

# CSE1004 NAC LAB

## ASSIGNMENT NO. – 4

### SOCKET PROGRAMMING EXERCISES

DATE – 27/01/22

NAME – AYUSHI TRIVEDI

REGISTRATION NUMBER – 20BPS1135

1. Simulate an online English dictionary using sockets. A user searches for meaning of a word in the dictionary which is present in server. The server responds to the user with the corresponding meaning of the word.

### SERVER:

#### ALGORITHM:

1. Reading server **port** number, declare an string for storing time
2. Creating a socket using **socket()** function
3. Binding the socket to the server address using the **bind()** function
4. Reading max number of clients for which server has to wait until that many requests are requested from the client.
5. Keep the server in passive mode by using **listen()** function, in which the server has to wait until there is a request from the client.
6. Connection is established between server and client using **accept()** function and then now both are ready to send and receive data.
7. Using **recv()** function to receive data from the client and store it in the buffer.
8. Create a for loop and run through all the words of dictionary structure and compare it with string obtained from client using **strcmp()** function
9. If strcmp returns 0, send the corresponding meaning stored in structure to the client.
10. If the strcmp doesn't return 0 after running through all the words stored in dictionary structure, send "No words found" to Client.
11. Sending the message to client.
12. Closing server socket using **close()** function.
- 13.

## **CODE:**

```
#include <stdio.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <string.h>

#include <stdlib.h>

#include <sys/socket.h>

#include <unistd.h>

#include <ctype.h>

int result=0;

int *r =&result;

struct dictionary

{

char word [20];

char meaning [20];

};

int main(){

int sd, sd2, nsd, clilen, sport, val ,pos=0 ;

char resultStr[100], search [100];

struct dictionary dict[10];

strcpy(dict[0].word, "Happy");

strcpy(dict[0].meaning, "Cheerful");

strcpy(dict[1].word, "Approach");

strcpy(dict[1].meaning, "Move towards");

strcpy(dict[2].word, "Establish");

strcpy(dict[2].meaning, "Set up or found");

strcpy(dict[3].word, "Utter");

strcpy(dict[3].meaning, "Without qualification");

strcpy(dict[4].word, "Conduct");
```

```
strcpy(dict[4].meaning, "Manage or Control" );
strcpy(dict[5].word, "Engage");
strcpy(dict[5].meaning, "Consume all of one's attention or time");
strcpy(dict[6].word, "Obtain");
strcpy(dict[6].meaning, "Come into possession of");
struct sockaddr_in servaddr, cliaddr;
printf("Enter the server port: \n");
scanf("%d", &sport);
sd = socket(AF_INET, SOCK_STREAM, 0);
if (sd < 0)
printf("Can't create \n ");
else
printf("Socket is created \n");
servaddr.sin_family= AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(sport);
servaddr.sin_port = htons(sport);
sd2 = bind(sd, (struct sockaddr *)&servaddr, sizeof(servaddr));
if (sd2 < 0)
printf("Can't bind\n");
else printf(" Binded \n");
listen (sd, 5);
clilen = sizeof(cliaddr);
nsd = accept(sd, (struct sockaddr *)&cliaddr, &clilen);
if (nsd < 0)
printf("Can't accept\n");
else
printf("Accepted\n");
recv(nsd, search, 100, 0);
```

```

printf("Word received from server %s", search);

for (int i=0; i<10; i++)
{
if (strcmp(dict[i].word, search)==0)
{
*r=1;

strcpy(resultStr,dict[i].meaning);
}
}

if (*r==0)
{

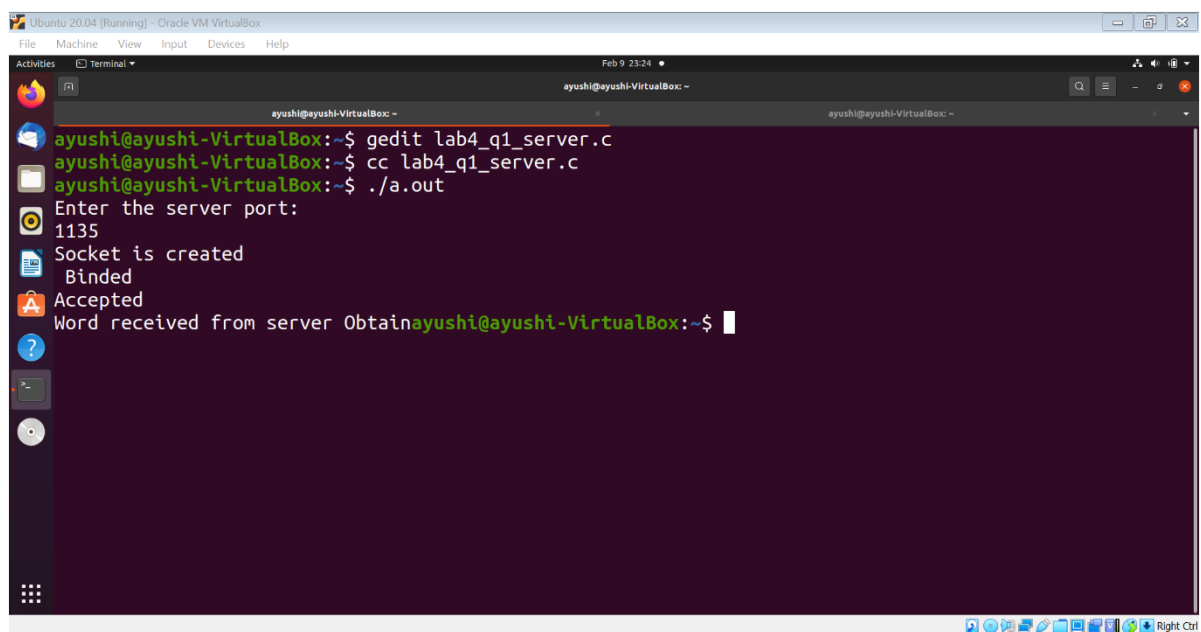
strcpy(resultStr,"No meaning found");
}

send(nsd, resultStr,100,0);

return 0;
}

```

## OUTPUT:



```

ayushi@ayushi-VirtualBox:~$ gedit lab4_q1_server.c
ayushi@ayushi-VirtualBox:~$ cc lab4_q1_server.c
ayushi@ayushi-VirtualBox:~$ ./a.out
Enter the server port:
1135
Socket is created
Binded
Accepted
Word received from server Obtainayushi@ayushi-VirtualBox:~$

```

# CLIENT:

## ALGORITHM:

1. Reading client **port** number
2. Creating socket using **socket()** function
3. Sending socket id and client address to server using **connect()** function  
[in the server this request is accepted using accept function]
4. Now client takes input from user for the word they want to search and storing it an array.
5. Call **send()** function with search as argument.
6. Receiving the message from server using **rev()** function.
7. Displaying the message.
8. Closing client socket using **close()** function.

## CODE:

```
#include <stdio.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <stdlib.h>

#include <arpa/inet.h>


int main()

{

int csd, lenstr, val, cport, res;

char search [100], result[100];

struct sockaddr_in servaddr;

printf("Enter the word you want to search: \n");

fgets(search, 100, stdin);

strtok(search, "\n");
```

```

printf("Enter the port: \n");

scanf("%d", &cport);

csd = socket(AF_INET, SOCK_STREAM, 0);

if (csd < 0)

printf("Can't create\n");

else

printf("Socket is created\n");

servaddr.sin_family= AF_INET;

servaddr.sin_addr.s_addr = htonl(INADDR_ANY);

servaddr.sin_port = htons (cport);

if (connect(csd, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0)

printf("Can't connect\n");

else

printf("Connected sucessfully\n");

send (csd, search, 100, 0);

recv(csd, result, 100, 0);

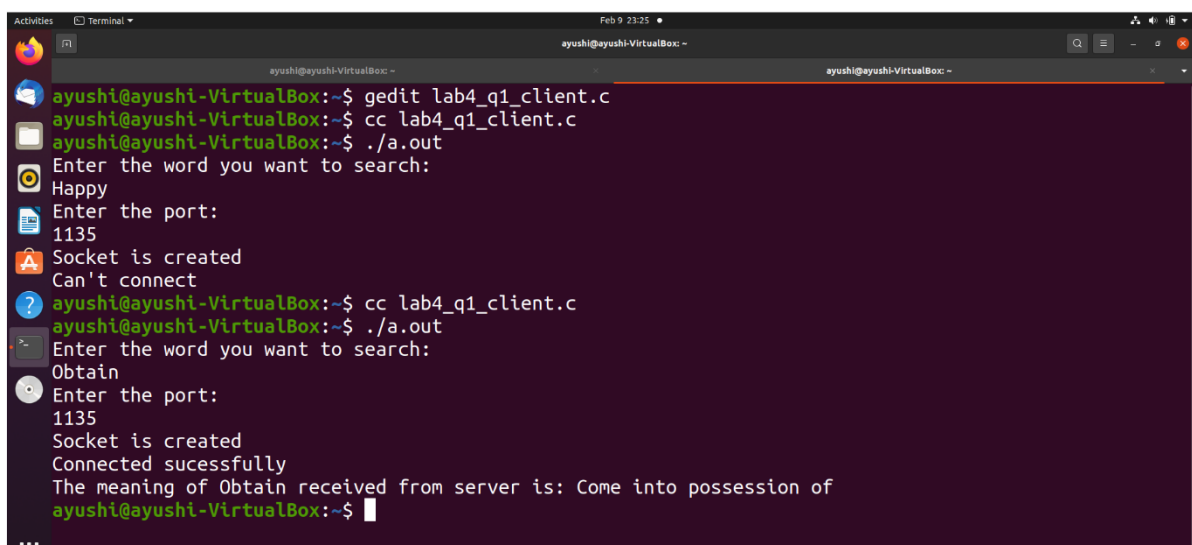
printf("The meaning of %s received from server is: %s\n", search, result);

return 0;

}

```

## OUTPUT:



```

ayushi@ayushi-VirtualBox:~$ gedit lab4_q1_client.c
ayushi@ayushi-VirtualBox:~$ cc lab4_q1_client.c
ayushi@ayushi-VirtualBox:~$ ./a.out
Enter the word you want to search:
Happy
Enter the port:
1135
Socket is created
Can't connect
ayushi@ayushi-VirtualBox:~$ cc lab4_q1_client.c
ayushi@ayushi-VirtualBox:~$ ./a.out
Enter the word you want to search:
Obtain
Enter the port:
1135
Socket is created
Connected sucessfully
The meaning of Obtain received from server is: Come into possession of
ayushi@ayushi-VirtualBox:~$

```

2. Develop a "Remote Calculator" application that works as follows: The client program inputs two integers and an arithmetic operation (\*, %, +, -) from the user and sends these three values to the server side. The server does the given operation on the two integers and sends back the result of the operation to the client. Help server to implement this scenario using connection-oriented sockets.

## SERVER:

### ALGORITHM:

14. Reading server **port** number, declare an string for storing time
15. Creating a socket using **socket()** function
16. Binding the socket to the server address using the **bind()** function
17. Reading max number of clients for which server has to wait until that many requests are requested from the client.
18. Keep the server in passive mode by using **listen()** function, in which the server has to wait until there is a request from the client.
19. Connection is established between server and client using **accept()** function and then now both are ready to send and receive data.
20. Using **recv()** function to receive data from the client and store it in the buffer.
21. Evaluating the operation and store the ans in a string 'sendmsg'.
22. Sending the 'sendmsg' to client.
23. Closing server socket using **close()** function.

### CODE:

```
#include<stdio.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<string.h>

#include<stdlib.h>

int main()

{
```

```

int socketid, sd, nsd, clilen, s_port;

char sendmsg[100], operand1[10], operand2[10], operatorr[2];

struct sockaddr_in servaddr, cliaddr;

printf("Enter the server port: \n");

scanf("%d", &s_port);

printf("Port number is %d", s_port);

socketid = socket(AF_INET, SOCK_STREAM, 0);

if(socketid < 0) printf("\nCan't create. \n");

else printf("\nSocket is created \n");

servaddr.sin_family = AF_INET;

servaddr.sin_addr.s_addr = htonl(INADDR_ANY);

servaddr.sin_port = htons(s_port);

sd = bind(socketid, (struct sockaddr *) &servaddr, sizeof(servaddr));

if(sd < 0) printf("Can't bind. \n");

else printf("Binded \n");

listen(socketid, 5);

clilen = sizeof(cliaddr);

nsd = accept(socketid, (struct sockaddr *) &cliaddr, &clilen);

if(nsd < 0) printf("Can't accept. \n");

else{

    printf("Accepted \n");

    recv(nsd, operand1, 10, 0);

    recv(nsd, operand2, 10, 0);

    recv(nsd, operatorr, 1, 0);

    int O1 = atoi(operand1);

    int O2 = atoi(operand2);

    int x;

    switch(operatorr[0])

    {

```



```

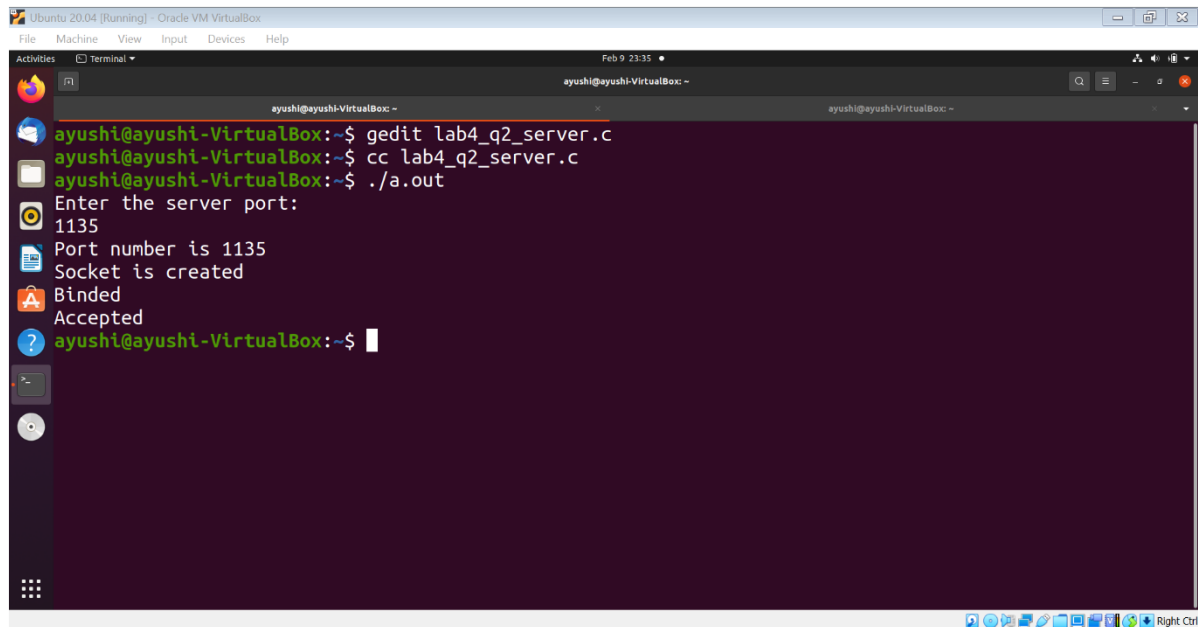
        case '+':
            x = O1+O2;
            break;
        case '-':
            x = O1-O2;
            break;
        case '*':
            x = O1*O2;
            break;
        case '/':
            x = O1/O2;
            break;
        case '%':
            x = O1%O2;
            break;
    }

    int res = x, count=0;
    while(res>0)
    {
        res = res/10;
        count++;
    }
    sendmsg[count]='\0';
    for(int i=count-1;i>=0;i--)
    {
        sendmsg[i]=x%10+'0';
        x=x/10;
    }
    send(nsd,sendmsg,100,0);

```

```
}  
  
return 1;  
  
}
```

## OUTPUT:



```
ayushi@ayushi-VirtualBox:~$ gedit lab4_q2_server.c  
ayushi@ayushi-VirtualBox:~$ cc lab4_q2_server.c  
ayushi@ayushi-VirtualBox:~$ ./a.out  
Enter the server port:  
1135  
Port number is 1135  
Socket is created  
Binded  
Accepted  
ayushi@ayushi-VirtualBox:~$
```

## CLIENT:

### ALGORITHM:

9. Reading client **port** number
10. Creating socket using **socket()** function
11. Sending socket id and client address to server using **connect()** function  
[in the server this request is accepted using accept function]
12. Now client takes input from user for operands and operator and storing it a character array.
13. Sending the above stored character array to the server using **send()** function.
14. Receiving the evaluated output from server using **recv()** function.
15. Displaying the output.
16. Closing client socket using **close()** function.

## **CODE:**

```
#include<stdio.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<string.h>

int main()
{
    int csd,c_port,len;
    char revmsg[100],operand1[10],operand2[10],operatorr[2];
    struct sockaddr_in servaddr;
    printf("Enter the port: \n");
    scanf("%d",&c_port);
    printf("Port number is %d",c_port);
    csd = socket(AF_INET,SOCK_STREAM,0);
    if(csd<0) printf("\nCan't create\n");
    else printf("\nSocket is created\n");
    servaddr.sin_family=AF_INET;
    servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
    servaddr.sin_port=htons(c_port);
    if(connect(csd,(struct sockaddr*)&servaddr,sizeof(servaddr))<0)
    {
        printf("Can't connect\n");
    }
    else
    {
        printf("Connected\n");
        printf("Enter the first operand: ");
        scanf("%s",operand1);
```

```

        printf("Enter the second operand: ");

        scanf("%s",operand2);

        send(csd,operand1,10,0);

        send(csd,operand2,10,0);

        printf("Enter the operator: ");

        scanf("%s",operatorr);

        send(csd,operatorr,2,0);

        recv(csd,revmsg,20,0);

        printf("%s %s %s = %s\n",operand1,operatorr,operand2,revmsg);

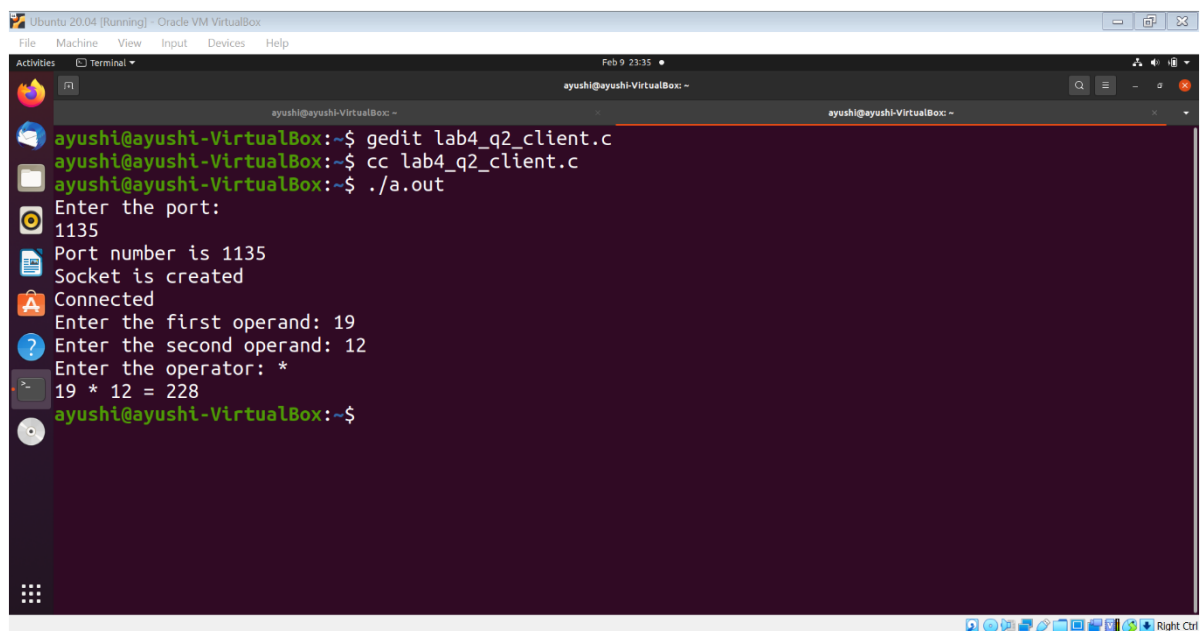
    }

    return 1;

}

```

## OUTPUT:



```

ayushi@ayushi-VirtualBox:~$ gedit lab4_q2_client.c
ayushi@ayushi-VirtualBox:~$ cc lab4_q2_client.c
ayushi@ayushi-VirtualBox:~$ ./a.out
Enter the port:
1135
Port number is 1135
Socket is created
Connected
Enter the first operand: 19
Enter the second operand: 12
Enter the operator: *
19 * 12 = 228
ayushi@ayushi-VirtualBox:~$

```