Behiye Erdemir
240206013

CENG 421- Assignment 4

For the assignment, I created a simple UDP server. Firstly, I will set up the #include statements and defined the sizes of message and buffer as 1KB.

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

After that, there are initializations of variables.

```
int main(int argc, char* argv[]){
    int udpSocket,bufferSize,messageSize;
    int returnStatus = 0;
    struct sockaddr_in udpServer,udpClient;
    char buff[SIZE];
    char message[SIZE];

    socklen_t lengthofClient;
    lengthofClient = sizeof(udpClient);
```

It is need to be sure, there are the right number of arguments. If it is not provided, the code will be exit.

```
    /* check for the right number of arguments */
    if (argc < 2) {
        fprintf(stderr, "Usage: %s <port>\n", argv[0]);
        exit(1); }
```

By using the `SOCK_DGRAM`, I created a UDB socket and checked that it is created.

```
    /*Create a socket*/
    udpSocket=socket(AF_INET,SOCK_DGRAM,0);
    if(udpSocket == -1) {
        fprintf(stderr, "Could not create a socket!\n");
        exit(1);
    }
    else{
        printf("Socket created.\n");
    }
```

After, I populate the sockaddr_in structure reserved for the server with the information I have.

```
udpServer.sin_family = AF_INET;
udpServer.sin_addr.s_addr = htonl(INADDR_ANY);
udpServer.sin_port = htons(atoi(argv[1]));
```

And, I set messages to a known value before using them.

```
memset(buff,'\0',SIZE);
memset(message,'\0',SIZE);
strcpy(message,"\nAnybody there? ");
messageSize=strlen(message);
write(0,message,messageSize);
```

Next, there are check operations for binding, receiving and sending. In this block, server sends the time and date information to the client and gets the message that send by the client.

```
returnStatus=bind(udpSocket,(struct sockaddr*)&udpServer,sizeof(udpServer));
    /*check for binding*/
    if(returnStatus==0){
        for(;;){
            time_t date;
            time(&date);

            /*check for receiving*/
            if((recvfrom(udpSocket,buff,sizeof(buff),0,(struct
sockaddr*)&udpClient,&lengthofClient))<0){
                 memset(message,'\0',SIZE);
                 strcpy(message,"\nOops, error in receiving");
                 messageSize=strlen(message);
                 write(0,message,messageSize);
                }

            memset(message,'\0',SIZE);
            strcpy(message,"\nI got the message: ");
            messageSize=strlen(message);
            write(0,message,messageSize);
            write(0,buff,strlen(buff));

            /*check for sending*/
            if((sendto(udpSocket,ctime(&date),strlen(ctime(&date)),0,(struct
sockaddr*)&udpClient,sizeof(udpClient)))<0){
                memset(message,'\0',SIZE);
                strcpy(message,"\nOops,Error in sending");
                messageSize=strlen(message);
                write(0,message,messageSize);
               }

        }
```

```
    }

    else{
        memset(message,'\0',SIZE);
        strcpy(message,"\nOops,Error in bind");
        messageSize=strlen(message);
        write(0,message,messageSize);
    }
}
```

For the UDP client, I will set up the #include statements and defined the sizes of message and buffer as 1KB.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>

#define BUFF 1024
```

After that, there are initializations of variables.

```
int main(int argc, char* argv[]){

    int udpSocket;
    struct sockaddr_in udpClient, udpServer;
    socklen_t servlen;
    char buff[BUFF];
```

It is need to be sure, there are the right number of arguments. If it is not provided, the code will be exit.

```
if (argc < 3) {
        fprintf(stderr, "Usage: %s <ip address> <port>\n", argv[0]);
        exit(1);
        }
```

By using the `SOCK_DGRAM`, I created a UDB socket and checked that it is created.

```
    udpSocket = socket(AF_INET,SOCK_DGRAM,0);
    if(udpSocket == -1){
        fprintf(stderr, "Could not create a socket!\n");
        exit(1);
        }
    else {
        printf("Socket created.\n");
        }
```

After, I populate the sockaddr_in structure reserved for the server with the information I have.

```
udpClient.sin_family = AF_INET;
    udpClient.sin_port = htons(atoi(argv[2]));
    udpClient.sin_addr.s_addr = INADDR_ANY;
```

In this block, client can send a message to the server and get the time and date information.

```
for(;;){
        printf("Enter the string : ");
        scanf("%s",buff);
        if((sendto(udpSocket,buff,strlen(buff),0,(struct
sockaddr*)&udpClient,sizeof(udpClient)))<0){
            printf("Error");
            }

        recvfrom(udpSocket,buff,sizeof(buff),0,(struct
sockaddr*)&udpClient,&servlen);
        printf("Time is %s\n",buff);
    }

}
```

Because of my misunderstanding, I could not accomplish the assignment according to the answer of the question of " time or date?" to be determined by the client. If I had time, I would compare the string specified by the client with the strings assigned time and date on the server side, and send the correct information to the client.

Server side,

```
Behiye-MacBook-Pro:backyard behiyeerdemir$ ./a.out 4444
Socket created.

Anybody there?
I got the message: hello
```

Client side,

```
Behiye-MacBook-Pro:client behiyeerdemir$ ./a.out 127.0.0.1 4444
Socket created.
Enter the string : hello
Time is Sun Sep  6 23:50:40 2020

Enter the string :
```

References

[1] J. W. T. a. N. Y. Keir Davis, The Definitive Guide to Linux Network Programming.

[2] «TCP SOCKET(DATE AND TIME) in C,» Forget Code, [Online]. Available: https://forgetcode.com/c/1476-tcp-socket-date-and-time.

[3] R. Rajput, «udp-time-server-client,» [Onlline]. Available: https://github.com/iamrahul404/udp-time-server-client.