**CSE1004 – Network and Communication Lab [L52+L53]**

**Arvind CB 19BCE1221**

**Faculty:** Dr Kanchana Devi V

**CLIENT CODE:**

#include <stdlib.h>

#include <stdio.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <arpa/inet.h>

#define MAXLINE 4096 /\*max text line length\*/

#define SERV\_PORT 3000 /\*port\*/

int

main(int argc, char \*\*argv)

{

int sockfd;

struct sockaddr\_in servaddr;

char sendline[MAXLINE], recvline[MAXLINE];

//basic check of the arguments

//additional checks can be inserted

if (argc !=2) {

perror("Usage: TCPClient <IP address of the server");

exit(1);

}

//Create a socket for the client

//If sockfd<0 there was an error in the creation of the socket

if ((sockfd = socket (AF\_INET, SOCK\_STREAM, 0)) <0) {

perror("Problem in creating the socket");

exit(2);

}

//Creation of the socket

memset(&servaddr, 0, sizeof(servaddr));

servaddr.sin\_family = AF\_INET;

servaddr.sin\_addr.s\_addr= inet\_addr(argv[1]);

servaddr.sin\_port = htons(SERV\_PORT); //convert to big-endian order

//Connection of the client to the socket

if (connect(sockfd, (struct sockaddr \*) &servaddr, sizeof(servaddr))<0) {

perror("Problem in connecting to the server");

exit(3);

}

while (fgets(sendline, MAXLINE, stdin) != NULL) {

ticks = time(NULL);

snprintf(sendline, sizeof(sendline), "%.24s\r\n", ctime(&ticks));

write(connfd, sendline, strlen(sendline));

send(sockfd, sendline, strlen(sendline), 0);

if (recv(sockfd, recvline, MAXLINE,0) == 0){

//error: server terminated prematurely

perror("The server terminated prematurely");

exit(4);

}

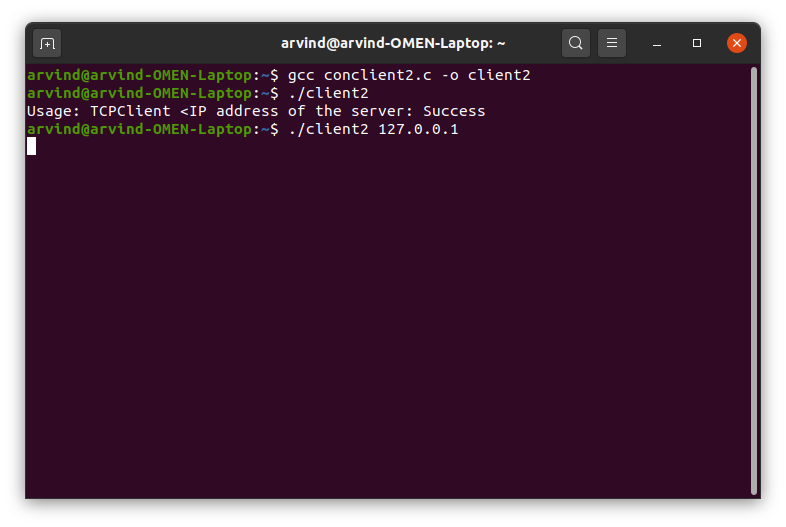
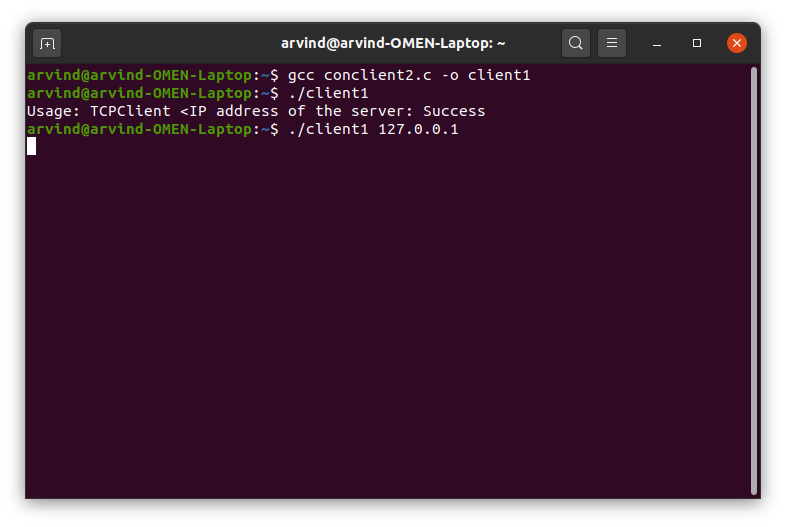
printf("%s", "String received from the server: ");

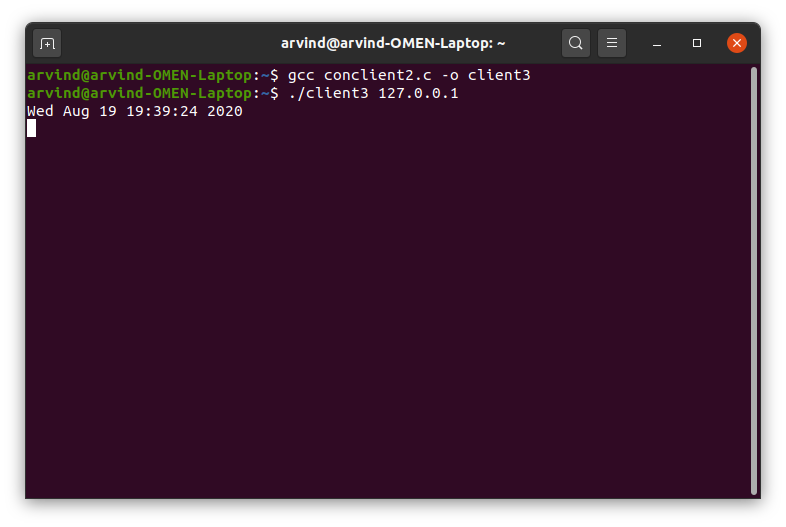
fputs(recvline, stdout);

}

exit(0);

}





**SERVER CODE:**

#include <stdlib.h>

#include <stdio.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <unistd.h>

#define MAXLINE 4096 /\*max text line length\*/

#define SERV\_PORT 3000 /\*port\*/

#define LISTENQ 8 /\*maximum number of client connections\*/

int main (int argc, char \*\*argv)

{

int listenfd, connfd, n;

pid\_t childpid;

socklen\_t clilen;

char buf[MAXLINE];

struct sockaddr\_in cliaddr, servaddr;

//Create a socket for the soclet

//If sockfd<0 there was an error in the creation of the socket

if ((listenfd = socket (AF\_INET, SOCK\_STREAM, 0)) <0) {

perror("Problem in creating the socket");

exit(2);

}

//preparation of the socket address

servaddr.sin\_family = AF\_INET;

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

servaddr.sin\_port = htons(SERV\_PORT);

//bind the socket

bind (listenfd, (struct sockaddr \*) &servaddr, sizeof(servaddr));

//listen to the socket by creating a connection queue, then wait for clients

listen (listenfd, LISTENQ);

printf("%s\n","Server running...waiting for connections.");

for ( ; ; ) {

clilen = sizeof(cliaddr);

//accept a connection

connfd = accept (listenfd, (struct sockaddr \*) &cliaddr, &clilen);

printf("%s\n","Received request...");

if ( (childpid = fork ()) == 0 ) {//if it’s 0, it’s child process

printf ("%s\n","Child created for dealing with client requests");

//close listening socket

close (listenfd);

while ( (n = recv(connfd, buf, MAXLINE,0)) > 0) {

printf("%s","String received from and resent to the client:");

puts(buf);

send(connfd, buf, n, 0);

}

if (n < 0)

printf("%s\n", "Read error");

exit(0);

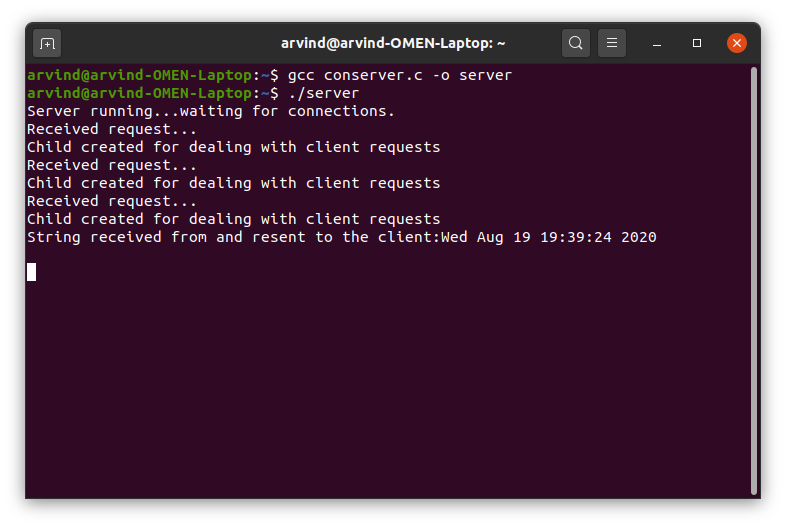
}

//close socket of the server

close(connfd);

}

}



**I am unable to get the date and time correctly but I feel the code is right and it is error free**