COMP1521 18s2

Week 10 Lab Exercise
DNS Lookup

Computer System Fundamentals

Objectives

- to explore Linux IP address system calls

Admin

Grades	A+=outstanding, A=very good, B=adequate, C=sub-standard, D=hopeless
Demo	in the Week10 Lab or at the start of the **Week12** Lab (Week 11 is the Practice Exam)
Submit	`give cs1521 lab10 dns.c` or via WebCMS
Deadline	must be submitted by 11:59pm Sunday 7 October

Note: you need to do something *truly* outstanding, above and beyond the "call of duty" to get A+. Doing the exercise well and correctly as specified will get you an A grade. An A grade gets you full marks; an A+ grade gives a small bonus.

Background

The `host` command on Linux/Unix allows you to map between host names and IP addresses by doing DNS server lookups. For example:

```
$ host 127.0.0.1
1.0.0.127.in-addr.arpa domain name pointer localhost.
$ host www.cse.unsw.edu.au
www.cse.unsw.edu.au has address 129.94.242.51
$ host 129.94.242.51
51.242.94.129.in-addr.arpa domain name pointer albeniz.orchestra.cse.unsw.EDU.AU
```

The `host` command is useful, but also has a wide range of options, most of which you might never need. The aim of this lab is to write a simple version of a DNS lookup program that maps between hostnames and IP addresses.

To do this, you could use some simple library functions:

`struct hostent *gethostbyname(char *name)`

> Takes a hostname (e.g. `tuba00.cse.unsw.edu.au`) and returns a pointer to a `hostent` structure that contains, among other things, an array of IP addresses associated with the hostname. If it cannot resolve the hostname, it returns a NULL pointer.

`struct hostent *gethostbyaddr(struct in_addr *ip, socklen_t len, int typ)`

> Takes an IP address, packaged in an `in_addr` structure, the size of the structure and an address family, and returns a a pointer to a `hostent` structure that contains, among other things, an array of IP addresses associated with the hostname. If it cannot resolve the hostname, it returns a NULL pointer. For our purposes, the only useful value for `typ` is `AF_INET`.

`int inet_aton(char *addr, struct in_addr *ip)`

> Converts a string representing an IP address (e.g. `129.94.242.51`) and converts it to an internet address (`in_addr`) structure which is used in various network functions. If the `addr` looks like a "dotted-quad" IP address, then the function returns 1, otherwise it returns 0.

`char *inet_ntoa(struct in_addr *ip)`

Produces a printable string representation of an IP address stored in an `in_addr` structure. The string is dynamically allocated, and the function returns a pointer to the first character in the string.

You can find more information about these functions and the `hostent` structure by reading the Unix `man` pages for them. E.g. the command:

```
$ man 3 gethostbyname
```

will tell you about the two `gethost...()` functions and `hostent`.

The `in_addr` structure holds a binary representation of an IP address (the `in` stands for "internet").

Note that the `gethostbyname()` and `gethostbyaddr()` have been flagged as obsolete in the Unix documentation. However, they are simpler to use than the new and more general function that is replacing them (`getaddrinfo()`). If you wish, you are free to use the new function for this lab.

Setting Up

Create a directory for this lab; let's call it *Lab10Dir*.

Change into your *Lab10Dir* directory and run the following command:

```
$ unzip /home/cs1521/web/18s2/labs/week10/lab10.zip
```

If you're working at home, download `lab.zip` by right-clicking on the above link and then run the above command on your local machine.

If you've done the above correctly, you should now find the following files in the directory:

`dns.c`	skeleton for dns lookup program
`Makefile`	compiles the `dns` program

Note that, as supplied, the program will not compile. You will need to add some code before this happens.

Exercise

You need to write the rest of the main program so that it does DNS lookups and reports results as follows:

```
$ ./dns www.cse.unsw.edu.au
www.cse.unsw.edu.au -> 129.94.242.51
$ ./dns 127.0.0.1
127.0.0.1 -> localhost
$ ./dns 202.58.60.194
No name associated with 202.58.60.194
$ ./dns 129.94.242.51
129.94.242.51 -> albeniz.orchestra.cse.unsw.EDU.AU
$ ./dns abc.def.ghi
Can't resolve abc.def.ghi
$
```

To achieve this requires only about 15 lines of code, but you will need to understand the functions and data types mentioned above. The program will look something like this pseduo-code:

```
if (argv[1] looks like an IP address) {
   use gethostbyaddr() to determine its hostname
   if (no match)
      print error message
   else
      print "IP -> hostname"
}
else {  // might be a hostname
   use gethostbyname() to determine its IP address
   if (no match)
      print error message
   else
      print "hostname -> IP"
}
```

You can check your results by comparing the output from your program against the output of the `host` command. The outputs will not match exactly, but you should be able to determine whether your program has found one of the correct names or IP addresses.

Challenges

Print any aliases (either names or IP addresses) for the discovered host.

Submission

You need to submit the file: `dns.c`. You can submit this via the command line using `give` or you can submit it from within WebCMS. After submitting the code, show your tutor, who'll give you feedback on your work and award a grade.

Have fun, *jas*