Modern Linux distributions also ship with a command mtr, which is similar to traceroute but shows real-time data which keeps refreshing. It is very useful to check your network carrier quality and so on.

Listing all the machines alive on a network

When we deal with a large local area network, we may need to check the availability of other machines in the network. A machine may not be alive in two conditions: either it is not powered on, or due to a problem in the network. By using shell scripting, we can easily find out and report which machines are alive on the network. Let's see how to do it.

Getting ready

In this recipe, we use two methods. The first method uses `ping` and the second method uses `fping`. `fping` is easier to use for scripts and has more features as compared to the `ping` command. Usually it won't be shipped with your Linux distribution by default, so manually install it using your package manager.

How to do it...

Let's go through the script to find out all the live machines on the network and alternate methods to find out the same.

1. The first method is as follows:

   We can write our own script using the `ping` command to query a list of IP addresses and check whether they are alive or not as follows:

```
#!/bin/bash
#Filename: ping.sh
# Change base address 192.168.0 according to your network.

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
```

The output is as follows:

```
$ ./ping.sh
192.168.0.1 is alive
192.168.0.90 is alive
```

2. Using `fping`, the second method is as follows:

   We can use an existing command-line utility to query the status of machines on a network as follows:

   ```
   $ fping -a 192.160.1/24 -g 2> /dev/null
   192.168.0.1
   192.168.0.90
   ```

   Or, use:

   ```
   $ fping -a 192.168.0.1 192.168.0.255 -g
   ```

How it works...

In the first method, we used the `ping` command to find out the alive machines on the network. We used a `for` loop for iterating through a list of IP addresses generated using the expression `192.168.0.{1..255}`. The `{start..end}` notation will expand and will generate a list of IP addresses, such as `192.168.0.1`, `192.168.0.2`, `192.168.0.3` up to `192.168.0.255`.

`ping $ip -c 2 &> /dev/null` will run a `ping` command to the corresponding IP address in each execution of the loop. The `-c` option is used to restrict the number of echo packets to be sent to a specified number. `&> /dev/null` is used to redirect both `stderr` and `stdout` to `/dev/null` so that it won't be printed on the terminal. Using `$?` we evaluate the exit status. If it is successful, the exit status is `0` else non-zero. Hence, the IP addresses which replied to our ping are printed.

In this script, each `ping` command for the IP address is executed one after the other. Even though all the IP addresses are independent of each other, the `ping` command is executed as a sequential program, it takes a delay of sending two echo packets and receiving them or the time-out for a reply for executing the next `ping` command.

There's more...

We discussed a method for finding out the alive machines on a network. Let's see some enhancements and another method to do the same thing.