

Computer Networks Lab

**ASSIGNMENT**



20F-0336

5C

Ahmed Kasteer

# **Problem 1.**

# **Client:**

#include <netdb.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#define MAX 80

#define PORT 8080

#define SA struct sockaddr

void socketFunction(int socketVar)

{

int var;

FILE \* file;

file = fopen("clientID.txt", "fileReadObj");

if (file!=NULL)

{

printf("Client IDs are assigned.\n");

}

else

{

printf("Client IDs are missing.\n");

}

fscanf(file,"%d",&var);

fclose(file);

file=fopen("clientID.txt","fileWriteObj");

char buff[MAX];

bzero(buff, sizeof(buff));

const char\* name = "Hello i am client My Id is:";

strcpy(buff,name);

printf(buff);

var=var+1;

fprintf(file,"%d",var);

printf("n=%d\n",var);

write(socketVar,&var, sizeof(buff));

}

int main()

{

int socketVar;

struct sockaddr\_in servaddr;

// socket create and varification

socketVar = socket(AF\_INET, SOCK\_STREAM, 0);

if (socketVar == -1) {

printf("socket creation failed...\n");

exit(0);

}

else

printf("Socket successfully created..\n");

bzero(&servaddr, sizeof(servaddr));

// assign IP, PORT

servaddr.sin\_family = AF\_INET;

servaddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

servaddr.sin\_port = htons(PORT);

// connect the client socket to server socket

if (connect(socketVar, (SA\*)&servaddr, sizeof(servaddr)) != 0) {

printf("connection with the server failed...\n");

exit(0);

}

else

printf("connected to the server..\n");

socketFunction(socketVar);

// close the socket

close(socketVar);

return 0;

}

# **Server:**

#include <stdio.h>

#include <netdb.h>

#include <netinet/in.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#define MAX 80

#define PORT 8080

#define SA struct sockaddr

int temp=0;

void func(int socketFunction)

{

char buff[MAX];

int num;

int n;

// infinite loop for chat

bzero(buff, MAX);

// read the message from client and copy it in buffer

read(socketFunction,&num, sizeof(num));

// print buffer which contains the client contents

temp=temp+1;

printf("Hello i am Server Your recieved Id is: %d\t\n: ",temp);

bzero(buff, MAX);

}

// Driver function

int main()

{

int socketFunction, connfd, len;

struct sockaddr\_in servaddr, cli;

// socket create and verification

socketFunction = socket(AF\_INET, SOCK\_STREAM, 0);

if (socketFunction == -1) {

printf("socket creation failed...\n");

exit(0);

}

else

printf("Socket successfully created..\n");

bzero(&servaddr, sizeof(servaddr));

// assign IP, PORT

servaddr.sin\_family = AF\_INET;

servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

servaddr.sin\_port = htons(PORT);

// Binding newly created socket to given IP and verification

if ((bind(socketFunction, (SA\*)&servaddr, sizeof(servaddr))) != 0) {

printf("socket bind failed...\n");

exit(0);

}

else

printf("Socket successfully binded..\n");

// Now server is ready to listen and verification

if ((listen(socketFunction, 5)) != 0) {

printf("Listen failed...\n");

exit(0);

}

else

printf("Server listening..\n");

len = sizeof(cli);

while(1)

{

// Accept the data packet from client and verification

connfd = accept(socketFunction, (SA\*)&cli, &len);

if (connfd < 0)

{

printf("server acccept failed...\n");

exit(0);

}

else

{

printf("server acccept the client...\n");

}

func(connfd);

}

// After chatting close the socket

close(socketFunction);

}

# **Output:**

Text

Description automatically generated

Text

Description automatically generated

# **Problem 2.**

# **Client:**

# #include <netinet/in.h>

# #include <stdio.h>

# #include <stdlib.h>

# #include <string.h>

# #include <sys/socket.h>

# #include <unistd.h>

# #define PORT 8080

# int main()

# {

# int server\_fd, new\_socket, var1;

# struct sockaddr\_in address;

# char arrString[100];

# int addrlen = sizeof(address);

# char buffer[1024] = { 0 };

# 

# if ((server\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) == 0)

# {

# perror("Socket Connection Failure.\n");

# exit(EXIT\_FAILURE);

# }

# address.sin\_family = AF\_INET;

# address.sin\_addr.s\_addr = INADDR\_ANY;

# address.sin\_port = htons(PORT);

# if (bind(server\_fd, (struct sockaddr\*)&address, sizeof(address)) < 0)

# {

# perror("Socket Binding Failed.\n");

# exit(EXIT\_FAILURE);

# }

# if (listen(server\_fd, 3) < 0)

# {

# perror(" Listen");

# exit(EXIT\_FAILURE);

# }

# if ((new\_socket = accept(server\_fd,(struct sockaddr\*)&address,(socklen\_t\*)&addrlen)) < 0)

# {

# perror(" Accept ");

# exit(EXIT\_FAILURE);

# }

# var1 = read(new\_socket, arrString,sizeof(arrString));

# int i, j, temp;

# int l = strlen(arrString);

# printf("\nString Input by Client: %s\n", arrString);

# for (i = 0, j = l - 1; i < j; i++, j--)

# {

# temp = arrString[i];

# arrString[i] = arrString[j];

# arrString[j] = temp;

# }

# send(new\_socket, arrString, sizeof(arrString), 0);

# return 0;

# }

# **Server:**

# #include <arpa/inet.h>

# #include <netinet/in.h>

# #include <stdio.h>

# #include <stdlib.h>

# #include <string.h>

# #include <sys/socket.h>

# #include <unistd.h>

# #define PORT 8080

# 

# int main()

# {

# struct sockaddr\_in address;

# int sock = 0, var1;

# struct sockaddr\_in serv\_addr;

# char arrString[100];

# 

# printf("Input String: ");

# scanf("%[^\n]s", arrString);

# char buffer[1024] = { 0 };

# if ((sock = socket(AF\_INET,SOCK\_STREAM, 0))< 0)

# {

# printf("\nSocket Unsucessful.\n");

# return -1;

# }

# memset(&serv\_addr, '0', sizeof(serv\_addr));

# serv\_addr.sin\_family = AF\_INET;

# serv\_addr.sin\_port = htons(PORT);

# if (inet\_pton(AF\_INET, "127.0.0.1",&serv\_addr.sin\_addr) <= 0)

# {

# printf("\nNo Address Found.\n");

# return -1;

# }

# if (connect(sock, (struct sockaddr\*)&serv\_addr,sizeof(serv\_addr))< 0)

# {

# printf("\n Connection Failure.\n");

# return -1;

# }

# int l = strlen(arrString);

# send(sock, arrString, sizeof(arrString), 0);

# var1 = read(sock, arrString, l);

# printf("\nInverted String is: %s\n", arrString);

# return 0;

# }

# **Problem 3.**

# **Client:**

#include <netdb.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#define MAX 80

#define PORT 8080

#define SA struct sockaddr

int main()

{

int server\_fd, sockfd, var1;

struct sockaddr\_in address;

char arrString[80];

int addrlen = sizeof(address);

char buffer[1024] = { 0 };

if ((server\_fd = socket(AF\_INET, SOCK\_DGRAM, 0)) == 0)

{

perror("Socket Connection Failure.\n");

exit(EXIT\_FAILURE);

}

address.sin\_family = AF\_INET;

address.sin\_addr.s\_addr = INADDR\_ANY;

address.sin\_port = htons(PORT);

if (bind(server\_fd, (struct sockaddr\*)&address, sizeof(address)) < 0)

{

perror("Socket Binding Failed.\n");

exit(EXIT\_FAILURE);

}

var1 = read(sockfd, arrString,sizeof(arrString));

int i, j, temp;

int l = strlen(arrString);

printf("\nString Input by Client: %s\n", arrString);

for (i = 0, j = l - 1; i < j; i++, j--)

temp = arrString[i];

arrString[i] = arrString[j];

arrString[j] = temp;

send(sockfd, arrString, sizeof(arrString), 0);

return 0;

}

# **Server:**

#include <stdio.h>

#include <netdb.h>

#include <netinet/in.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#define MAX 80

#define PORT 8080

#define SA struct sockaddr

int main()

{

struct sockaddr\_in address;

int sock = 0, var1;

struct sockaddr\_in serv\_addr;

char arrString[100];

printf("Input String: ");

scanf("%[^\n]s", arrString);

char buffer[1024] = { 0 };

if ((sock = socket(AF\_INET,SOCK\_DGRAM, 0))< 0)

{

printf("\nSocket Unsucessful.\n");

return -1;

}

memset(&serv\_addr, '0', sizeof(serv\_addr));

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(PORT);

if (inet\_pton(AF\_INET, "127.0.0.1",&serv\_addr.sin\_addr) <= 0)

{

printf("\nNo Address Found.\n");

return -1;

}

if (connect(sock, (struct sockaddr\*)&serv\_addr,sizeof(serv\_addr))< 0)

{

printf("\n Connection Failure.\n");

return -1;

}

int l = strlen(arrString);

send(sock, arrString, sizeof(arrString), 0);

var1 = read(sock, arrString, l);

printf("\nInverted String is: %s\n", arrString);

return 0;

}