SSN College of Engineering, Kalavakkam
Department of Computer Science and Engineering
V Semester - CSE 'B'
UCS1511 NETWORKS LAB

Date :08/10/2020 **Name :** Rahul Ram M
Exercise : 05 **Reg No :** 185001121

DOMAIN NAME SERVER USING UDP

Learning Objective:

To simulate the concept of Domain Name Server using UDP.

Algorithm for Server:

1. Creating a socket using the function socket(domain, type, protocol) which the returns an integer as the status of the socket creation. Here the domain is AF_INET(iPv4 protocol), type is SOCK_DGRAM for UDP protocol .
2. Using bzero(&server_addr, sizeof(server_addr)) function setting values of all the socket structures to null.
3. Using bind() to binf the socket to the address and port number specified in addr(custom data structure). Here, we bind the server to the localhost, hence we use INADDR_ANY to specify the IP address.
4. Using addEntry() function in "dns.h" adding the Entry to the table.
5. Using printTable() printing the table.
6. Reading the option for modification of the table from the user.
7. IF yes, read the correct values of IP from the user and modify the table.
8. If correct value is not given, repeat asking for correct value.
9. Define a while loop that runs till ctrl+z is given
   - Recieve the request from the user using recvfrom() function.
   - Using the getAddress() function in "dns.h" get the correct address for the domain name.
   - Copy the result to result.
   - Send the result to the client using sendto() function.

Algorithm for Client:

1. Creating a socket using the function socket(domain, type, protocol) which the returns an integer as the status of the socket creation. Here the domain is AF_INET(iPv4 protocol), type is SOCK_DGRAM for UDP protocol and value as 0.
2. Using bzero(&server_addr, sizeof(server_addr)) function setting values of all the socket structures to null.
3. The above two steps are same as the server.
4. Setting a while loop which runs till 'exit' is given as message.
   1. Clearing the buffer using bzero().
   2. Reading the domain name from the user using scanf().
   3. If the message is 'exit', close the descriptor using close() and break the loop.
   4. Else send the domain name to the server using sendto() function.
   5. Using the recvfrom() function read the IP addresses sent by the server to the client.

6. If the buffer is empty, means domain is not found.
7. Else, Print the IP address of the domain in the correct format.

Program for Server:

```
#include <stdio.h>
#include <netdb.h>
#include <fcntl.h>
#include <unistd.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include "dns.h"

#define PORT 8080

int main()
{

        char result[100], opt[10], domain[20], address[20], buffer[1024];
        int sockfd;
        socklen_t len;
        struct sockaddr_in server_addr, client_addr;
        struct Entry* table = NULL;

        // SOCK_DGRAM -UDP
        if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
        {
                perror("Socker error");
                exit(1);
        }

        bzero(&server_addr, sizeof(server_addr));
        // assign IP, PORT
        server_addr.sin_family = AF_INET;
        server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
        server_addr.sin_port = htons(PORT);

        // Binding newly created socket to given IP and verification
        if ((bind(sockfd, (struct sockaddr *)&server_addr, sizeof(server_addr))) < 0)
        {
                perror("Bind error: ");
                exit(1);
        }

        len = sizeof(client_addr);
```

```
	addEntry(&table, "www.yahoo.com", "10.2.45.67");
	addEntry(&table, "www.annauniv.edu", "197.34.53.122");
	addEntry(&table, "www.google.com", "142.89.78.66");

	printTable(table);

	int flag = 0;
	printf("Do you want to modify (yes/no): ");
	scanf("%s", opt);
	if (strcmp(opt, "yes") == 0)
	{
		printf("Enter domain: ");
		scanf("%s", domain);
		do
		{
			printf("Enter IP address: ");
			scanf("%s", address);
			flag = addEntry(&table, domain, address);

		} while (flag != 1);

		printf("\nUpdated table\n");
		printTable(table);
	}

	while (1)
	{
		bzero(buffer, 1024);
		recvfrom(sockfd, buffer, sizeof(buffer), MSG_WAITALL, (struct
sockaddr*)&client_addr, &len);
		printf("Checking and sending IP address for %s\n", buffer);
		strcpy(result,getAddress(table, buffer));
		sendto(sockfd, &result, sizeof(result), MSG_CONFIRM, (struct
sockaddr*)&client_addr, len);
	}

	close(sockfd);
}
```

Program for Client:

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define PORT 8080
```

```
int main()
{
	int client_fd, count;
	struct sockaddr_in server_addr;
	socklen_t len;
	char buffer[1024], domain[30], address[10][20], temp[20];

	if((client_fd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
	{
		perror("Socket error");
	}

	bzero(&server_addr,sizeof(server_addr));

	server_addr.sin_family = AF_INET;
	server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
	server_addr.sin_port = htons(PORT);

	while(1)
	{
		bzero(buffer, 1024);
		printf("Enter the server name : ");
		scanf("%s", domain);

		if(strcmp(domain, "exit") == 0){
			close(client_fd);
			printf("Disconnected from server...\n");
			break;
		}
		else
		{
			sendto(client_fd, domain, sizeof(domain), MSG_CONFIRM, (struct
sockaddr *)&server_addr, sizeof(server_addr));
		}

		recvfrom(client_fd, &buffer, sizeof(buffer), MSG_WAITALL, (struct
sockaddr*)&server_addr, &len);

		if(strcmp(buffer, "") == 0)
		{
			printf("Domain not found!\n");
		}
		else
		{
			count = 0;
			strcpy(temp, "");
			for(int i = 0; i < strlen(buffer); i++)
```

```
                    {
                            if(buffer[i] == ' ')
                            {
                                    strcpy(address[count++], temp);
                                    strcpy(temp, "");
                            }
                            else
                            {
                                    strncat(temp, &buffer[i], 1);
                            }
                    }
                    strcpy(address[count++], temp);
                    printf("The IP address is : ");
                    for(int i = 0; i < count; i++)
                    {
                            printf("%s\n", address[i]);
                            if((count - i) != 1)
                            {
                                    printf("\t\t    ");
                            }
                    }
                    printf("\n");
            }
    }

    return 0;
}
```

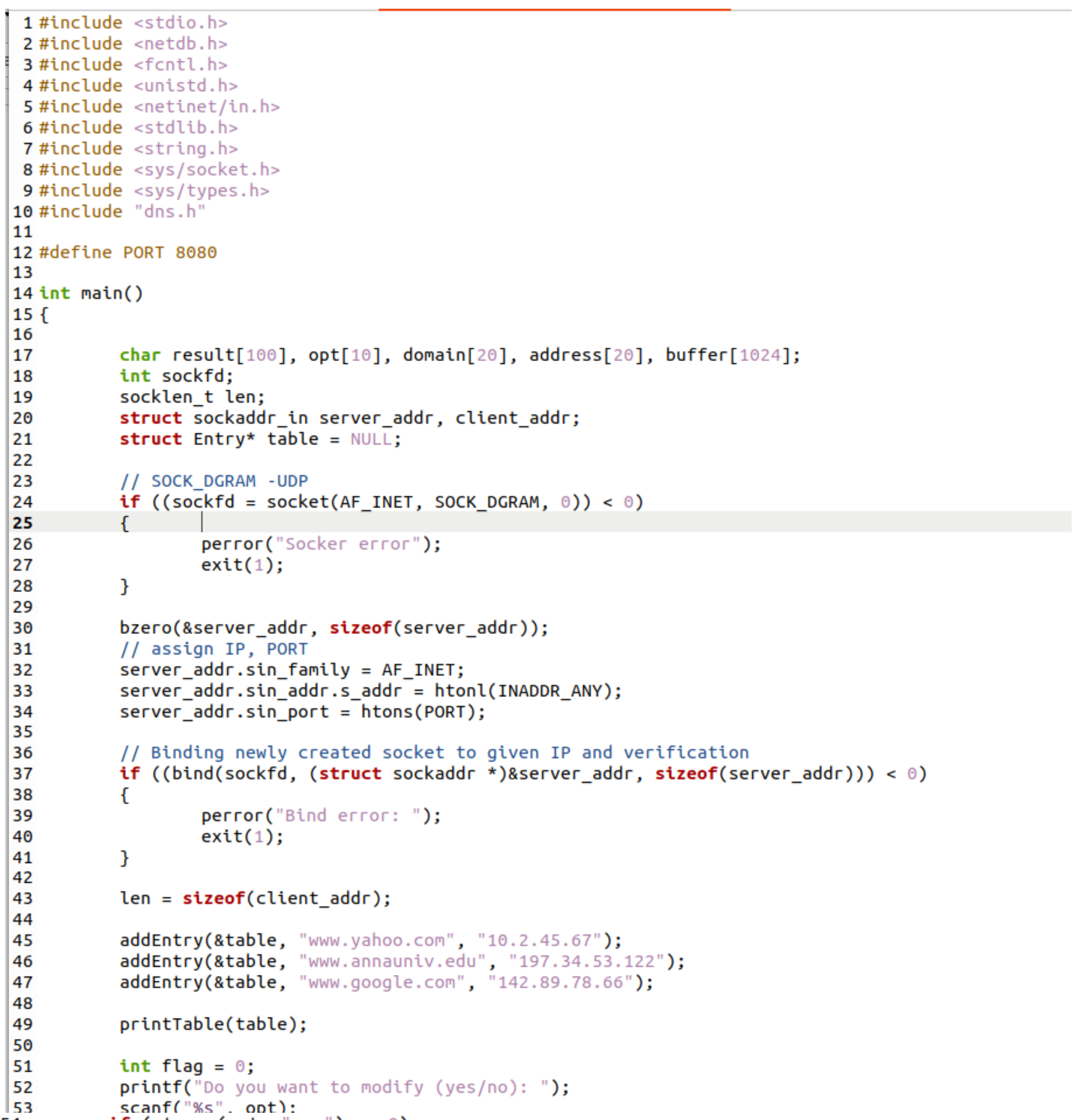
dns.h:

“performs all the functions related to dns”

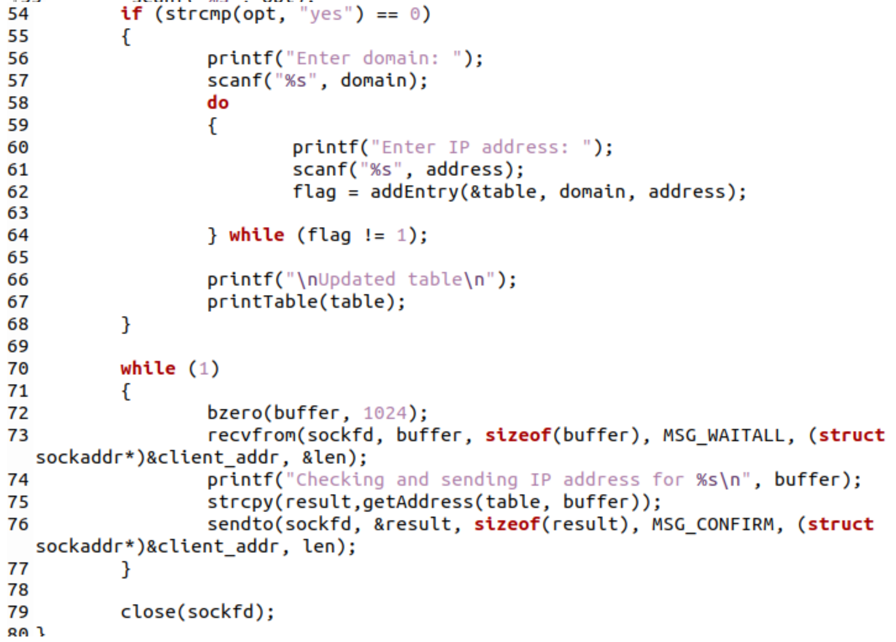
```
struct Entry
{
        char domain[20];
        char address[10][16];
        int count;
        struct Entry* next;
};

int searchDomain(struct Entry* head, char *domain)
{
        struct Entry *temp = head;
        if (head == NULL)
        {
                return 0;
        }
        while (temp != NULL)
        {
                if(strcmp(temp->domain, domain) == 0)
                {
```

```
                return 1;
            }
            temp = temp->next;
        }
        return 0;
}

int searchAddress(struct Entry* head, char *address)
{
        struct Entry *temp = head;
        if (head == NULL)
        {
            return 0;
        }
        while (temp != NULL)
        {
            for(int i = 0; i < temp->count; i++)
            {
                if(strcmp(temp->address[i], address) == 0)
                {
                    return 1;
                }
            }
            temp = temp->next;
        }
        return 0;
}

int checkIP(char *address)
{
        int count = 0;
        int num[5];
        for(int i = 0; i < strlen(address); i++)
        {
            if(address[i] == '.')
            {
                count++;
            }
        }

        if(count != 3)
        {
            return 1;
        }
        count = 0;
        char value[10];
        strcpy(value, "");
```

```
	for(int i = 0; i < strlen(address); i++)
	{
		if(address[i] == '.')
		{
			num[count++] = atoi(value);
			strcpy(value, "");
		}
		else
		{
			strncat(value, &address[i], 1);
		}
	}

	num[count++] = atoi(value);

	for(int i = 0; i < count; i++)
	{
		if(num[i] < 0 || num[i] > 255)
		{
			return 1;
		}
	}
	return 0;
}

void modifyEntry(struct Entry** head, char *domain, char* address)
{
	struct Entry* current = *head;  // Initialize current
	if(searchAddress(*head, address) == 1)
	{
		printf("Address already present!\n");
		return;
	}
	while (current != NULL)
	{
		if (strcmp(current->domain, domain) == 0)
		{
			strcpy(current->address[current->count++], address);
			return;
		}
		current = current->next;
	}
}

int addEntry(struct Entry** head, char *domain, char *address)
{

	if(checkIP(address) == 1)
	{
		printf("Invalid IP address!\n");
		return 0;
	}
	if(searchAddress(*head, address) == 1)
	{
		printf("Address already present!\n");
		return 0;
	}
	if(searchDomain(*head, domain) == 1)
	{
		modifyEntry(head, domain, address);
		return 1;
	}

	struct Entry* new_Entry = (struct Entry*) malloc(sizeof(struct Entry));
	struct Entry *last = *head;

	strcpy(new_Entry->domain, domain);
	new_Entry->count = 0;
	strcpy(new_Entry->address[new_Entry->count++], address);

	new_Entry->next = NULL;

	if (*head == NULL)
	{
		*head = new_Entry;
		return 1;
	}

	while (last->next != NULL)
		last = last->next;

	last->next = new_Entry;
	return 1;
}


char* getAddress(struct Entry* head, char *domain)
{
	static char address[100];
	strcpy(address, "");;

	if (head == NULL)
	{
		printf("List is empty.\n");
		return NULL;
	}
	struct Entry *temp = head;
	while (temp != NULL)
	{
		if(strcmp(temp->domain, domain) == 0)
		{
			for(int i = 0; i < temp->count; i++)
			{
				strcat(address, temp->address[i]);

				if((temp->count - i) != 1)
				{
					strcat(address, " ");
				}
			}
			break;
		}
		temp = temp->next;
	}
	return address;
}

void printTable(struct Entry *head) {
	struct Entry *temp;
	if (head == NULL) {
		printf("List is empty.\n");
		return;
	}
	//printf("Server domain\t\tIP address\n");
	printf("+-------------------+----------------------+\n");
	printf("|   Server Domain   |      IP Address      |\n");
	printf("+-------------------+----------------------+\n");
	temp = head;
	while (temp != NULL) {
		printf("| %-17s ", temp->domain);
		for(int i = 0; i < temp->count; i++)
		{
			printf("| %-20s |\n", temp->address[i]);
			if((temp->count - i) != 1)
			{
				printf("|\t\t    ");
			}
		}
		printf("+-------------------+---------------------+\n");
		temp = temp->next;
	}
	printf("\n");
}
```
Screenshot for Server:

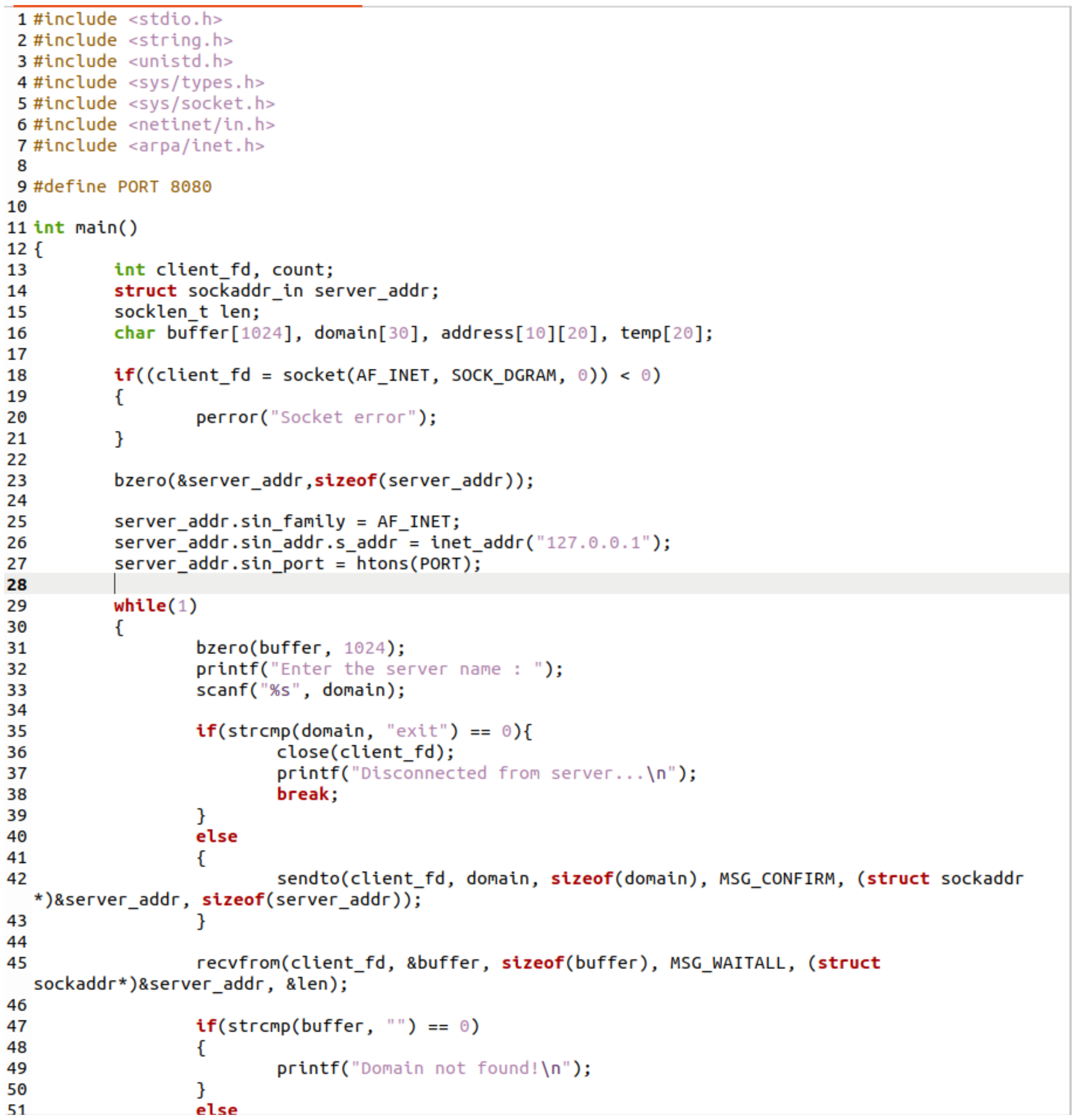


```c
#include <stdio.h>
#include <netdb.h>
#include <fcntl.h>
#include <unistd.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include "dns.h"

#define PORT 8080

int main()
{

        char result[100], opt[10], domain[20], address[20], buffer[1024];
        int sockfd;
        socklen_t len;
        struct sockaddr_in server_addr, client_addr;
        struct Entry* table = NULL;

        // SOCK_DGRAM -UDP
        if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
        {
                perror("Socker error");
                exit(1);
        }

        bzero(&server_addr, sizeof(server_addr));
        // assign IP, PORT
        server_addr.sin_family = AF_INET;
        server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
        server_addr.sin_port = htons(PORT);

        // Binding newly created socket to given IP and verification
        if ((bind(sockfd, (struct sockaddr *)&server_addr, sizeof(server_addr))) < 0)
        {
                perror("Bind error: ");
                exit(1);
        }

        len = sizeof(client_addr);

        addEntry(&table, "www.yahoo.com", "10.2.45.67");
        addEntry(&table, "www.annauniv.edu", "197.34.53.122");
        addEntry(&table, "www.google.com", "142.89.78.66");

        printTable(table);

        int flag = 0;
        printf("Do you want to modify (yes/no): ");
        scanf("%s", opt);
```

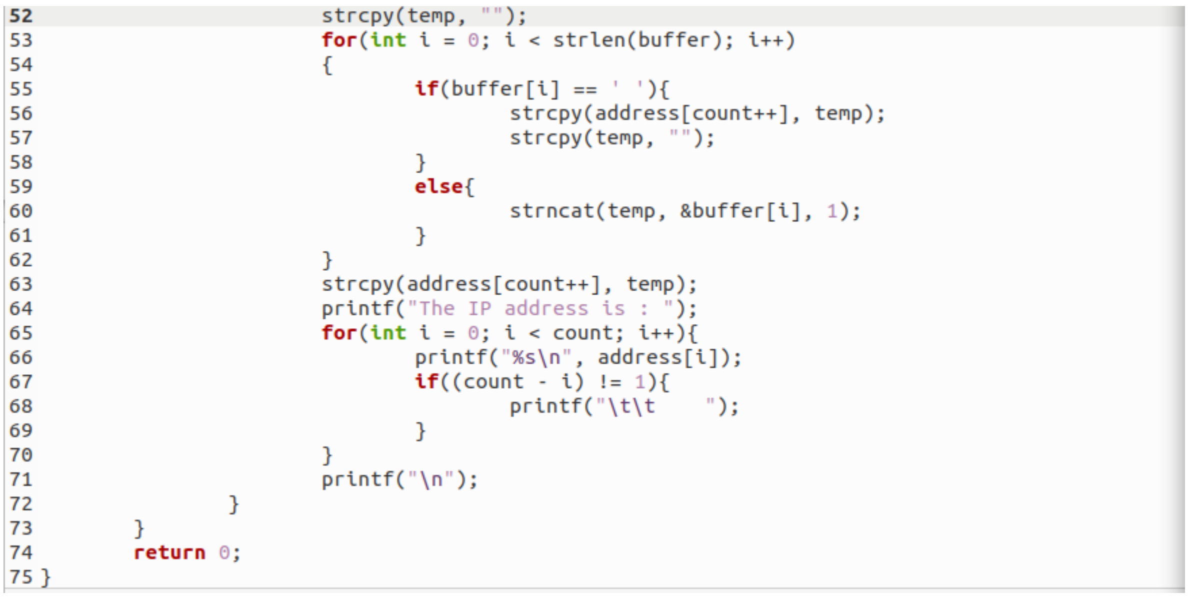


```c
        if (strcmp(opt, "yes") == 0)
        {
                printf("Enter domain: ");
                scanf("%s", domain);
                do
                {
                        printf("Enter IP address: ");
                        scanf("%s", address);
                        flag = addEntry(&table, domain, address);

                } while (flag != 1);

                printf("\nUpdated table\n");
                printTable(table);
        }

        while (1)
        {
                bzero(buffer, 1024);
                recvfrom(sockfd, buffer, sizeof(buffer), MSG_WAITALL, (struct
sockaddr*)&client_addr, &len);
                printf("Checking and sending IP address for %s\n", buffer);
                strcpy(result,getAddress(table, buffer));
                sendto(sockfd, &result, sizeof(result), MSG_CONFIRM, (struct
sockaddr*)&client_addr, len);
        }

        close(sockfd);
}
```

Screenshot for Client:

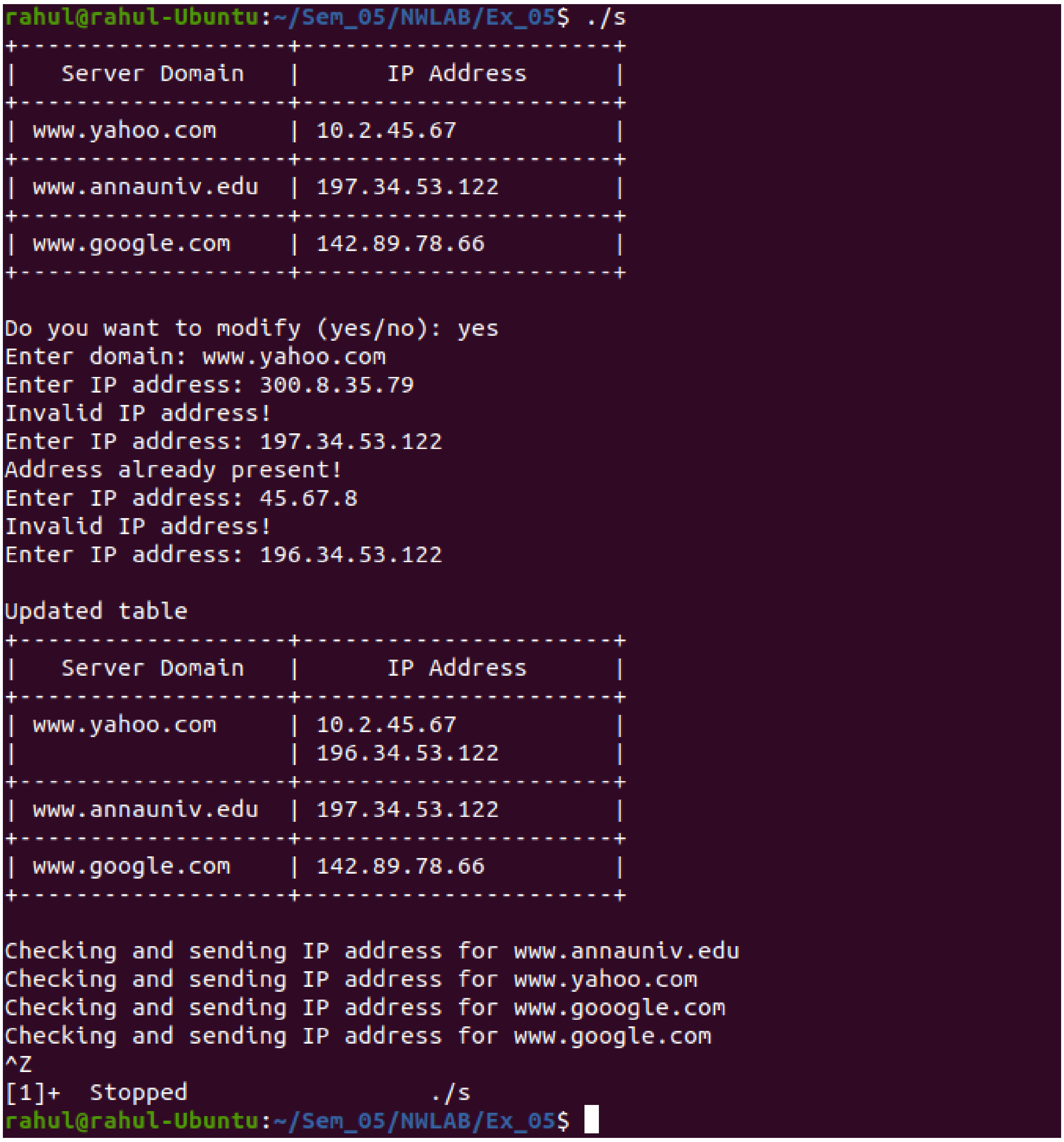


```c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define PORT 8080

int main()
{
        int client_fd, count;
        struct sockaddr_in server_addr;
        socklen_t len;
        char buffer[1024], domain[30], address[10][20], temp[20];

        if((client_fd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
        {
                perror("Socket error");
        }

        bzero(&server_addr,sizeof(server_addr));

        server_addr.sin_family = AF_INET;
        server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
        server_addr.sin_port = htons(PORT);

        while(1)
        {
                bzero(buffer, 1024);
                printf("Enter the server name : ");
                scanf("%s", domain);

                if(strcmp(domain, "exit") == 0){
                        close(client_fd);
                        printf("Disconnected from server...\n");
                        break;
                }
                else
                {
                        sendto(client_fd, domain, sizeof(domain), MSG_CONFIRM, (struct sockaddr *)&server_addr, sizeof(server_addr));
                }

                recvfrom(client_fd, &buffer, sizeof(buffer), MSG_WAITALL, (struct sockaddr*)&server_addr, &len);

                if(strcmp(buffer, "") == 0)
                {
                        printf("Domain not found!\n");
                }
                else
```



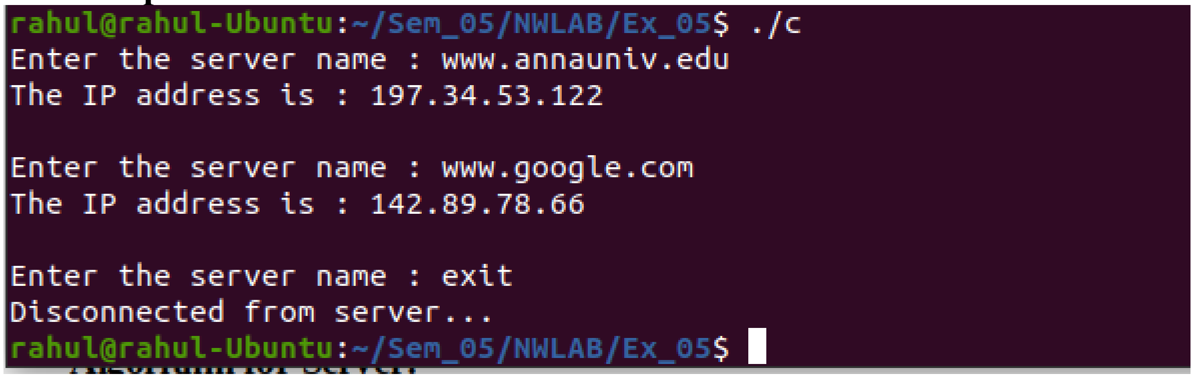
```c
                        strcpy(temp, "");
                        for(int i = 0; i < strlen(buffer); i++)
                        {
                                if(buffer[i] == ' '){
                                        strcpy(address[count++], temp);
                                        strcpy(temp, "");
                                }
                                else{
                                        strncat(temp, &buffer[i], 1);
                                }
                        }
                        strcpy(address[count++], temp);
                        printf("The IP address is : ");
                        for(int i = 0; i < count; i++){
                                printf("%s\n", address[i]);
                                if((count - i) != 1){
                                        printf("\t\t    ");
                                }
                        }
                        printf("\n");
                }
        }
        return 0;
}
```

Server Output:



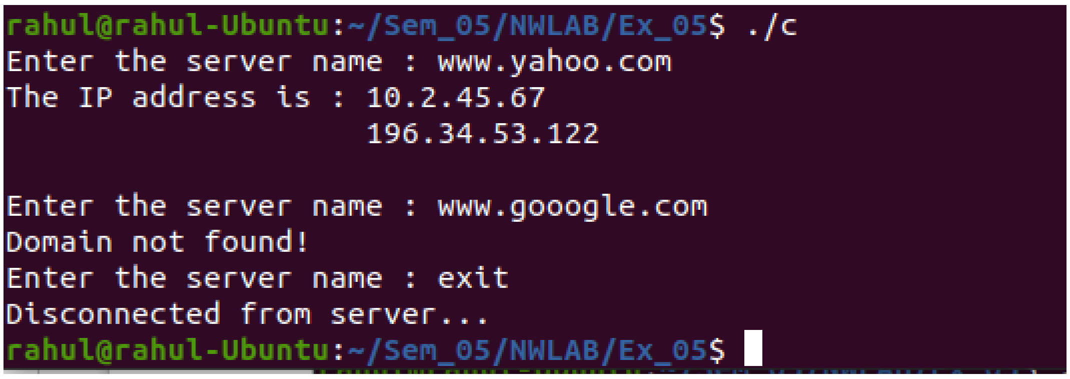
```
rahul@rahul-Ubuntu:~/Sem_05/NWLAB/Ex_05$ ./s
+--------------------+-----------------------+
|   Server Domain    |      IP Address       |
+--------------------+-----------------------+
| www.yahoo.com      | 10.2.45.67            |
+--------------------+-----------------------+
| www.annauniv.edu   | 197.34.53.122         |
+--------------------+-----------------------+
| www.google.com     | 142.89.78.66          |
+--------------------+-----------------------+

Do you want to modify (yes/no): yes
Enter domain: www.yahoo.com
Enter IP address: 300.8.35.79
Invalid IP address!
Enter IP address: 197.34.53.122
Address already present!
Enter IP address: 45.67.8
Invalid IP address!
Enter IP address: 196.34.53.122

Updated table
+--------------------+-----------------------+
|   Server Domain    |      IP Address       |
+--------------------+-----------------------+
| www.yahoo.com      | 10.2.45.67            |
|                    | 196.34.53.122         |
+--------------------+-----------------------+
| www.annauniv.edu   | 197.34.53.122         |
+--------------------+-----------------------+
| www.google.com     | 142.89.78.66          |
+--------------------+-----------------------+

Checking and sending IP address for www.annauniv.edu
Checking and sending IP address for www.yahoo.com
Checking and sending IP address for www.gooogle.com
Checking and sending IP address for www.google.com
^Z
[1]+  Stopped                 ./s
rahul@rahul-Ubuntu:~/Sem_05/NWLAB/Ex_05$ 
```

Client1 Output:



```
rahul@rahul-Ubuntu:~/Sem_05/NWLAB/Ex_05$ ./c
Enter the server name : www.annauniv.edu
The IP address is : 197.34.53.122

Enter the server name : www.google.com
The IP address is : 142.89.78.66

Enter the server name : exit
Disconnected from server...
rahul@rahul-Ubuntu:~/Sem_05/NWLAB/Ex_05$ 
```

Client2 Output:



```
rahul@rahul-Ubuntu:~/Sem_05/NWLAB/Ex_05$ ./c
Enter the server name : www.yahoo.com
The IP address is : 10.2.45.67
                    196.34.53.122

Enter the server name : www.gooogle.com
Domain not found!
Enter the server name : exit
Disconnected from server...
rahul@rahul-Ubuntu:~/Sem_05/NWLAB/Ex_05$
```

Learning Outcomes:

This assignment helped me to

1. Write program for server and client with socket programming using UDP protocol.
2. Understand various functions invloved in creating, estabilishing, maintaining, Sending, recieving and termininating the connection between the server and client.
3. Write code to make server and client communicate with each other using readfrom() and sendto() functions.