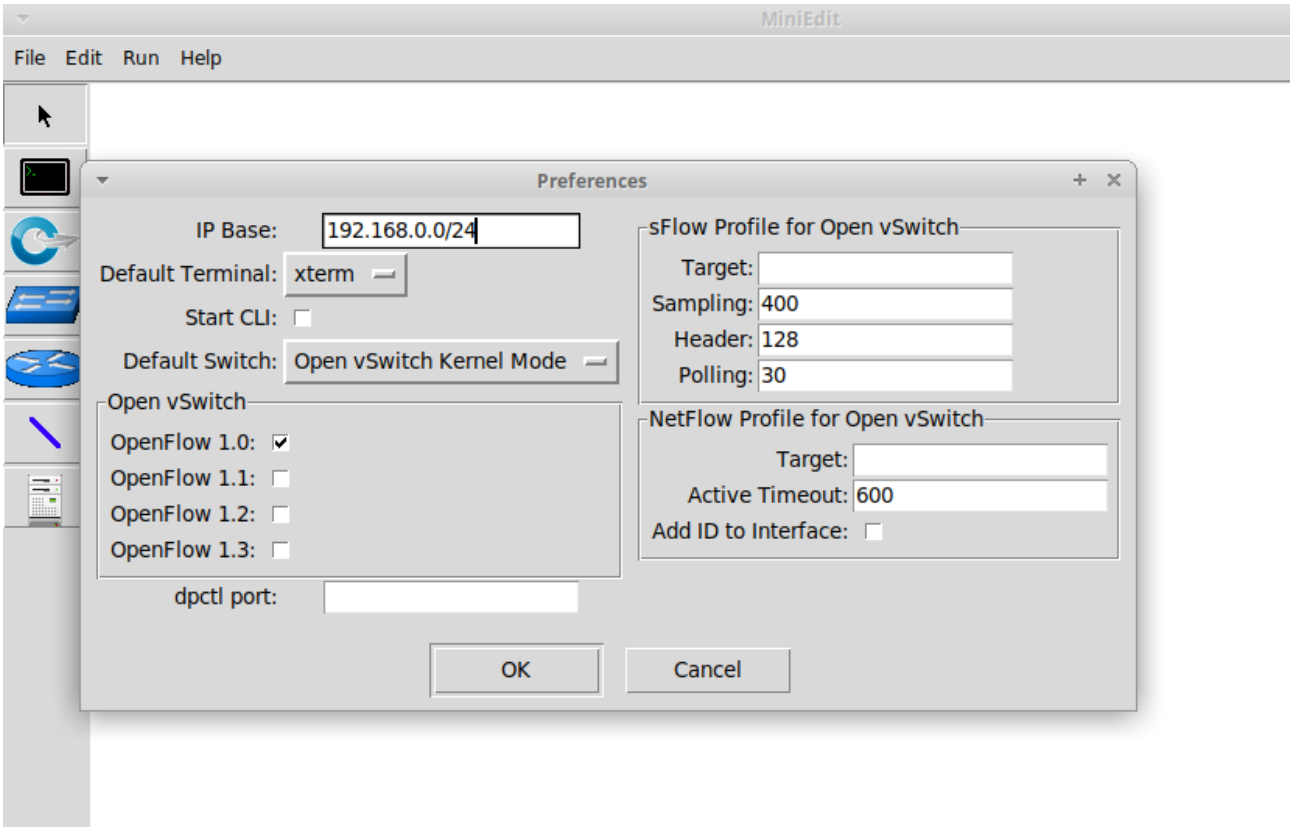
Відкриємо збережену топологію



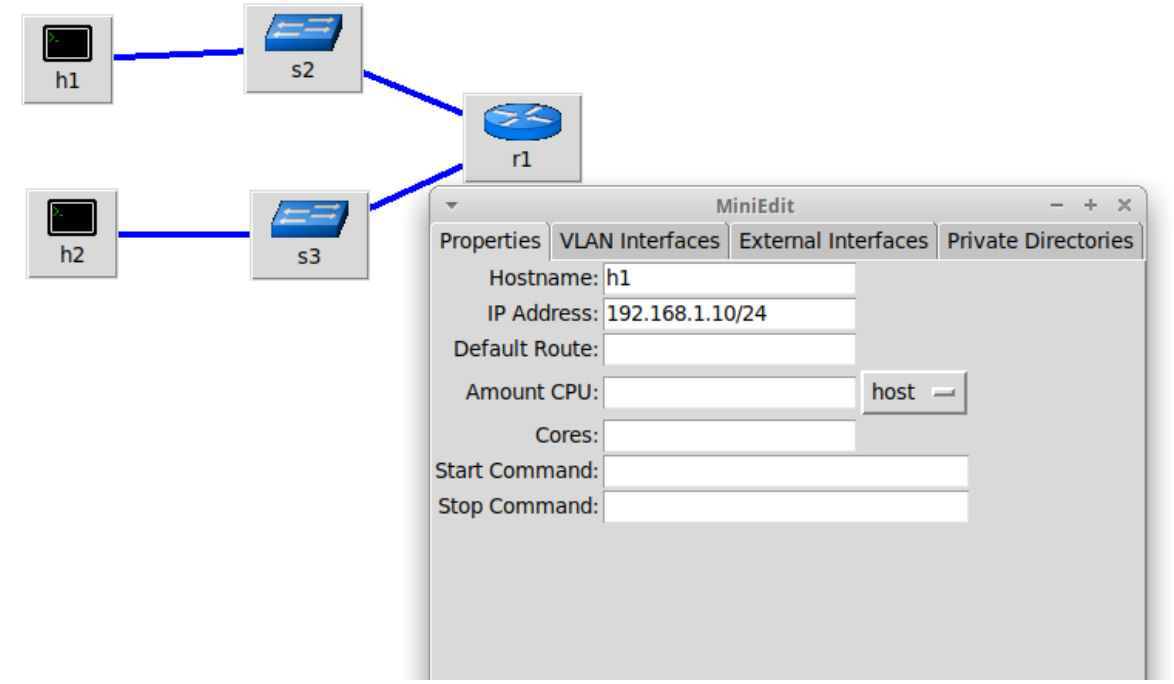
Створимо топологію з роутером r1

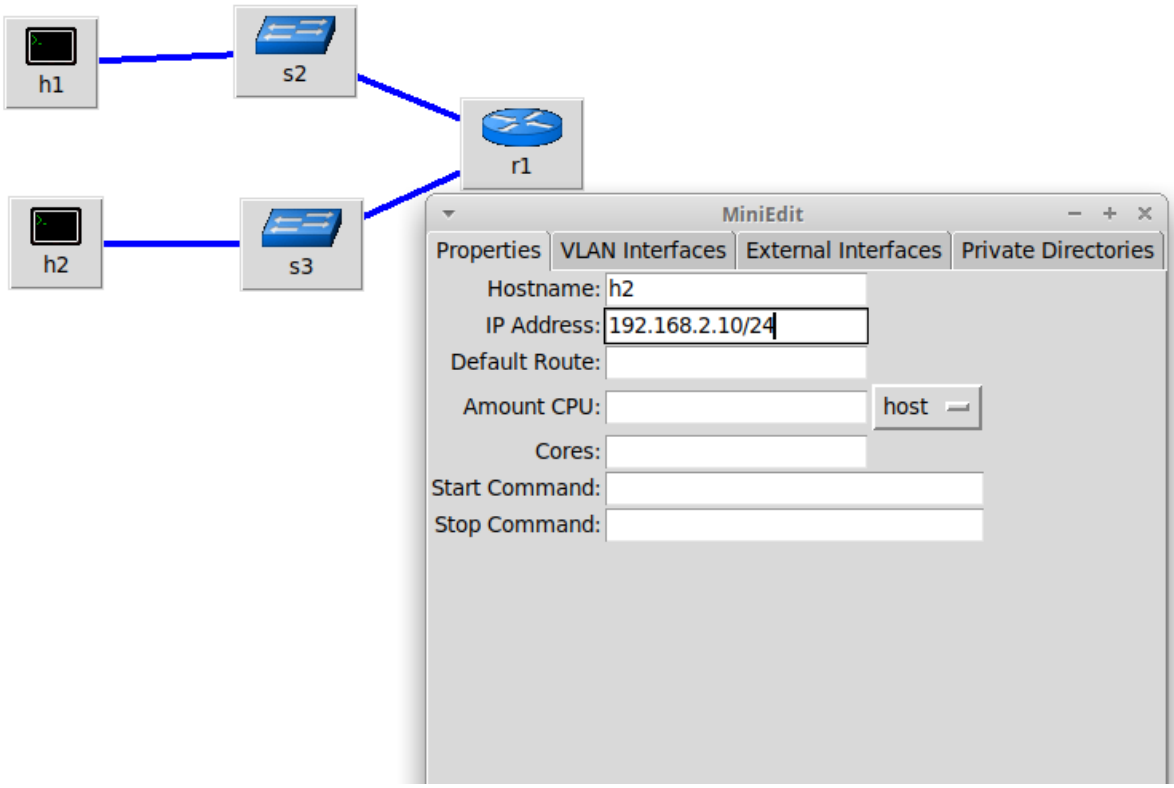


Встановимо базу IP адрес

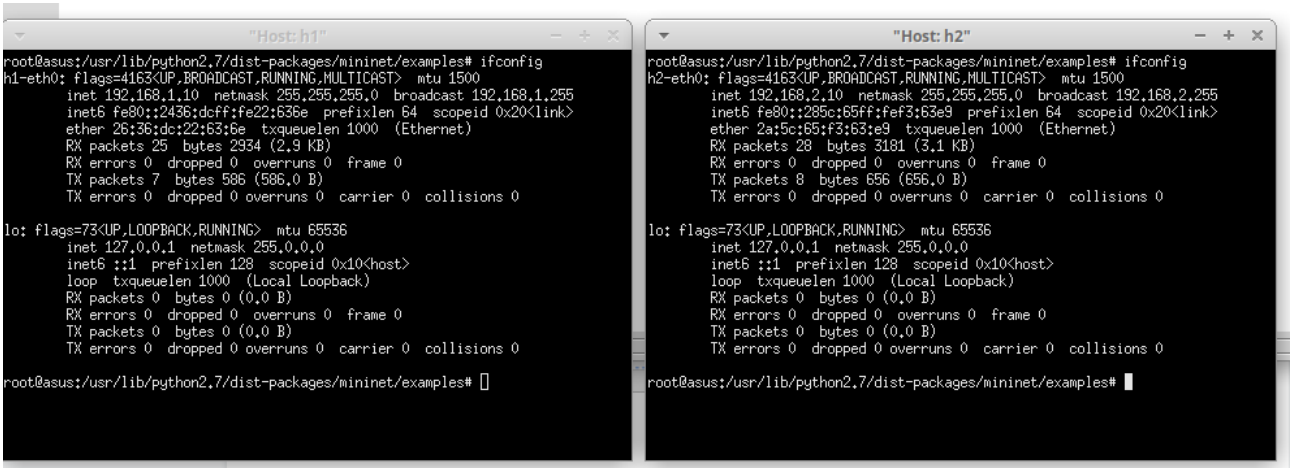


Встановимо IP адреси для хостів

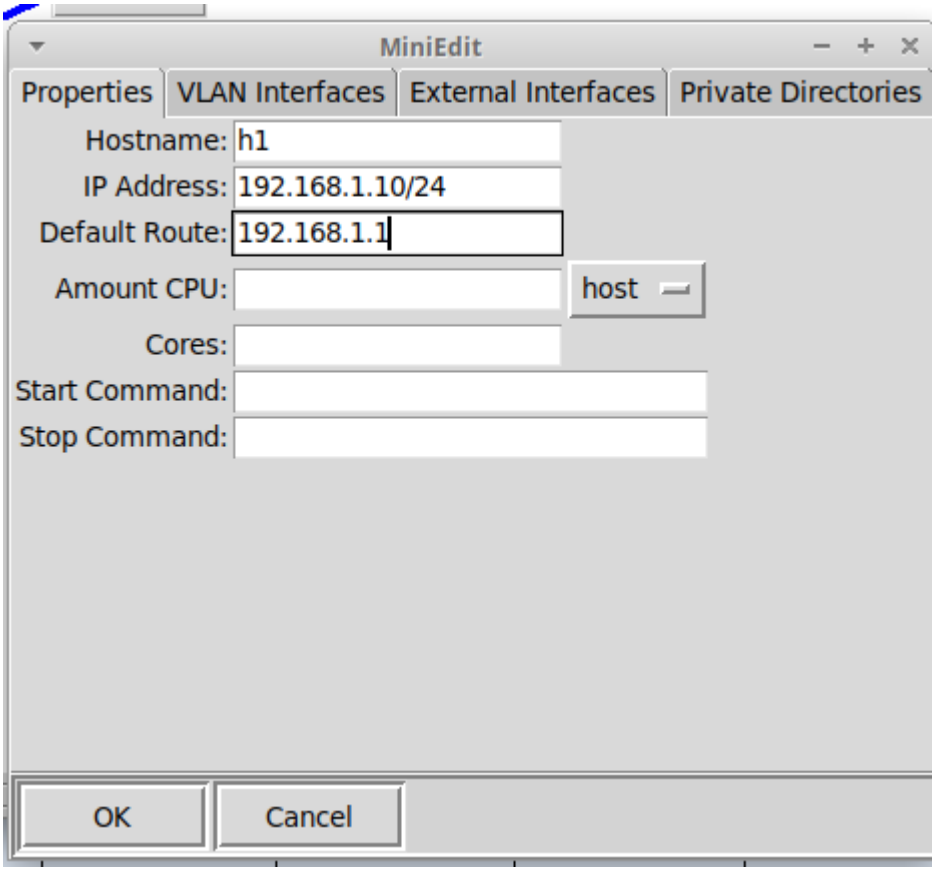


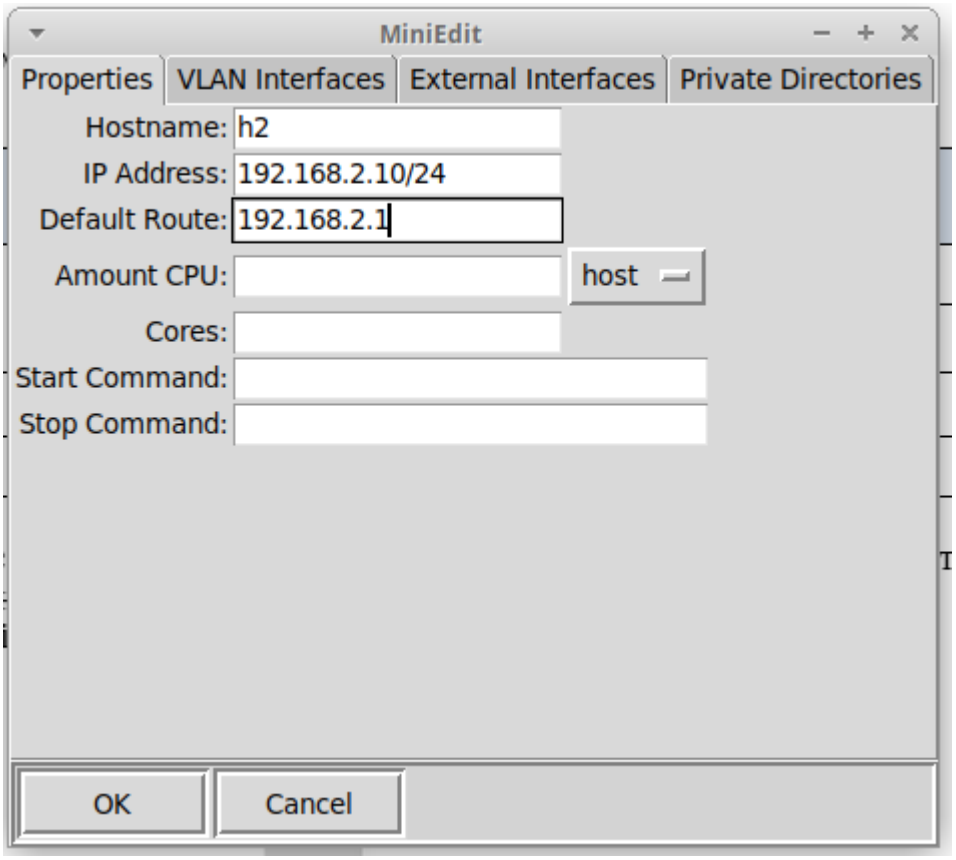


Перевіримо IP адреси хостів з терміналу

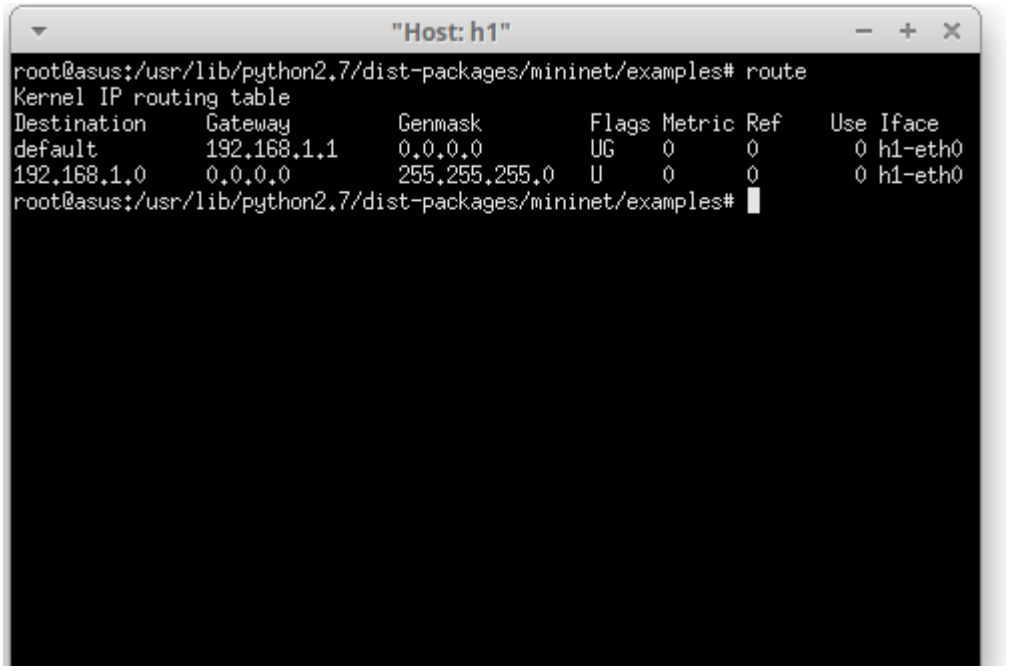


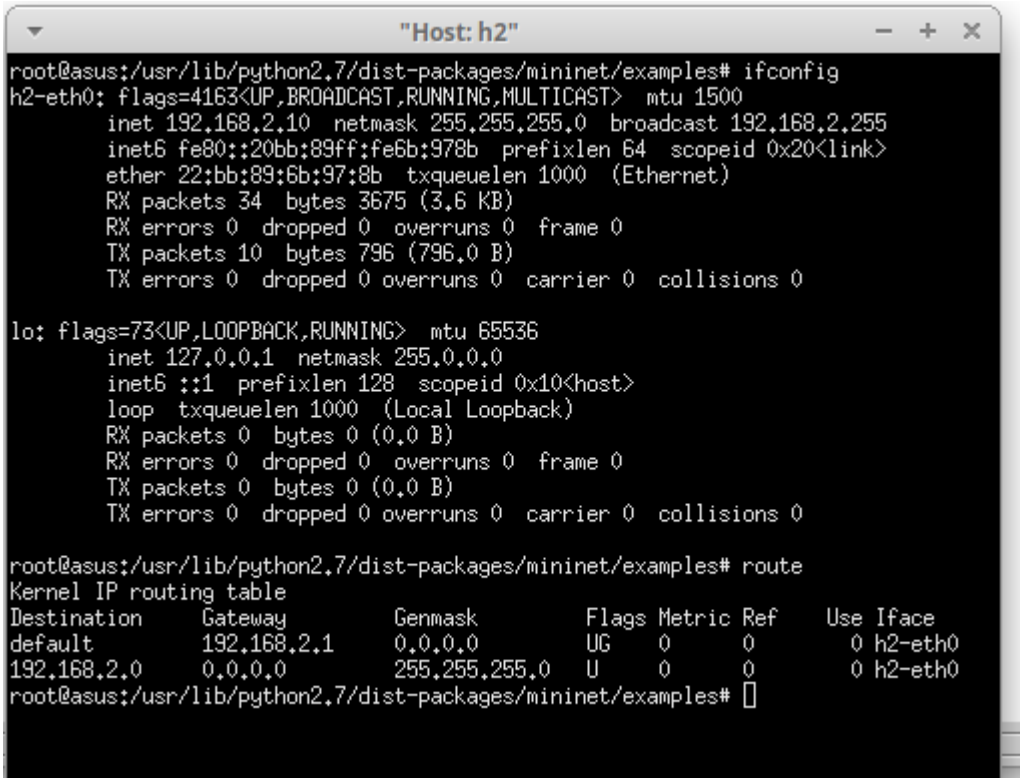
Встановимо дефолтні роути





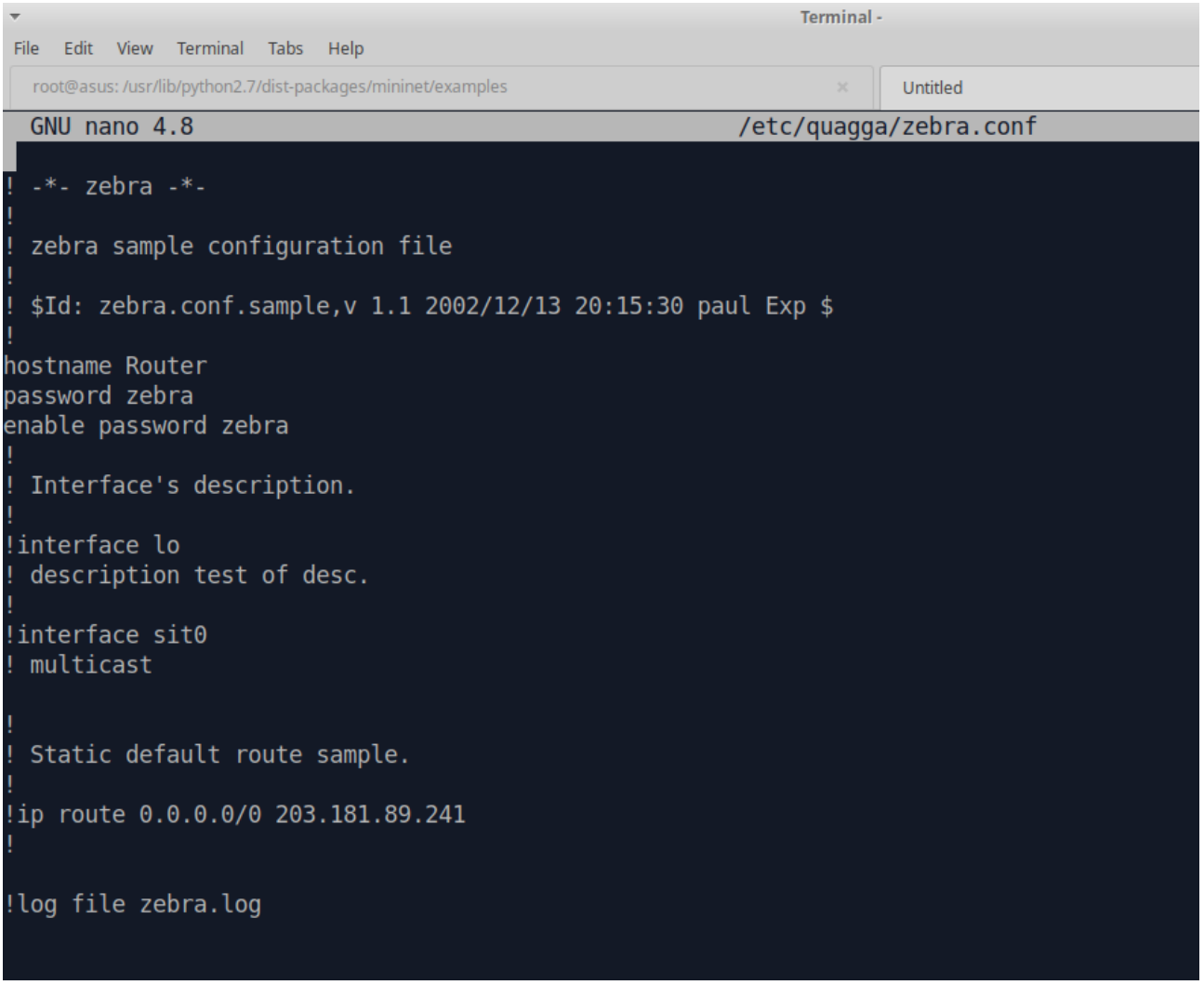
Перевіримо ці дефолтні роути





Налаштуємо роутер через термінал

Налаштуємо дефолтний zebra conf

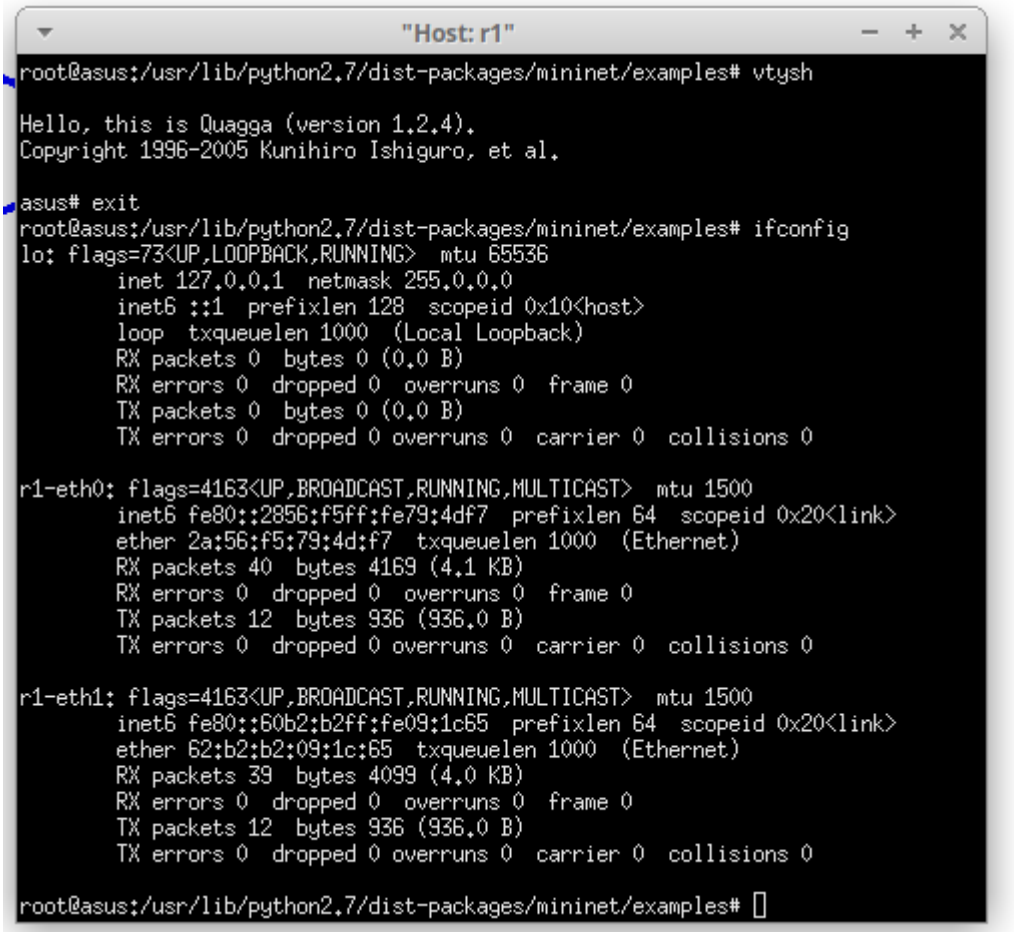


Активуємо демон zebra

```
hennadii@asus:/usr/lib/python2.7/dist-packages/mininet/examples
$ sudo systemctl status zebra.service
● zebra.service - GNU Zebra routing manager
     Loaded: loaded (/lib/systemd/system/zebra.service; enabled; vendor preset: enable
     Active: inactive (dead)
       Docs: man:zebra
hennadii@asus:/usr/lib/python2.7/dist-packages/mininet/examples
$ sudo systemctl enable zebra.service
Synchronizing state of zebra.service with SysV service script with /lib/systemd/system
Executing: /lib/systemd/systemd-sysv-install enable zebra
hennadii@asus:/usr/lib/python2.7/dist-packages/mininet/examples
$ sudo systemctl start zebra.service
hennadii@asus:/usr/lib/python2.7/dist-packages/mininet/examples
$ sudo systemctl status zebra.service
● zebra.service - GNU Zebra routing manager
     Loaded: loaded (/lib/systemd/system/zebra.service; enabled; vendor preset: enable
     Active: active (running) since Mon 2022-10-03 22:04:05 EEST; 2s ago
       Docs: man:zebra
    Process: 32406 ExecStartPre=/sbin/ip route flush proto zebra (code=exited, status=
    Process: 32407 ExecStartPre=/bin/chmod -f 640 /etc/quagga/vtysh.conf /etc/quagga/z
    Process: 32408 ExecStartPre=/bin/chown -f quagga:quagga /etc/quagga/zebra.conf (co
    Process: 32409 ExecStartPre=/bin/chown -f quagga:quaggavty /etc/quagga/vtysh.conf
    Process: 32410 ExecStart=/usr/sbin/zebra -d -A 127.0.0.1 -f /etc/quagga/zebra.conf
   Main PID: 32411 (zebra)
      Tasks: 1 (limit: 4637)
```

Після налаштування, vtysh запускається

Бачимо r1-eth0, r1-eth1 інтерфейси через ifconfig



налаштуємо r1-eth0

```
root@asus:/usr/lib/python2.7/dist-packages/mininet/examples# vtysh

Hello, this is Quagga (version 1.2.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

asus# configure terminal
asus(config)# interface r1-eth0
asus(config-if)# ip address 192.168.1.1/24
asus(config-if)# exit
asus(config)#
```

налаштуємо r1-eth1

```
asus(config-if)# exit
asus(config)# interface r1-eth1
asus(config-if)# ip address 192.168.2.1/24
asus(config-if)# exit
asus(config)# exit
```

Поки що, IP route table не працює, але в чаті написали, що цей крок не є обов’язковим

```
asus(config-if)# exit
asus(config)# exit
asus# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, P - PIM, A - Babel, N - NHRP,
       > - selected route, * - FIB route

K>* 0.0.0.0/0 via 10.0.2.2, enp0s3
C>* 10.0.2.0/24 is directly connected, enp0s3
C>* 127.0.0.0/8 is directly connected, lo
K>* 169.254.0.0/16 is directly connected, enp0s3
asus# exit
root@asus:/usr/lib/python2.7/dist-packages/mininet/examples# ping 192.168.2.10
ping: connect: Network is unreachable
root@asus:/usr/lib/python2.7/dist-packages/mininet/examples# ifconfig
```