

LPIC-1 102-500 – Lesson 15

109.3 Basic network troubleshooting



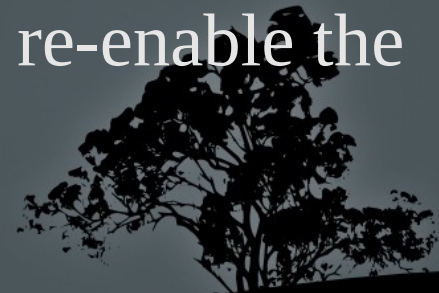
Problems with DHCP

- If a computer refuses to get an IP address from DHCP, or gets a 169.254.0.0/16 address this could mean that there is not working DHCP on the network.
 - Try to manually set an available IP address from your network and give the correct netmask and gateway (**ip addr/ifconfig**).
 - Then try to communicate with another host in the same network (**ping**) like the default gateway.



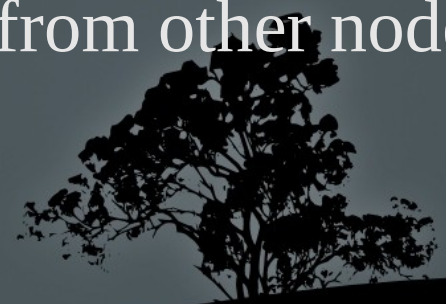
• IP Address Conflict

- IP address conflict happens when we use the same IP in two or more nodes. This is one of the hardest problems to detect. When 2 or more nodes have the same IP address you will observe interruptions in their operation.
 - Use **ping** from another computer and see if you get a reply. If you disconnect (**ifdown**) the suspect computer and you still get replies, this means that the IP address is in use somewhere else.
 - Try to assign an available IP address from the correct network (**ip addr/ifconfig**) using the correct netmask and gateway, and try to re-enable the interface (**ifup**).



IP from another subnet

- Sometimes we may be confused and set an IP address that it is apparently from our own network but it is, in fact, on another network. For example 192.168.10.250 does not belong to 192.168.10.0/25.
 - Try to calculate the boundaries of your network. The **ipcalc** command can be a valuable tool in such cases.
 - Then try to set an IP address from your network and using the correct mask.
 - Finally use **ping** to get an answer from other nodes in your network.



Unreachable remote networks

- If you can communicate with nodes inside your own subnet but you cannot communicate with other subnets, this is a typical problem, related to the default gateway .
 - The problem could be a gateway which is offline. Try to get an answer from it with **ping**.
 - The wrong gateway may have been specified. Try to set the correct one with **ip ro add default via** after consulting your network administrator.

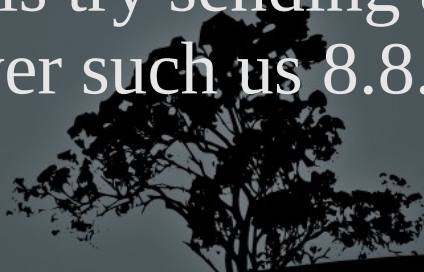


Wrong hostname

- Some networks can pick up the hostname from the computer and set it up in their DNS. In such scenarios, if you have the wrong hostname, the other computers will not be able to connect to you using that hostname.
 - Correct your hostname with the command **hostname** (or **hostnamectl**)
 - Try to **ping** it from another node in the network.
 - If it works make the hostname persistent by adding it in **/etc/hostname** or setting it with **hostnamectl**.



Problems with DNS

- If a system fails to resolve named to IP addresses, the reason can be the lost communication with the DNS server. If we can talk to an IP address but not the hostname this is a typical DNS problem.
 - Try to ping the IP address of the host and then the hostname (**ping**).
 - If the IP address responds but not the hostname, check the communication with the DNS server with **ping** or try to send a DNS query to it with **host**, **dig** or **nslookup**. If that fails try sending a DNS query to a public DNS server such as 8.8.8.8 (Google DNS).
- 

Check communication with remote networks

- If we fail to communicate with a remote node or network we should try to find the intermediate device that causes this problem.
 - The **ping** command can verify the problem but it does not help us find where exactly is the problem.
 - To troubleshoot the problem we have to use **tracert** or a similar command like **tracert** or **mtr**.



The `ip` command

- The **ip** command is a better replacement for the legacy **ifconfig** and **route** commands.
- `# ip address #` show interfaces with ip information.

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
```

```
2: enp0s25: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
fq_codel state UP group default qlen 1000
    link/ether 28:d2:44:33:84:9c brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.225/24 brd 192.168.10.255 scope global dynamic
noprofixroute enp0s25
        valid_lft 38044sec preferred_lft 38044sec
    inet6 fd64:d180:8b30:0:2ad2:44ff:fe33:849c/64 scope global
dynamic mngtmpaddr
        valid_lft 6993sec preferred_lft 1593sec
    inet6 fe80::2ad2:44ff:fe33:849c/64 scope link
        valid_lft forever preferred_lft forever
```

Check network information with `ip`

- # `ip address # = ip a` show IP information on all interfaces.
- # `ip addr show dev enp0s2 #` show IP information on `enp0s2`.
- # `ip route # = ip ro` show IPv4 routing table.
- # `ip -6 ro #` show IPv6 routing table.
- # `ip link #` show link status.



Configure transient network settings with `ip`

- `# ip addr add 192.168.1.7/24 dev enp0s2 # set non-persistent IP address on enp0s2.`
- `# ip addr del 192.168.1.7/24 dev enp0s2 # delete IP address from enp0s2 (non-persistent).`
- `# ip ro add default via 192.168.4.1 dev eno1 # set default gateway.`
- `# ip ro add 192.168.5.0/24 via 192.168.5.1 dev enp0s3 # Add static route.`
- `# ip link set wlp3s0 down # disable interface.`
- `# ip link set wlp3s0 up # enable interface.`

The `hostname` command

- The **hostname** command can be used for setting and displaying the hostname and domainname. Changing the hostname with **hostname** is not persistent!
- `$ hostname # show hostname`
`name-lpi`
- `$ hostname -f # show host and domain name`
`name-lpi.example.com`
- `# hostname other-name # change hostname.`
This change is temporary and the name will roll back on the next reboot unless we set the hostname in `/etc/hostname` and `/etc/hosts`.



The `ss` command

- The `ss` (socket statistic) is a utility for displaying active connections, active ports and detailed statistics about network usage.

Options:

- `-i` # internal TCP informations.
- `-s` # detailed per protocol statistics.
- `-a` # show all listening ports and active connections.
- `-l` # show listening ports.
- `-p` # show the process behind each connection or listening port.
- `-r` # resolve hostnames.
- `-n` # numeric results. It does not resolve hostnames and ports which means faster results.
- `-t` # show TCP connections.
- `-u` # show UDP communication.



The `ss` command

- # **ss** # show all sockets, TCP, UDP, ICMP and unix.
- # **ss -tun** # show TCP and UDP traffic in numeric form.
- # **ss -tur** # display of TCP and UDP traffic and resolve hostnames.
- # **ss -a** # show connections and ports.
- # **ss -lnptu** # show listening TCP and UDP ports, in numeric form, along with programs that occupy these ports.
- # **ss -i** # show internal TCP information.
- # **ss -s** # show detailed, per protocol statistics.



Check network availability with `ping`

- The **ping** command check the availability of network nodes using the **ICMP** protocol.

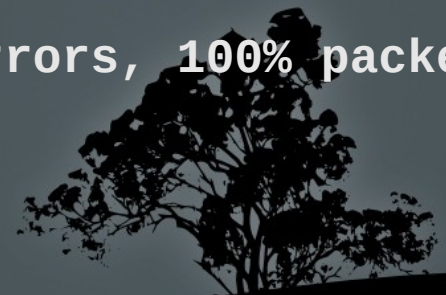
- ```
$ ping 127.0.0.1 # check the local TCP/IP stack
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_req=1 ttl=64 time=0.031 ms
64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.051 ms
^C # <= *** Ctrl-C to terminate ***
--- 127.0.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.031/0.041/0.051/0.010 ms
```

- ```
$ ping www.google.com # check the response of www.google.com
PING www.l.google.com (173.194.69.147) 56(84) bytes of data.
64 bytes from bk-in-f147.1e100.net (173.194.69.147): icmp_req=1
ttl=47 time=100 ms
64 bytes from bk-in-f147.1e100.net (173.194.69.147): icmp_req=2
ttl=47 time=100 ms
64 bytes from bk-in-f147.1e100.net (173.194.69.147): icmp_req=3
ttl=47 time=102 ms
^C                               # <= *** Ctrl-C to terminate ***
--- www.l.google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 100.609/101.233/102.330/0.820 ms
```

Check network availability with `ping`

- ```
$ ping -c4 2.1.1.1 # send only 4 ICMP packets to IP 2.1.1.1.
No response.
PING 2.1.1.1 (2.1.1.1) 56(84) bytes of data.

--- 2.1.1.1 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time
2999ms
```
- ```
$ ping -c4 192.168.2.8 # send only 4 ICMP packets to  
# 192.168.2.8.  
PING 192.168.2.8 (192.168.2.8) 56(84) bytes of data.  
From 192.168.2.11 icmp_seq=1 Destination Host Unreachable  
From 192.168.2.11 icmp_seq=2 Destination Host Unreachable  
From 192.168.2.11 icmp_seq=3 Destination Host Unreachable  
From 192.168.2.11 icmp_seq=4 Destination Host Unreachable  
# reply from 192.168.2.11 that 192.168.2.8 is offline  
--- 192.168.2.8 ping statistics ---  
4 packets transmitted, 0 received, +4 errors, 100% packet  
loss, time 3013ms
```



Check network availability of IPv6 with `ping6`

- ```
$ ping6 -c4 2001:4860:4860::8844
PING 2001:4860:4860::8844(2001:4860:4860::8844) 56 data
bytes
64 bytes from 2001:4860:4860::8844: icmp_seq=1 ttl=119
time=92.3 ms
64 bytes from 2001:4860:4860::8844: icmp_seq=2 ttl=119
time=93.1 ms
64 bytes from 2001:4860:4860::8844: icmp_seq=3 ttl=119
time=92.7 ms
64 bytes from 2001:4860:4860::8844: icmp_seq=4 ttl=119
time=92.9 ms

--- 2001:4860:4860::8844 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 7ms
rtt min/avg/max/mdev = 92.270/92.719/93.058/0.363 ms
```

**Note:** on recent versions **ping** also works with IPv6.



# Check network paths with `traceroute` and `tracpath`

- These commands are used to check the path of a route until a certain destination. The results will display the intermediate nodes and if there is a problem, we will know exactly where the problem is. **traceroute** has more options than **tracpath** but the latter is default for most distributions. **traceroute** and **tracpath** work for IPv6 on recent systems but for older systems you may want to try **traceroute6** and **tracpath6**.
- There is also **mtr** which combines the results of **traceroute/tracpath** and **ping** and continues producing results until we press Ctrl-C.



# Check network paths with `traceroute` and `tracpath`

```
$ traceroute malena.theo-andreou.org # find path to malena.theo-andreou.org.
```

```
traceroute to malena.theo-andreou.org (37.247.48.150), 30 hops max, 60 byte packets
 1 gateway (192.168.10.1) 0.231 ms * *
 2 gw.primeoffice.thunderworx.net (78.158.142.254) 1.094 ms 1.078 ms 1.424 ms
 3 gw.ip.primehome.com (46.21.57.254) 49.094 ms 49.110 ms 49.089 ms
 4 j1.lim-2.nsp-transit.net (78.158.134.118) 49.069 ms 49.066 ms j1.lim.nsp-transit.net
 (78.158.134.250) 49.031 ms
 5 v3068.j1.fra.prime-tel.net (78.158.141.157) 97.865 ms 100.546 ms 100.570 ms
 6 213.140.39.140 (213.140.39.140) 102.387 ms 100.581 ms 102.761 ms
 7 5.53.5.253 (5.53.5.253) 115.164 ms 115.172 ms 111.848 ms
 8 5.53.4.28 (5.53.4.28) 102.615 ms 112.005 ms 113.865 ms
 9 be12956.agr41.fra03.atlas.cogentco.com (130.117.14.117) 129.002 ms 128.962 ms
 128.894 ms
10 be3187.ccr42.fra03.atlas.cogentco.com (130.117.1.118) 131.774 ms 107.528 ms 105.803
 ms
11 be2960.ccr22.muc03.atlas.cogentco.com (154.54.36.254) 112.813 ms
 be2959.ccr21.muc03.atlas.cogentco.com (154.54.36.54) 113.138 ms
 be2960.ccr22.muc03.atlas.cogentco.com (154.54.36.254) 112.830 ms
12 be3072.ccr51.zrh02.atlas.cogentco.com (130.117.0.17) 125.978 ms 125.930 ms 125.661
 ms
13 be3586.rcr21.mil01.atlas.cogentco.com (154.54.60.114) 126.047 ms 125.913 ms
 be2043.rcr21.mil01.atlas.cogentco.com (154.54.38.102) 125.998 ms
14 be3459.nr51.b019138-1.mil01.atlas.cogentco.com (154.25.12.74) 126.769 ms 128.092 ms
 127.883 ms
15 prometheus.demarc.cogentco.com (149.14.134.122) 131.175 ms 131.180 ms 133.485 ms
16 37.247.50.20 (37.247.50.20) 129.294 ms 37.247.50.54 (37.247.50.54) 130.344 ms
 130.225 ms
17 malena.theo-andreou.org (37.247.48.150) 130.919 ms 117.366 ms 37.247.50.51
 (37.247.50.51) 130.660 ms
```

# Check network paths with `traceroute` and `tracert`

```
$ tracert malena.theo-andreou.org
1?: [LOCALHOST] pmtu 1500
1: gateway 0.640ms
1: gateway 0.635ms
2: gw.primeoffice.thunderworx.net 1.337ms
3: gw.ip.primehome.com 48.664ms asymm 4
4: j1.lim.nsp-transit.net 49.124ms asymm 5
5: v3068.j1.fra.prime-tel.net 115.959ms asymm 6
6: 213.140.39.140 129.760ms
7: 5.53.5.253 103.743ms
8: 5.53.4.28 109.318ms
9: be12956.agr41.fra03.atlas.cogentco.com 141.998ms asymm 11
10: be3186.ccr41.fra03.atlas.cogentco.com 120.809ms asymm 12
11: be2960.ccr22.muc03.atlas.cogentco.com 127.217ms asymm 13
12: be3073.ccr52.zrh02.atlas.cogentco.com 131.622ms asymm 14
13: be3586.rcr21.mil01.atlas.cogentco.com 148.614ms asymm 15
14: be3459.nr51.b019138-1.mil01.atlas.cogentco.com 141.369ms asymm 16
15: prometheus.demarc.cogentco.com 139.371ms asymm 17
16: 37.247.50.20 133.596ms asymm 12
17: malena.theo-andreou.org 133.715ms reached
Resume: pmtu 1500 hops 17 back 12
```



# The *netcat* (`nc`) command

- The **netcat** command is a utility to test network connections across hosts.
- `$ nc -l 4545 # open a listen daemon to port 4545 on the local machine.`
- `$ nc 192.168.0.8 4545 # connect to port 4545 on the machine above. If you type something here it will appear on the remote machine.`
- `$ nc -zv 192.168.0.8 80 # check if port 80 is # open.`  
Connection to 192.168.0.8 80 port [tcp/http] succeeded!

**Note:** connections with **netcat** are unencrypted!

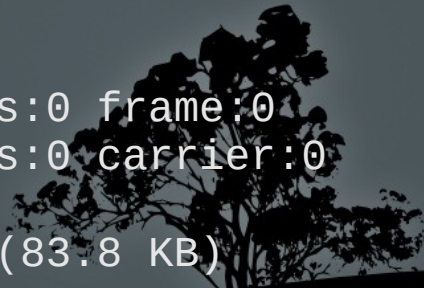


# Configure a network interface with `ifconfig`

- The **ifconfig** command is a legacy utility that can be used to configure a network interface and display network card settings. This configuration is not persistent!
- **# ifconfig #** show only active network interfaces

```
eth0 Link encap:Ethernet HWaddr 00:1c:25:9b:19:65
 inet addr:192.168.2.10 Bcast:192.168.2.255 Mask:255.255.255.0
 inet6 addr: fe80::21c:25ff:fe9b:1965/64 Scope:Link
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:5364 errors:0 dropped:0 overruns:0 frame:0
 TX packets:5047 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:5400323 (5.4 MB) TX bytes:785883 (785.8 KB)
 Interrupt:20 Memory:fc200000-fc220000

lo Link encap:Local Loopback
 inet addr:127.0.0.1 Mask:255.0.0.0
 inet6 addr: ::1/128 Scope:Host
 UP LOOPBACK RUNNING MTU:16436 Metric:1
 RX packets:260 errors:0 dropped:0 overruns:0 frame:0
 TX packets:260 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:0
 RX bytes:83809 (83.8 KB) TX bytes:83809 (83.8 KB)
```



# Configure a network interface with `ifconfig`

- `# ifconfig -a # show inactive interfaces as well.`
- `# ifconfig eth0 # show eth0 network settings.`
- `# ifconfig eth0 192.168.0.34 netmask 255.255.255.0 # set IP and netmask for eth0.`
- `# ifconfig eth0 192.168.0.34 netmask 255.255.255.0 \ broadcast 192.168.0.255 # set IP, netmask and broadcast address in eth0.`
- `# ifconfig eth0 down # disable eth0.`
- `# ifconfig eth0 up # enable eth0.`
- `# ifconfig eth0 up 192.168.0.34 netmask 255.255.255.0 # configure the network interface and enable it at the same time.`



# The `netstat` command

- The **netstat** is a legacy utility for displaying active connections, active ports, the routing table and detailed statistics about network usage.

## Options:

- `-i` # interfaces list with statistics.
- `-s` # detailed per protocol statistics.
- `-a` # show all listening ports and active connections.
- `-l` # show listening ports.
- `-p` # show the process behind each connection or listening port.
- `-r` # show routing table.
- `-n` # numeric results. It does not resolve hostnames which means faster results.
- `-t` # show TCP connections.
- `-u` # show UDP communication.
- `-c` # repeatedly show results every second.

**NOTE:** in modern systems **netstat** is being phased out by the **ss** utility.



# The `netstat` command

- `# netstat #` show all sockets, TCP, UDP and unix.
- `# netstat -tuc #` continues update of TCP and UDP traffic.
- `# netstat -tun #` numeric display of TCP and UDP traffic.
- `# netstat -an #` show connections and ports in numeric form.
- `# netstat -lnptu #` show listening TCP and UDP ports, in numeric form, along with programs that occupy these ports.
- `# netstat -r #` show routing table.
- `# netstat -i #` show interface and statistics.
- `# netstat -s #` detailed, per protocol statistics.

# Configure a Linux system as a router

- Most Linux systems are not set as a router. This means that if a packet arrives at an interface it cannot be forwarded from another interface.
- `# echo '1' > /proc/sys/net/ipv4/ip_forward # set IPv4 packet forwarding (non persistent).`
- `# echo '1' > /proc/sys/net/ipv6/conf/all/forwarding # set IPv4 packet forwarding (non persistent).`
- For a more persistent solution set the following variables in the `/etc/sysctl.conf` file:

```
net.ipv4.ip_forward = 1
net.ipv6.conf.all.forwarding=1
```



# Configure static routing with `route`

- The **route** command is a legacy utility for showing the routing table and also for adding and removing static routes.
- `$ route # show routing table with hostnames.`
- `$ route -n # show routing table numerically (IP Addresses).`
- `# route add default gw 10.0.2.2 eth0 # set 10.0.2.2 as default gateway.`
- `# route add -net 10.200.0.0 netmask 255.255.0.0 gw 10.10.10.250 # set the 10.10.0.0/16 network over the 10.10.10.250 gateway into the routing table.`
- `# route del -host 10.5.4.6 netmask 255.255.255.0 gw 172.16.1.1 # delete host 10.5.4.6 with gateway 172.16.1.1 from the routing table.`

# Connect to FTP servers with `ftp`

- The `ftp` command is a client for connecting to ftp servers via CLI.
- `# ftp ftp.debian.org #` connect to ftp.debian.org.
- `# ftp -v ftp.debian.org #` connect in verbose mode.
- Commands:
  - `ftp> ls #` list files/directories.
  - `ftp> cd dir #` change into directory `dir`.
  - `ftp> get file1 #` get file `file1`.
  - `ftp> mget file[1-9] #` get multiple files `file1`, `file2`, ..., `file9`.
  - `ftp> put file2 #` upload file `file2` from local working directory.
  - `ftp> mput file[a-f] #` upload multiple files.
  - `ftp> pwd #` print working directory on server.
  - `ftp> quit # = exit`. Exit ftp server.

# Connect to services with `telnet`

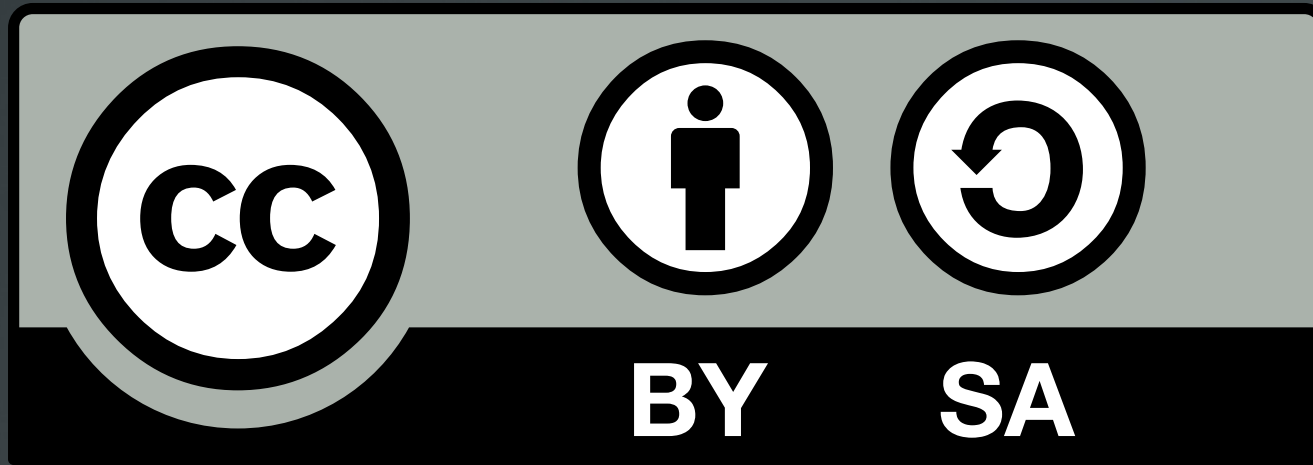
- The **telnet** command was used in the past for shell access on remote nodes. Because of its inherent weakness to send everything in cleartext, it was replaced by **ssh** which supports encryption.
- Nevertheless it is a useful troubleshooting tool for non encrypted services like HTTP, SMTP etc.
- `$ telnet telehack.com # connect to telehack.com.`
- `$ telnet www.debian.org 80 # connect to the debian  
# webserver for  
# checking the service.`
- `GET`
- `$ telnet mail.theo-andreou.org 25 # connect to  
# mailserver for basic health check.`
- `quit`
- `$ telnet towel.blinkenlights.nl # try and see! :).`

# Search for domain and IP owners with `whois`

- The **whois** command sends queries in domain registries and IP assigning authorities for finding who is responsible.
- `$ whois lpi.org # search for people or organizations responsible for the lpi.org domain.`
- `$ whois 8.8.8.8 # search for people or organizations responsible for the 8.8.8.8 domain.`
- `$ whois ellak.org.cy # alas it does not work for .cy domains!`



# License



The work titled "LPIC-1 102-500 – Lesson 15" by Theodotos Andreou is distributed with the Creative Commons Attribution ShareAlike 4.0 International License.

