How to Confirm which Ports are Open on Your Linux System

If you wish to see which ports are open on your Linux system, perhaps to check your configuration, you can use the nmap tool. It’s a powerful tool, but we’ll focus on just this simple task.

If you don’t have nmap, first install it. For example, on Ubuntu just run *sudo apt-get install nmap*. On Fedora it should be *sudo yum install nmap*. On Arch it should be *sudo pacman -Sy nmap*.

Once you’ve got nmap just run this simple command. Note that we’re running it with superuser privileges (*sudo*), which is necessary.

$ **nmap localhost**

Your output may look something like this:

```
Starting Nmap 6.40 ( http://nmap.org ) at 2014-11-26 23:56 CET
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0089s latency).
Other addresses for localhost (not scanned): 127.0.0.1
Not shown: 994 closed ports
PORT STATE SERVICE
21/tcp open ftp
22/tcp open ssh
53/tcp open domain
80/tcp open http
443/tcp open https
3306/tcp open mysql
```

So it shows you the open port numbers and the service that is using each. The above is pretty standard stuff. If you don’t see what you expected you should check your configuration.

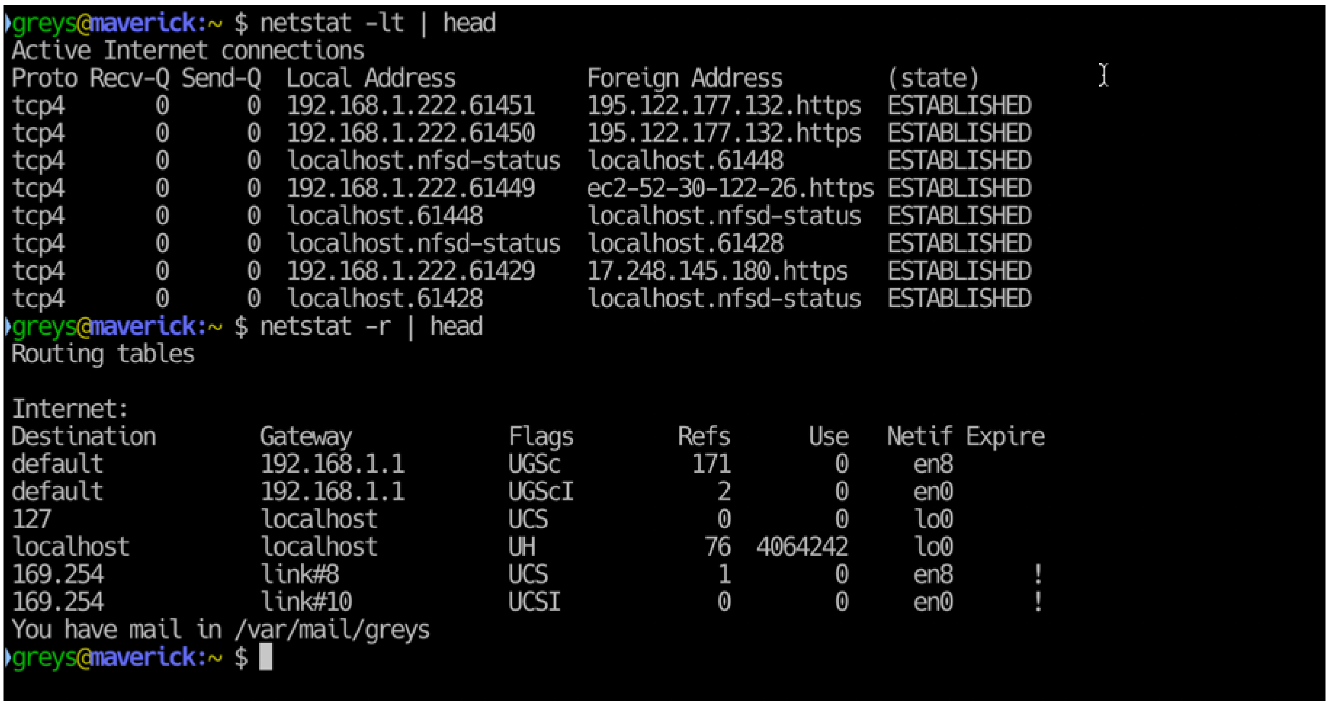
If you’d like to do more with nmap you can explore the nmap built in documentation by running *man nmap*, which contains a breadth of information.

See Also

- 5 ways to use netstat
- Advanced Unix commands

---

5 things you can do with netstat command



```
greys@maverick:~ $ netstat -lt | head
Active Internet connections
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  192.168.1.222.61451    195.122.177.132.https  ESTABLISHED
tcp4       0      0  192.168.1.222.61450    195.122.177.132.https  ESTABLISHED
tcp4       0      0  localhost.nfsd-status  localhost.61448        ESTABLISHED
tcp4       0      0  192.168.1.222.61449    ec2-52-30-122-26.https ESTABLISHED
tcp4       0      0  localhost.61448        localhost.nfsd-status  ESTABLISHED
tcp4       0      0  localhost.nfsd-status  localhost.61428        ESTABLISHED
tcp4       0      0  192.168.1.222.61429    17.248.145.180.https   ESTABLISHED
tcp4       0      0  localhost.61428        localhost.nfsd-status  ESTABLISHED
greys@maverick:~ $ netstat -r | head
Routing tables

Internet:
Destination        Gateway            Flags        Refs      Use   Netif Expire
default            192.168.1.1        UGSc          171        0     en8
default            192.168.1.1        UGScI           2        0     en0
127                localhost          UCS             0        0     lo0
localhost          localhost          UH             76  4064242     lo0
169.254            link#8             UCS             1        0     en8      !
169.254            link#10            UCSI            0        0     en0      !
You have mail in /var/mail/greys
greys@maverick:~ $ 
```

The netstat command, which stands for “network statistics”, can show you a lot of information about your network including statistics on connections to and from others on the network, used network interfaces, services, ports, and routing tables.

So what could all this information be used for? Just running *netstat* alone will give you an overview of your network, which will show a list of addresses connected to your system, over which port they're connected, and what services or programs they're talking to.

Here are five relatively simple examples of what you can actually do with netstat.

Show who is connected to your system

One of the most useful things you can do with netstat is show exactly who is connected to your system either through an incoming or outgoing connection (whether it is your system which initiated it or the other system). This will simply list all of them:

```
netstat -a
```

Look at the "Foreign Address" column to see where the connection is coming from, and "Local Address" to see what on the local machine is it connected.

The following command will show just the TCP (*-t*) and UDP (*-u*) connections:

```
netstat -tua
```

If you want to turn off hostnames, or domain names, and display only IP numbers just add the *-n* option.

```
netstat -tuan
```

If you want it to display this continuously to see as connections come and go add the *-c* option.

```
netstat -tuanc
```

Needless to say, perhaps, with IP addresses of everyone connecting revealed you can use other tools like traceroute to determine where exactly is it coming from.

Show listening ports with netstat

If you'd like to see which services are actually listening for incoming connections, perhaps to ensure you don't have something listening that you don't want to be listening, just use the *-l* option.

```
netstat -l
```

You can also limit this to only a specific type of traffic, like TCP in this example (for UDP just use *-u*):

```
netstat -lt
```

Find the port used by a program

We can get a little bit more specific by combining the netstat command with other common UNIX utilities like grep, in this example, where we make it easier to find which port is used by a program. We use grep to conveniently dig this info out of the netstat output:

```
netstat -ap | grep znc
```

In this example we get a list of all connections mentioning ZNC with the ports it is using, and addresses it is connected to.

Show the network routing table

With netstat you can easily see the kernel IP routing table being used on your system using the -r option:

```
netstat -r
```

Show all netstat statistics

Being a statistics utility you can of course see a summary of a great number of statistics about your system's networking.

Just run the netstat command with the -s option:

```
netstat -s
```

This will display a huge list of statistics, but you’ll immediately recognize the most interesting ones depending on what you’re looking for. For example you can see a total number of packets received, number of active TCP connections, and a number of extended more detailed statistics for each protocol.

Note

These examples are based on netstat in Linux, where it has been succeeded by the *ss* command from the iproute2 package, but it should apply to most UNIX and UNIX like systems. You can also check the manual page readily available via the *man netstat* command for more information.

See Also

- netstat unix
- Advanced Unix commands

How to capture network traffic with tcpdump

With tcpdump you can intercept, read, and save TCP/IP packets flowing through a particular network interface. These packets, which are the fundamental unit of data being transmitted over a TCP/IP network such as the internet, consist of two kinds of data. One is control data and the other is user data. Control data is the information about where the user data is to be delivered, where it’s coming from, what is its length, and other information about the actual user data. The user data is the actual data being transmitted, which could include just about anything. It could even include passwords and usernames if this data is sent in clear text and not encrypted.

Simply running tcpdump on the command line will capture and display packets flowing through the eth0 network interface, which is the typical default interface used. However, it will only be indiscriminately listing packets with their control data, and you wont actually see any user data. To display that you'll need to run tcpdump with the -X option:

```
tcpdump -X
```

To make what you're getting more useful though we can use a few options. For example, we could save this stuff in a file instead of having it just be dumped on our screen, which makes it pretty hard to read anyway:

```
tcpdump -X -w packets.txt
```

Once you run this your packets.txt file will start getting filled up with lots of information really quickly so long as there's any traffic flowing through eth0. Let's say that you're running a web server and someone visits your web site. You would see the HTML contents of the web page being requested in the packets.txt file as user data of that packet. You see everything that's being transmitted. If what is being transmitted is by any chance encrypted though you might only see incomprehensible gibberish, but not making it easy to discern what's being transmitted by intercepting these packets is the whole point of encryption.

What if you wanted to read another network interface, like eth1? Simple, just tell it to capture eth1 packets with the -i option:

```
tcpdump -X -w packets.txt -i eth1
```

To listen for any and all traffic, just use *-i any* instead, and it will listen to all network interfaces.

Here are a few more useful options that help you specify what you want to capture and have dumped by tcpdump. To see all of the options you can check the manpage by running *man tcpdump*.

To disable resolving hostnames and domains, which can save a bit of time, and display only IP addresses use the *-n* option. To disable port names, use *-nn*. With these options the first example would look like this:

```
tcpdump -Xnn
```

To show only a certain number of packets and then stop instead of running indefinitely you can specify the *-c 20* option, where -c stands for “count”, and “20” would represent 20 packets.

```
tcpdump -Xnnc 20
```

Finally, if you want to make absolutely sure you see the maximum possible information that is being captured use the verbosity options. You can increase verbosity up to three times. With just *-v*, *-vv*, or *-vvv* for maximum verbosity. Also, we can use the *-S* option to show absolute rather than relative sequence numbers just to make sure we see the actual numbers. So let’s construct a command that would show the maximum possible information on a sample of 100 packets, and store it into packets.txt.

```
tcpdump -XSvvvc 100 -w packets.txt
```

And that should get you on the right track to playing with and learning network traffic capture with tcpdump.