

This project is due at 11:59:59pm on November 12, 2014 and is worth 10% of your grade. You must complete it with a partner. You may only complete it alone or in a group of three if you have the instructor's explicit permission to do so for this project.

Note that there is a milestone deadline for this project, at 11:59:59pm on November 5, 2014. More details are in the Milestone section below.

1 Description

The Domain Name System (DNS) is a hierarchical system for converting domain names (e.g., `www.google.com`) to Internet Protocol (IP) addresses (e.g., `209.85.129.99`). DNS is often referred to as a “phone book” for the Internet, translating human-friendly domain names into machine-friendly IP addresses. In this project, you will implement a DNS client program, which handles DNS requests by querying other machines. Note that the graduate version of this project has additional requirements, which serve as an opportunity for extra credit for students enrolled in the undergraduate version of this course.

2 Requirements

Your will write a DNS client program which, given a name to query for and a DNS server to query will:

- Construct a DNS query packet for the specified name
- Send the query to the specified DNS server using UDP
- Wait for the response to be returned from the server
- Interpret the response and output the result to `STDOUT`

Your client must support the following features:

- Queries for A records (IP addresses)
- Responses that contain A records (IP addresses) and CNAMEs (DNS aliases)

You should be strict; if the returned message does not conform to the DNS specification, you should assert an error. You may receive other packets that are not responses to your query; you should ignore these and continue to wait for a response to your query. Remember that network-facing code should be written defensively. We will test your code by sending corrupted packets to your client; you should handle these errors gracefully and *not* crash.

3 Your client program

For this project, you must use C. You may *not* use any DNS libraries in your project (e.g., `getaddrinfo` or `gethostbyname`). **You must construct the DNS request packet yourself, and interpret the reply yourself.**

3.1 Input and output

The command line syntax for your client is given below. The client program takes command line argument of the domain name to interpret and the IP address of the domain server to query. The syntax for launching your program is therefore:

```
./3600dns @<server:port> <name>
```

`port` (Optional) The UDP port number of the DNS server. Default value: 53.

`server` (Required) The IP address of the DNS server, in a.b.c.d format.

`name` (Required) The name to query for.

Your code should always use a query ID of 1337. After sending the request, your client should wait for a reply for 5 seconds. If no reply is heard within this time window, you should exit indicating that a timeout occurred, by printing out the `NORESPONSE` message.

To help us compare with the reference solution, your code must print out the packet to standard output using the provided `dump_packet` function. For example, if you have your packet in `buf` and it is `size` bytes, long, you should call `dump_packet(buf, size)` right before you call `sendto()`. You should see output like

```
[0000] 68 78 01 00 00 01 00 00 00 00 00 00 03 77 77 77  hx..... www
[0010] 06 67 6F 6F 67 6C 65 03 63 6F 6D 00 00 01 00 01  .google. com.....
```

Your client must then wait for a response from the server, and print the result to standard output using the following format:

```
IP <tab> <IP address> <tab> <auth|nonauth>
CNAME <tab> <alias> <tab> <auth|nonauth>
NOTFOUND
NORESPONSE
ERROR <tab> <description of the error>
```

If an response to a query contains multiple answers (such as multiple IP addresses or aliases), your client must print an IP or CNAME line for each one of these. If the requested name does not exist, your client must print a `NOTFOUND` line. If no response is ever received from the server (i.e., you've waited 5 seconds and not received anything), your client must print a `NORESPONSE` line. Finally, if any other error occurs, your client should print an `ERROR` line containing a description of the error.

3.2 Starter code

Very basic starter code for the assignment is available in `/course/cs3600f14/code/project3`. You must use this code as a basis for your project. Provided is a `3600dns` code and header file, and a `Makefile`. The `Makefile` is configured to correctly compile your code and to test your code on sample input.

To get started, you should copy down this directory into your own local directory (i.e., `cp -r /course/cs3600f14/code/project3 ~/`). You can compile your code by running `make`. You can also delete any compiled code and object files by running `make clean`.

3.3 Development

In this project, you will likely need to use *bit masking* to access certain bits of data you receive. For example, at one point, you will need to check whether the first bit of a `char a` is a 1. To check this, you can use the C bitwise AND (`&`) and bitwise OR (`|`):

```
unsigned char a = ...;
if (a & 0x80) { ... }
```

You can also use masking to *set* bits. For example, if you wanted to set the least significant bit of `a` to 0, you can do

```
a &= 0xfe;
```

You should develop your client program on the CCIS Linux machines, as these have the necessary compiler and library support. You are welcome to use your own Linux/OS X machines, but you are responsible for getting your code working, and your code *must* work when graded on the CCIS Linux machines. If you do not have a CCIS account, you should get one ASAP in order to complete the project.

Your code must be `-Wall clean` on `gcc`. Do not ask the TA for help on (or post to the forum) code that is not `-Wall clean` unless getting rid of the warning is what the problem is in the first place.

4 Extra credit (15 points)

For extra credit, you can also support queries for MX (mail server) and NS (name server) records. Therefore, your program should accept the following input syntax:

```
./3600dns [-ns|-mx] @<server:port> <name>
```

where the optional `-ns` or `-mx` flags request their respective records (if no flag is given, you should query the A record). Your output for these records should look like

```
MX <tab> <alias> <tab> <preference> <tab> <auth|nonauth>
NS <tab> <alias> <tab> <auth|nonauth>
```

5 Testing

We have set up a test DNS server which you should use while developing your code: `cs4700dns.ccs.neu.edu`, `129.10.112.152`. You should *not* send queries directly to any other DNS server (e.g., CCIS's DNS servers) until your code is reliably working when sending to our test machine. Otherwise, the server operators may view your packets as a security attack, with undesirable consequences for all involved.

You can use the `wireshark` utility in order to diagnose problems with packets that you send out (these will likely be malformed at the beginning). `Wireshark` will capture packets that you send and will let you view/explore the various fields. It will warn you about fields that are incorrect or missing, and can guide debugging your packets.

You can use the `dig` utility in order to help diagnose problems with interpreting responses from the DNS server that you query. To use it, see the man page, and an example of the output is shown below:

```
bash$ dig www.cnn.com

; <<>> DiG 9.7.3-P3 <<>> www.cnn.com
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 43304
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.cnn.com. IN A

;; ANSWER SECTION:
www.cnn.com. 116 IN A 157.166.226.25
www.cnn.com. 116 IN A 157.166.226.26
www.cnn.com. 116 IN A 157.166.255.18
www.cnn.com. 116 IN A 157.166.255.19

;; Query time: 170 msec
;; SERVER: 192.168.100.1#53(192.168.100.1)
;; WHEN: Mon Mar 12 08:55:51 2012
;; MSG SIZE rcvd: 93
```

You will note that the output includes a number of the DNS header fields, as well as the full contents of the question and answer sections (you can also see the output of the authority and additional sections via command-line arguments).

Additionally, we have included a basic test script to check the output of your code against our reference solution and check your code's compatibility with the grading script. If your code fails in the test script we provide, you can be assured that it will fare poorly when run under the grading script. To run the test script, simply type

```
bash$ make test
```

This will compile your code and then test your DNS client on a number of inputs, comparing the results against the reference solution. If any errors are detected, the test will print out the expected and actual output. For example, you might see something like

```
bash$ make test
./test
  Trying with script 'www.cnn.com' [FAIL]
  Diff in expected output:
< www.cnn.com 157.166.226.25
> blah
bash$
```

This indicates that the test with input `./cs3600` was expected to print out the line preceded by `<`, but instead returned `blah` (the output is simply a diff between the output of your shell and the reference shell). We include a few sample tests, but these are by no means exhaustive. We expect that you will create additional test to ensure that your programs behave as expected.

6 Submitting your project

6.1 Registering your team

You and your partner should first register as a team by running the `/course/cs3600f14/bin/register` script. You should pick out a team name (no spaces or non-alphanumeric characters, please) and run

```
/course/\coursedir/bin/register project3 <teamname>
```

This will either report back success or will give you an error message. If you have trouble registering, please contact the course staff.

You must register your team by 11:59:59pm on October 29, 2014.

6.2 Milestone

In order to ensure that you are making sufficient progress, you will have an interim milestone deadline. For the milestone, you must submit a draft of your README, which describes your overall approach. Your `3600dns` must be able to create a DNS request, but does not have to be able to interpret the response. In other words, your code must correctly generate a packet for an input query, and send it to the server, but does not have to do any more processing.

You should submit your milestone by running the `/course/cs3600f14/bin/turnin` script. Specifically, you should create a `project3` directory, and place all of your code in it. Then, run

```
/course/\coursedir/bin/turnin project3-milestone <dir>
```

Where `<dir>` is the name of the directory with your submission. The script will print out every file that you are submitting, make sure that it prints out all of the files you wish to submit! You should receive an email confirmation of your submission.

You must submit your milestone by 11:59:59pm on November 5, 2014. No slip days can be used on the milestone.

6.3 Final submission

For the final submission, you should submit you (thoroughly documented) code along with a plain-text (no Word or PDF) README file. In this file, you should describe your high-level approach, the challenges you faced, a list of properties/features of your design that you think is good, and an overview of how you tested your code.

You should submit your project by running the `/course/cs3600f14/bin/turnin` script. Specifically, you should create a `project3` directory, and place all of your code and README files in it. Then, run

```
/course/\coursedir/bin/turnin project3 <dir>
```

Where `<dir>` is the name of the directory with your submission. Again, the script will print out every file that you are submitting, make sure that it prints out all of the files you wish to submit! You should receive an email confirmation of your submission.

You must submit your project by 11:59:59pm on November 12, 2014.

7 Grading

The grading in this project will consist of

60% Program functionality

15% Error handling

15% Style and documentation

10% Milestone functionality

You are, however, going to be graded on how gracefully you handle errors. In other words, what will you do if you receive a corrupted response packet? Remember, network-facing code should be graded defensively; you should always assume that everyone is trying to break your program. To paraphrase John F. Woods, "Always code as if the [the remote machine you're communicating with] will be a violent psychopath who knows where you live."

8 Advice

A few pointers that you may find useful while working on this project:

- Remember to convert your integers, shorts, and longs to network ordering (using `hton()` and associated functions).
- Check the Piazza forum for question and clarifications. You should post project-specific questions there first, before emailing the course staff.
- Finally, get started early and come to the instructor's office hours and TA lab hours. You are welcome to come to the lab and work, and ask the TA any questions you may have.