

**Check Current Time**

|  |  |
| --- | --- |
| Name: | Wilson Vidyut Doloy |
| Reg No: | 19BCE1603 |
| Subject: | CSE1004 Network and Communication Lab |
| Slot: | L52+L653 (Prof. Ganeshan) |
| Date: | 19 August 2020 |

Assume you are designing an Internet Server, in which you are supposed to test the server and **multiple** client machines. The task here is to check the current time of all the client machines for synchronisation purpose. Develop a socket program in C in order to perform the same.

**Hint: 1 Server Program and at least 3 Client Programs**

Input Format:

Server:

Listening....

From Client: Wed Aug 19 15:16:26 2020

From Client: Wed Aug 19 15:17:32 2020

From Client: Wed Aug 19 15:18:44 2020

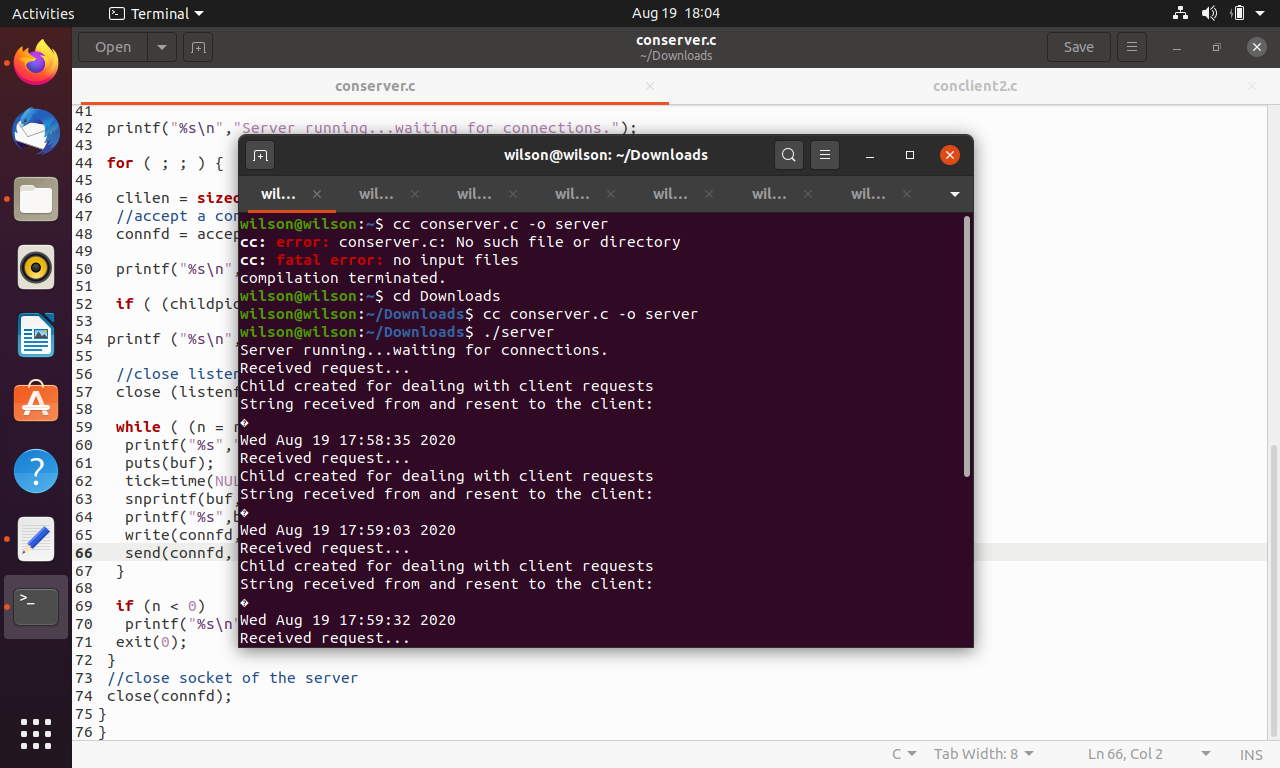
**Server Side Code:**

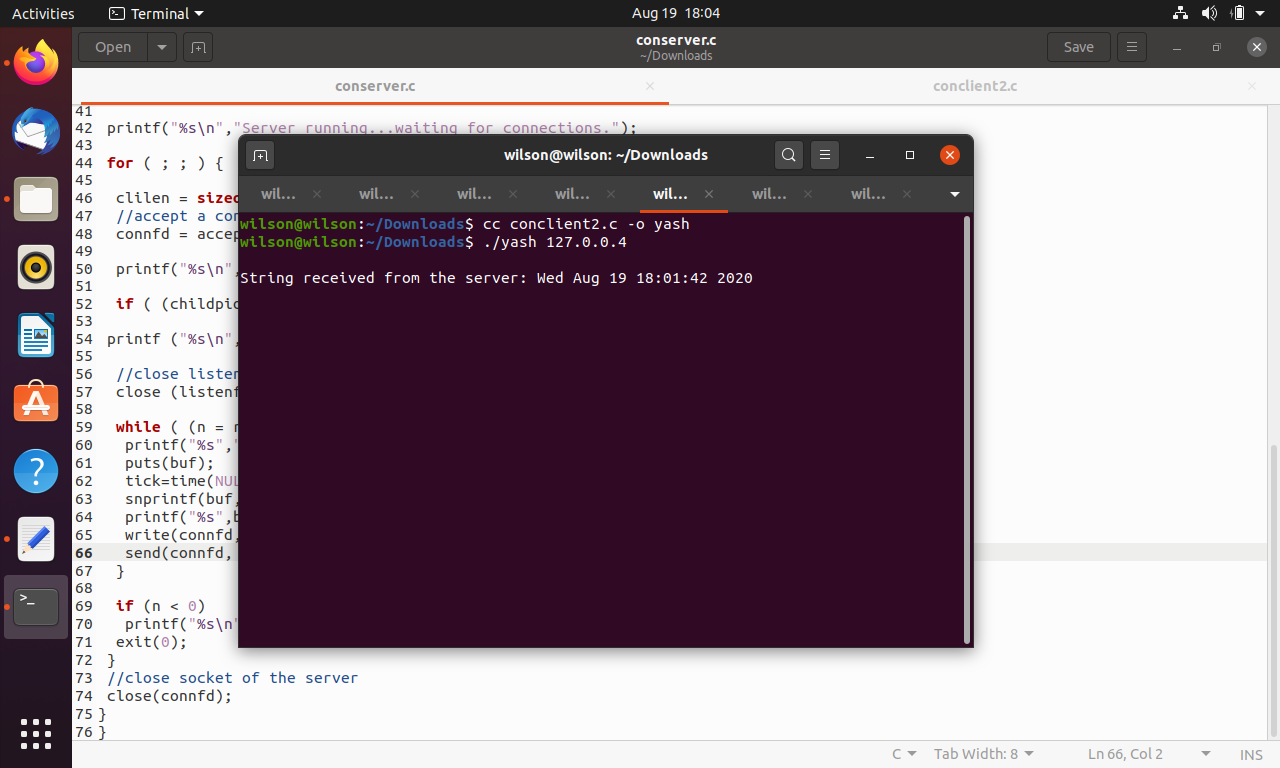
#include <stdlib.h>  
#include <stdio.h>  
#include <sys/types.h>  
#include <sys/socket.h>  
#include <netinet/in.h>  
#include <string.h>  
#include <unistd.h>  
#include <time.h>  
  
#define MAXLINE 4096   
#define SERV\_PORT 3000   
#define LISTENQ 8   
  
int main (int argc, char \*\*argv)  
{  
 int listenfd, connfd, n;  
 pid\_t childpid;  
 socklen\_t clilen;  
 char buf[MAXLINE];  
 struct sockaddr\_in cliaddr, servaddr;  
 time\_t tick;  
  
 if ((listenfd = socket (AF\_INET, SOCK\_STREAM, 0)) <0) {  
 perror("Problem in creating the socket");  
 exit(2);  
 }  
  
  
  
 servaddr.sin\_family = AF\_INET;  
 servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  
 servaddr.sin\_port = htons(SERV\_PORT);  
  
  
 bind (listenfd, (struct sockaddr \*) &servaddr, sizeof(servaddr));  
  
  
 listen (listenfd, LISTENQ);  
  
 printf("%s\n","Server running...waiting for connections.");  
  
 for ( ; ; ) {  
  
 clilen = sizeof(cliaddr);  
  
 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 (listenfd);  
  
 while ( (n = recv(connfd, buf, MAXLINE,0)) > 0) {  
 printf("%s","String received from and resent to the client:");  
 puts(buf);  
 tick=time(NULL);  
 snprintf(buf,sizeof(buf),"%s",ctime(&tick));  
 printf("%s",buf);  
 write(connfd,buf,100);  
 //send(connfd, buf, n, 0);  
 }  
  
 if (n < 0)  
 printf("%s\n", "Read error");  
 exit(0);  
 }  
  
 close(connfd);  
}  
}

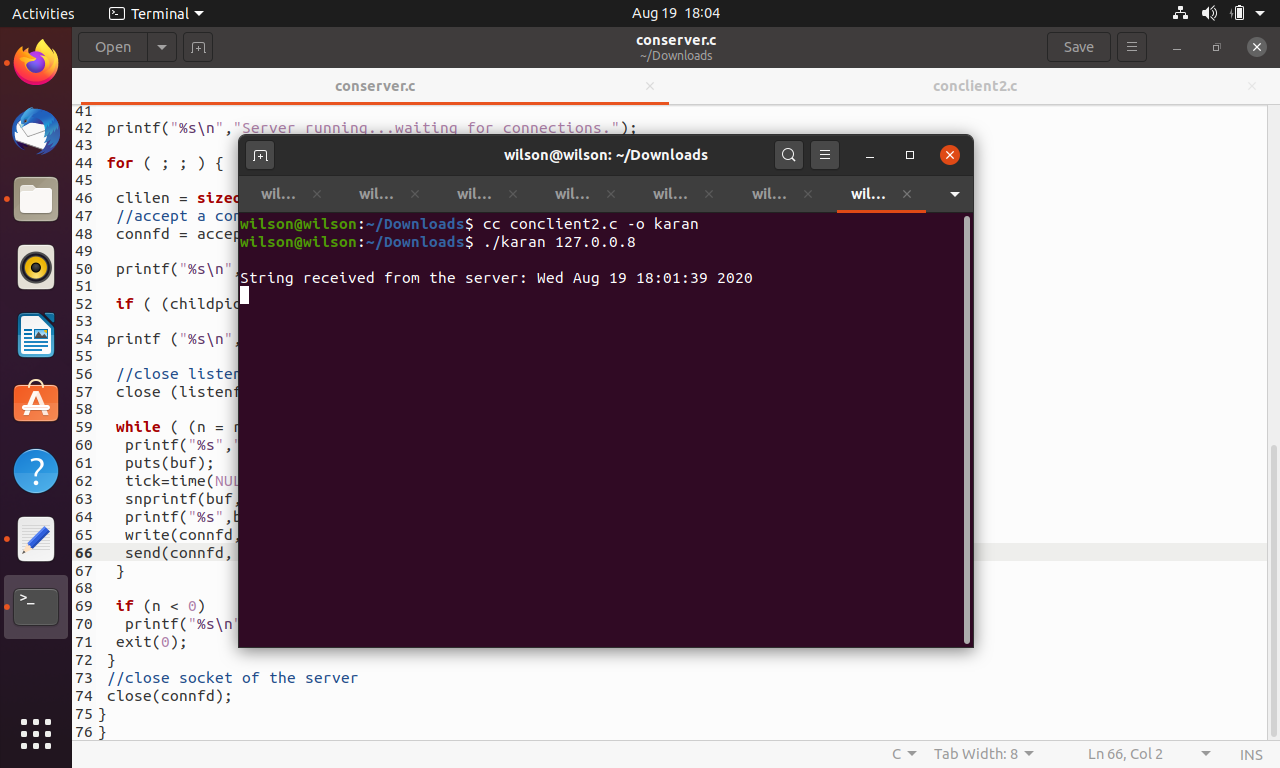
**Client Side Code:**

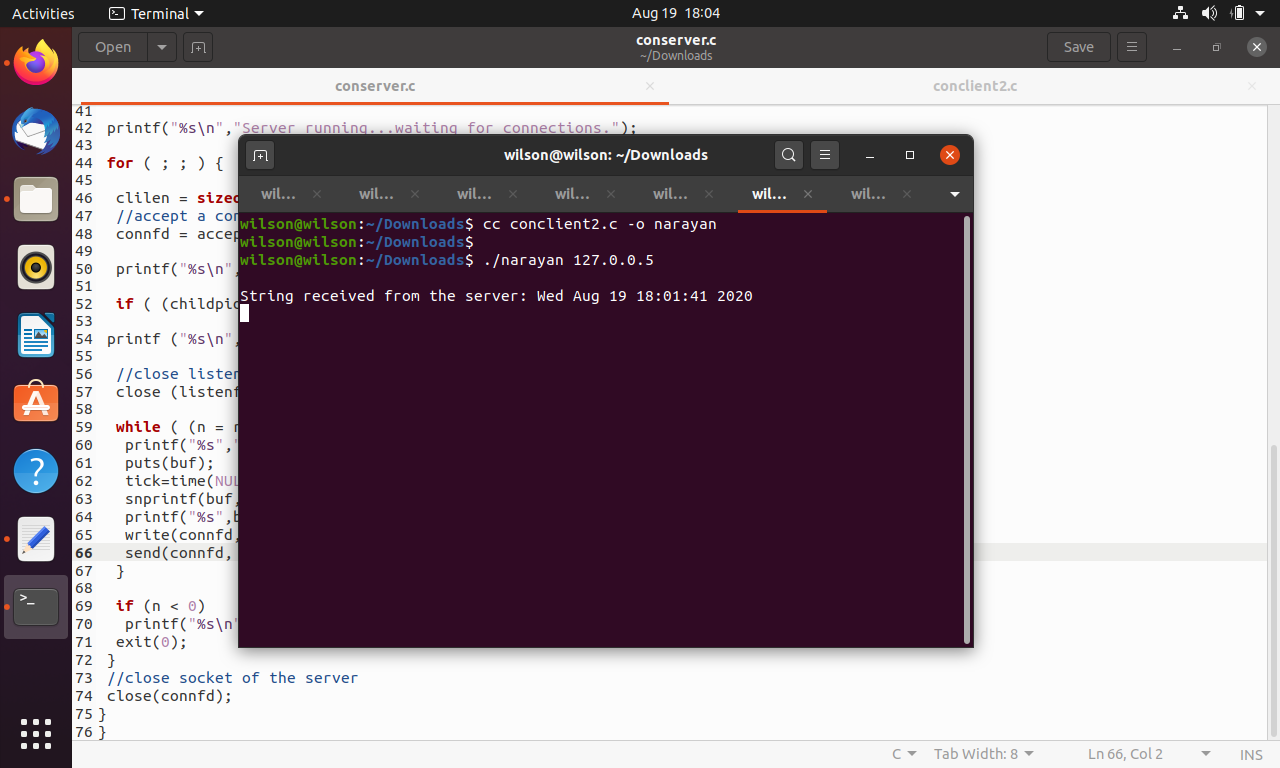
#include <stdlib.h>  
#include <stdio.h>  
#include <sys/types.h>  
#include <sys/socket.h>  
#include <netinet/in.h>  
#include <string.h>  
#include <arpa/inet.h>  
  
#define MAXLINE 4096 /max text line length/  
#define SERV\_PORT 3000 /port/  
  
int  
main(int argc, char \*\*argv)  
{  
 int sockfd;  
 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) {  
  
 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);  
}

**Output:**

****

****

****

****