Exercise 2.3: Building a DNS resolver

1. Install Unbound if it's not already there

```
    # dpkg --get-selections | grep unbound
```

If Unbound is already installed, the command above will display
some package names with "unbound" in their names. If it isn't, you
won't see anything, and you need to install it.

Another way of checking is to simply run the unbound command and
see whether it seems to be there. The -h option means "help" -- the
command below will not work if unbound is not installed.

```
    $ /usr/sbin/unbound -h
    usage:  unbound [options]
            start unbound daemon DNS resolver.
    -h      this help
    -c file config file to read instead of /etc/unbound/unbound.conf
    ...
    ... and so on
    ...
    Version 1.4.22
    ...
    ...
    $
```

If Unbound is not already installed, then install it:

```
    # apt-get update
    # apt-get install unbound
```

Unbound uses cryptographic keys to secure communication between the
unbound resolver itself and a control utility called "unbound-
control".
Happily, the unbound package comes with a script to set up
everything
we need for this to work.

```
    # unbound-control-setup
```

2. Start the server and check it is running

Then run these commands:

```
    # service unbound start
    # unbound-control status
```

You should see something like this:

```
    root@pc37:~# unbound-control status
    version: 1.4.22
    verbosity: 1
    threads: 1
    modules: 2 [ validator iterator ]
    uptime: 110 seconds
    unbound (pid 5013) is running...
    root@pc37:~#
```

It's also good practice to look in the system logs and make sure you don't see anything scary there:

```
    root@pc37:/var/log# tail syslog
    May 27 06:41:55 pc37 systemd[1]: Starting (null)...
    May 27 06:41:55 pc37 unbound-anchor: /var/lib/unbound/root.key
has content
    May 27 06:41:55 pc37 unbound-anchor: success: the anchor is ok
    May 27 06:41:55 pc37 unbound[4996]: Starting recursive DNS
server: unbound.
    May 27 06:41:55 pc37 unbound: [5013:0] notice: init module 0:
validator
    May 27 06:41:55 pc37 unbound: [5013:0] notice: init module 1:
iterator
    May 27 06:41:55 pc37 systemd[1]: Started (null).
    May 27 06:41:55 pc37 unbound: [5013:0] info: start of service
(unbound 1.4.22).
    May 27 06:41:55 pc37 systemd[1]: Starting Host and Network Name
Lookups.
    May 27 06:41:55 pc37 systemd[1]: Reached target Host and Network
Name Lookups.
    root@pc37:/var/log#
```

That all looks healthy. We don't see any lines describing errors or failures.

3. Reconfigure your resolver to use your own cache only

Edit `/etc/resolv.conf` as follows:

```
    search sse.ws.afnog.org
    nameserver 127.0.0.1
```

Remove any existing 'nameserver' lines, or comment them out by inserting '#' at the front. 127.0.0.1 is the loopback address; that is, an IP address which means 'send the packet to myself'.

4. Opening Unbound to external requests

By default, unbound will only process requests from localhost. We can
test this by trying to send a query from another machine, or from

our
laptops (for laptops that have dig):

```
  $ dig @pc37.sse.ws.afnog.org afnog.org mx
  ; <<>> DiG 9.8.3-P1 <<>> @pc37.sse.ws.afnog.org afnog.org mx
  ; (1 server found)
  ;; global options: +cmd
  ;; Got answer:
  ;; ->>HEADER<<- opcode: QUERY, status: REFUSED, id: 50290
  ;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
  ;; WARNING: recursion requested but not available

  ;; QUESTION SECTION:
  ;afnog.org.                   IN      MX

  ;; Query time: 27 msec
  ;; SERVER: 197.4.11.137#53(197.4.11.137)
  ;; WHEN: Wed May 27 07:46:57 2015
  ;; MSG SIZE  rcvd: 31

  [scallop:~]%
```

The return code "REFUSED" indicates that Unbound has refused to
answer our query. We can configure Unbound to accept requests from
other machines (but not ALL other machines! Why?)

We can expand the default access control list by creating a new
file as root in the directory /etc/unbound/unbound.conf.d called
clients.conf:

```
    server:
        interface: 0.0.0.0
        interface: ::0
        access-control: 196.200.219.0/20 allow
        access-control: 2001:43f8:220::/48 allow
```

The two interface lines instruct inbound to bind to all interfaces
on IPv4 (first line) and IPv6 (second line).

The first access-control line allows access from the local network
that contains the SSE servers over IPv4; the second line does the
same for IPv6.

To restart the server and make these changes take effect:

```
    # service unbound restart
```

5. Send some queries

Issue a query. Make a note of whether the response has the 'aa'
flag set. Look at the answer section and note the TTL of the
answer.

```
Also note how long the query took to process.

Then repeat the _exact same_ query, and note the information again.

    $ dig www.tiscali.co.uk.   Does it have the 'aa' flag?
______
                               What is the TTL of the answer?
______ seconds
                               How long is the Query Time?
______
                               milliseconds

    $ dig www.tiscali.co.uk.   Does it have the 'aa' flag?
______
                               What is the TTL of the answer?
______ seconds
                               How long is the Query Time?
______
                               milliseconds

Repeat it a third time. Can you explain the differences?

If your neighbour has got their cache working, then try sending
some queries to their cache (remember `dig @x.x.x.x ...`)


6. Watch the cache in operation
-------------------------------

You can take a snapshot of the cache contents.  To dump the cache,
change directory to somewhere you have write access (e.g. your home
directory) and use the unbound-control dump_cache command:

    # unbound-control dump_cache >cache.db

The cache is dumped direct to your terminal. This might well not
be the best idea -- you can also use normal Unix file redirection
to send the output to a file, where you can more easily look at it:

    # unbound-control dump_cache >/tmp/cache.db
    # more /tmo/cache.db

You can watch the cache making queries to the outside world using
`tcpdump` in a different window. We need to install tcpdump first:

    # apt-get install tcpdump

This is how we run it to watch for network traffic that matches the
expression "UDP port 53":

    # tcpdump -n -s 0 -vv udp port 53

While tcpdump is running, in the first window flush your cache (so
it forgets all existing data) and then issue some queries. Notice
```

the parameter to "unbound-control flush" is "." -- we are telling
Unbound to forget about everything it knows for every name from the
root down (i.e. flush the entire cache).

```
    # unbound-control flush .
    # dig www.tiscali.co.uk.   -- and watch tcpdump output. What do
you see?

    # dig www.tiscali.co.uk.   -- watch tcpdump again. This time?
```