Department of Information and Communication Technology
Network Programming

C Program to resolve domain names

Prepared by: DUONG Tuan Minh - BI11-172
Instructor: TRAN Giang Son
Date: May 25, 2022

Abstract

This document specifies a method and procedures to implement a DNS resolver program in C language. Specifically, two functions C functions gethostbyname() and inet_ntop() are used to resolve domain name to IP addresses and present the result.

Contents

1 Problem Analysis

1.1 Get domain name from user

- Domain name comes from program's arguments
- Domain name comes from user input

1.2 Resolve domain name

- Hostname is not found
- Hostname is resolved successfully

2 Get domain name from user

Initially, a char array of length 255 is declared to store the domain name

```
char domain[MAX_LENGTH];
```

2.1 Domain name comes from program's arguments

`argc` is the argument count. If `argc` is greater than one, there is at least one argument. We copy the second argument of `argv` (the first argument is the program name) into the `domain` char array

```
if (argc > 1)
{
    strncpy(domain, argv[1], MAX_LENGTH);
}
```

2.2 Domain name comes from user input

If no argument is provided, we prompt the user to enter a domain name and store the input into the `domain` char array

```
else
{
    printf("Enter a domain name: ");
    scanf("%255s", domain);
}
```

3 Resolve domain name

Utilizing the function `gethostbyname()`, we can resolve the `domain` and obtain either a pointer to a `struct hostent` or a null pointer.

3.1 Hostname is not found

If we obtain a null pointer, the domain name cannot be resolved. Then we inform the user and exit the program

```
if (host_ptr == NULL)
{
    printf("Cannot find address for hostname: %s\n", domain);
    return 0;
}
```

3.2 Hostname is resolved successfully

If we obtain a pointer to a `struct hostent`, we loop through its address list, convert each address to a char array using the function `inet_ntop()` and print the result

```
printf("Official hostname: %s\n", host_ptr->h_name);

char ip_addr[32];
int total_addr = sizeof(host_ptr->h_addr_list) /
sizeof(host_ptr->h_addr_list[0]);

for (int i = 0; i < total_addr; i++)
{
    printf("Address: %s\n", inet_ntop(host_ptr->h_addrtype,
    host_ptr->h_addr_list[i], ip_addr, sizeof(ip_addr)));
}
return 0;
```

4 Demonstration

Compile the program to an output file name `mynslookup`

4.1 Run the program locally

```
> ./mynslookup facebook.com
Official hostname: facebook.com
Address: 157.240.211.35

> ./mynslookup
Enter a domain name: facebook.com
Official hostname: facebook.com
Address: 157.240.211.35

> ./mynslookup randomtext
Cannot find address for hostname: randomtext

> ./mynslookup
Enter a domain name: randomtext
Cannot find address for hostname: randomtext
```

4.2 Run the program on a VPS in Singapore

```
> ./mynslookup facebook.com
Official hostname: facebook.com
Address: 157.240.235.35

> ./mynslookup
Enter a domain name: facebook.com
Official hostname: facebook.com
Address: 157.240.235.35

> ./mynslookup randomtext
Cannot find address for hostname: randomtext

> ./mynslookup
Enter a domain name: randomtext
Cannot find address for hostname: randomtext
```

The resolved IP address of facebook.com on my local machine `157.240.211.35` is different from the one from my VPS `157.240.235.35` because one domain name can map to multiple IP addresses and the geological distance can cause the mapping to behave differently in diffent region.