IP Dossier application project report

Preface: Our program structure consists of two main parts: The server program and a client program. The Server receives requests from the client through a request code ranging from 1-6, which it executes depending on the extra parameters passed. Communication protocol used here is TCP/IP. Program execution is done by first running the server program and then executing the client program to make sure the server is ready to accept the commands.

Main delivarables:

server.c : This is the server C program which handles the TCP requests from the client. It also handles the operations on the DNS file list by storing the DNS names, IP addresses and the number of times they're called in Linked Lists. The contents of the program are as follows:

a) **DieWithError():** Error handling function which is used to handle errors resulting from compilation or runtime errors.

b) **Searchtime():** This function uses a Linked List to store the time at which a client call is made each and every time. It compares consecutive times to determine whether multiple requests are made within the threshold of 60 seconds. If that's the case it calls the error function otherwise the time difference is printed.

c) **Insert():** It is called to insert a new line into the linked list as it is read from the DNS file.

d) **Delete():** This function deletes the particular node from the linked list.

e) **Traverse():** This node is used to travel through the linked list and select the required node. All the three functions Insert, delete and traverse are part of the standard Linked List definition required in any program.

f) **Gethostbyname():** This function is used to obtain the IP address of a particular domain by using the domain name entered by the user. It is a built in function and we have called it in case 1.

g) **Main():** In simple terms the main function consists of three parts.

   i) First part where it sets up a socket connection to listen for clients, this also includes a local address for server and client called echo.

   ii) The second part where the DNS IP Address file is opened, split into DNS names, IP Addresses and the number of times it is called and inserted into linked lists using the insert function.

iii) The third part is the most important part where it receives the client requests in the form of request codes from 1-6 which we use switch cases to handle all the different types of requests.

h) Command line arguments: The input from the user is extracted into a token variable and split to be used in the switch case.

i) Case 1: Finds an IP address for the requested domain name specified by the client

ii) Case 2: Uses linked lists to add a new record to the list. The client specifies the record and we use the search structure to search through the Linked list to see if it already exists in the list. If not we add the record using the insert() function. When the new record is added, the count value for the number of times its called is initialized to 0.

iii) Case 3: Deletes a particular record from the list using the delete().

iv) Case 4: Here we find the record which has been called the maximum number of times and return the value and the name of the particular record.

v) Case 5: Here we find the record which has been called the least number of times and return the value and the name of the particular record.

vi) **Case 6:** The token checks to see if the command line argument is same as the security code and shuts down the connection only if the security code matches. The output file for case 6 and the log file used for storing all the functions carried out by the server will be stored in the current project folder of the server.

vii) These paths can be modified to the current path name that is passed as the third command line argument at the server.

Client.c: The client program is fairly straightforward as it contains just a few functions which are necessary to process client requests and establishing a TCP/IP connection with the server. The main components are

a) **Validate():** This function is used to validate the IP address entered by the client during the command window phase. It checks for multiple scenarios to make sure the IP address is valid before passing it to the server to be evaluated.

b) **Main():** Similar to the server program, echo server addresses are given here as the local parameters.

I. The main function also has code for the error handling part before the command line arguments are passed to the server.

II. We check to make sure the right number of arguments are passed in the command line and that they're not less than or more than 6. And the individual scenarios are also addressed in case the arguments are equal to 5,6 or 7.

III. The similar condition is checked for the command code value entered at runtime. Once again we check for the appropriate passing of the number of bytes and make sure it is acceptable at the server end. The server sends an acknowledgement to the client once the data buffer is received and considering that the buffer size is within the acceptable dimensions the echo buffer is printed at the client side.

Output: We have included a few examples of the output screenshots below. This shows the successful operation of both our server and client.

Server in the beginning displaying the linked list from the text file

```
www.facebook.com 12 10.12.12.13
www.umd.edu 15 128.8.237.77
www.google.com 0 64.233.169.99
www.yahoo.com 5 98.138.253.109 206.190.36.45 98.139.183.24
```

Case 1 : searching for www.facebook.com

Client :

```
ARGC is 6servIP 127.0.0.1
echoServPort 1070
echoString www.facebook.com
command_code 1
Command sent
 Echostring
Received: 10.12.12.13
```

Case 2 : adding www.twitter.com to the linked list :

Client:

```
ARGC is 7servIP 127.0.0.1
echoServPort 1070
echoString www.twitter.com
command_code 2
command
Echostring
DomainIP  98.34.12.10
Received:
```

Server :

```
www.facebook.com 12 10.12.12.13
www.umd.edu 15 128.8.237.77
www.google.com 0 64.233.169.99
www.yahoo.com 5 98.138.253.109 206.190.36.45 98.139.183.24
Handling client 127.0.0.1
www.twitter.comNo record found

www.twitter.com 0 98.34.12.10
www.facebook.com 12 10.12.12.13
www.umd.edu 15 128.8.237.77
www.google.com 0 64.233.169.99
www.yahoo.com 5 98.138.253.109 206.190.36.45 98.139.183.24
```

Log File output:

logfile - Notepad

File Edit Format View Help

```
Mon Dec 19 12:54:18 2016
 www.bose.com  128.127.123.11 record added by client 127.0.0.1
Mon Dec 19 17:34:30 2016
  the ip adress for www.facebook.com is  10.12.12.13 requested by 127.0.0.1
Mon Dec 19 17:36:31 2016
  the ip adress for www.facebook.com is  10.12.12.13 requested by 127.0.0.1
Mon Dec 19 17:37:54 2016
 www.twitter.com  98.34.12.10 record added by client 127.0.0.1
```

Screenshots for binary search tree implementation :

Server :

c:\users\viraj deshpande\documents\visual studio 2015\Projects\Projectwithbinaryserach\Debug\Projectwithbinaryserach.exe

```
5
0
15
12
inorder Order Display
www.umd.edu 128.8.237.77 15
www.yahoo.com 98.138.253.109 206.190.36.45 98.139.183.24 5
www.google.com 64.233.169.99 0
www.facebook.com 10.12.12.13 12
Handling client 127.0.0.1
this is the echo buffer 1 www.yahoo.com
www.yahoo.comthis is the domain name www.yahoo.comIP address 98.139.183.24
IP address 98.139.180.149
98.139.180.149
```

Client :

C:\Users\Viraj Deshpande\documents\visual studio 2015\Projects\TCPIPCLIENTFINAL\Debug\TCPIPCLIENTFINAL.exe

```
ARGC is 6servIP 127.0.0.1
echoServPort 1070
echoString www.yahoo.com
command_code 1
Command sent
 Echostring
Received: 98.139.180.149
```

Note:

For the binary search tree portion, the linked list has been modified into a binary search tree where nodes are inserted and deleted using the following logic:

Since domain name for every entry is unique , its corresponding ASCII value is used as the value of the node. Any value greater than the root node value is stored to the right and a lesser value is stored to the left. This way the number of searches are reduced by half.