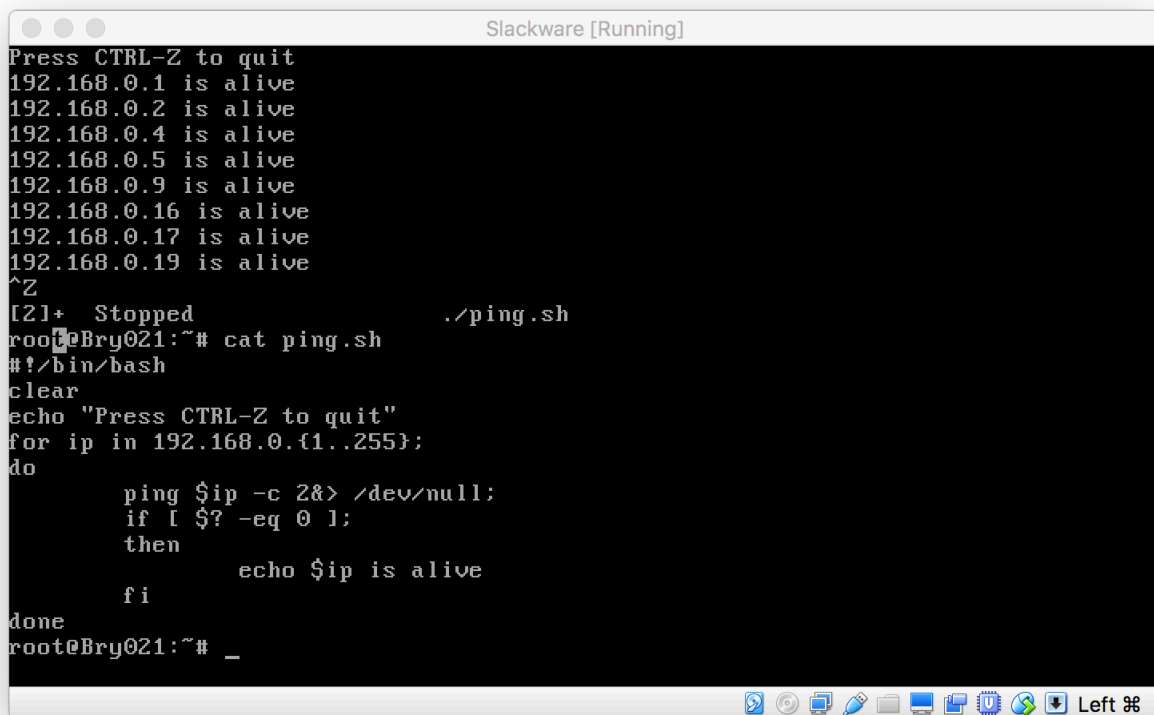


## Labs 9: Network traffic

### Exercise 1. Listing all the machines alive

A screenshot of a Slackware terminal window titled "Slackware [Running]". The terminal shows the output of a script that pings a range of IP addresses. The output lists 8 machines as alive: 192.168.0.1, 192.168.0.2, 192.168.0.4, 192.168.0.5, 192.168.0.9, 192.168.0.16, 192.168.0.17, and 192.168.0.19. The user then presses Ctrl-Z to stop the script, which is shown as "[Z] + Stopped ./ping.sh". The user then runs "cat ping.sh" to display the script's contents. The script is a shell script that iterates through IP addresses from 192.168.0.1 to 192.168.0.255, pinging each one with a count of 2 and redirecting output to /dev/null. It prints "is alive" for successful pings. The terminal prompt is "root@Bry021:~#".

```
Press CTRL-Z to quit
192.168.0.1 is alive
192.168.0.2 is alive
192.168.0.4 is alive
192.168.0.5 is alive
192.168.0.9 is alive
192.168.0.16 is alive
192.168.0.17 is alive
192.168.0.19 is alive
^Z
[Z] + Stopped ./ping.sh
root@Bry021:~# cat ping.sh
#!/bin/bash
clear
echo "Press CTRL-Z to quit"
for ip in 192.168.0.{1..255};
do
    ping $ip -c 2&> /dev/null;
    if [ $? -eq 0 ];
    then
        echo $ip is alive
    fi
done
root@Bry021:~# _
```

The script finds out 8 alive machines on the network. It uses a “for-loop” for iterating through a list of IP addresses generated using the expression 192.168.0.{1..255}

The -c option is used to restrict the number of echo packets to two. In order to redirect both stdout and stderr to the bin we used the option &> /dev/null.

The condition to evaluate in the “if statement” uses the \$? symbol which is the exit status of the previous command, meaning that if the IP address replies its value is zero. Conversely a value different from zero is obtained if the ping command times-out.

### Exercise 2. DNS lookup (optional)

A DNS server receives a request to translate a domain name to an IP address. There are two utilities in Linux which will request for an IP address resolution: host and nslookup. However it is possible to add symbolic names to IP addresses just by adding entries into the file **/etc/hosts** as shown (notice that this is local to our Slackware VM).

```
Slackware [Running]
root@Bry021:~# cat /etc/hosts | tail -n 5

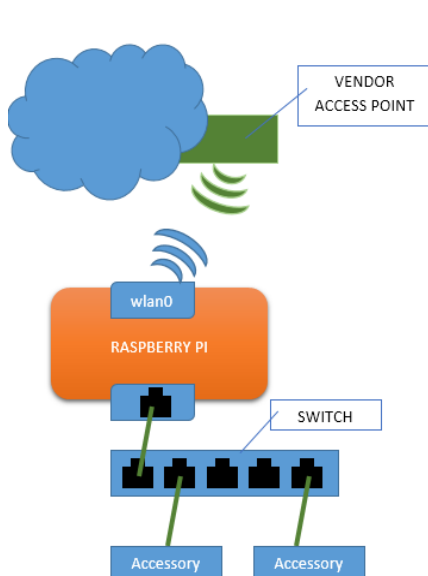
# For loopbacking.
127.0.0.1          localhost
127.0.0.1          Bry021.bryant Bry021
192.168.0.19       raspberry
root@Bry021:~#
root@Bry021:~# ping raspberry -c 2
PING raspberry (192.168.0.19) 56(84) bytes of data.
64 bytes from raspberry (192.168.0.19): icmp_req=1 ttl=64 time=2.83 ms
64 bytes from raspberry (192.168.0.19): icmp_req=2 ttl=64 time=2.79 ms

--- raspberry ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 2.797/2.813/2.830/0.055 ms
root@Bry021:~#
root@Bry021:~# host raspberry
raspberry has address 81.200.64.50
Host raspberry not found: 3(NXDOMAIN)
root@Bry021:~# _
```

In this example, the address of the host called **raspberry** is 192.168.0.19 in my VM, however the DNS server is telling that the hostname has been taken by a different machine (81.200.64.50).

### Exercise 3. Updating the routing table (optional)

The router maintains a table called the routing table, which contains the information on how packets are to be forwarded through machines on the network. Let's see how it looks like the routing table in a real example.



At home I have my raspberry pi sharing its WiFi connection (wlan0) through the port eth0.

When a device on my wireless LAN 192.168.0.0 wants to communicate with a device of the wired network - let's say 192.168.2.0 - it is required to go through the raspberry (as it is common to the two networks is known as gateway).

This gateway is set as follows (note that the IP address has been manually fixed to 192.168.2.1)

```
pi@raspberry:~$ route add 192.168.2.0 gw 192.168.2.1 eth0
```

The idea and the steps to configure the gateway has been taken from:

<https://raspberrypi.stackexchange.com/questions/48307/sharing-the-pis-wifi-connection-through-the-ethernet-port>

```
Slackware [Running]
pi@raspberrypi:~$ ifconfig wlan0 | grep inet
inet 192.168.0.19 netmask 255.255.255.0 broadcast 192.168.0.255
inet6 fe80::20f:60ff:fe06:7f86 prefixlen 64 scopeid 0x20<link>
pi@raspberrypi:~$ ifconfig eth0 | grep inet
inet 192.168.2.1 netmask 255.255.255.0 broadcast 192.168.2.255
inet6 fe80::ba27:ebff:fe4c:58e8 prefixlen 64 scopeid 0x20<link>
pi@raspberrypi:~$
pi@raspberrypi:~$ route -n
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0          192.168.0.1    0.0.0.0         UG      0      0      0 wlan0
192.168.0.0      0.0.0.0        255.255.255.0   U       0      0      0 wlan0
192.168.2.0      0.0.0.0        255.255.255.0   U       0      0      0 eth0
pi@raspberrypi:~$
pi@raspberrypi:~$ route add 192.168.2.0 gw 192.168.2.1 eth0
SIOCADDRT: Operation not permitted
pi@raspberrypi:~$ sudo route add 192.168.2.0 gw 192.168.2.1 eth0
pi@raspberrypi:~$ route -n
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0          192.168.0.1    0.0.0.0         UG      0      0      0 wlan0
192.168.0.0      0.0.0.0        255.255.255.0   U       0      0      0 wlan0
192.168.2.0      192.168.2.1    255.255.255.255 UGH     0      0      0 eth0
192.168.2.0      0.0.0.0        255.255.255.0   U       0      0      0 eth0
pi@raspberrypi:~$
```

Let's setup now the default gateway of a second raspberry connected to the first one through an switch. Its IP address has been manually reserved to 192.168.2.2 and its default gateway is the first raspberry:

```
pi@presspi:~$ route add default gw 192.168.2.1 eth0
```

```
Slackware [Running]
pi@presspi:~$ route -n
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.2.0      0.0.0.0        255.255.255.0   U       0      0      0 eth0
pi@presspi:~$ sudo route add default gw 192.168.2.1 eth0
pi@presspi:~$ r
-bash: r: command not found
pi@presspi:~$ route -n
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0          192.168.2.1    0.0.0.0         UG      0      0      0 eth0
192.168.2.0      0.0.0.0        255.255.255.0   U       0      0      0 eth0
pi@presspi:~$
```