**Exercise for 02.09.2020**

**NAC Lab**

**Dr. Kanchana Devi V & Dr. R. Ganesan**

**Name: Wilson Vidyut Doloy**

**Reg no: 19BCE1603**

1. The Temperature server which contains the temperature details of all the cities in India located at Mumbai. The current temperature for the city is returned by the Temperature Server to the requested Client. If the requested city cannot be found, an appropriate message is returned. Perform this task between two different systems using an **unreliable protocol**.

Eg: Client: Chennai Server: Chennai 40 degree

Server Code:

#include<stdio.h>

#include<netinet/in.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netdb.h>

#include<string.h>

#include<stdlib.h>

#define MAX 10

#define PORT 43454

#define SA struct sockaddr

void func(int sockfd)

{

char buff[MAX];

int i,clen,counter=0;

struct sockaddr\_in cli;

clen=sizeof(cli);

char city[10][10]={"delhi","chennai","jamshedpur","ranchi"};

int temp[5]={40,41,42,43};

bzero(buff,MAX);

recvfrom(sockfd,buff,sizeof(buff),0,(SA \*)&cli,&clen);

for(i=0;i<4;i++)

{

if(strcmp(city[i],buff)==0)

{

printf("%s is %d degree ",city[i],temp[i]);

}

else

counter++;

}

if(counter==4)

{

printf("No such data found");}

//label:

bzero(buff,MAX);

}

int main()

{

int sockfd;

struct sockaddr\_in servaddr;

sockfd=socket(AF\_INET,SOCK\_DGRAM,0);

if(sockfd==-1)

{

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

exit(0);

}

else

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

bzero(&servaddr,sizeof(servaddr));

servaddr.sin\_family=AF\_INET;

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

servaddr.sin\_port=htons(PORT);

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

{

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

exit(0);

}

else

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

func(sockfd);

close(sockfd);

}

Client Code:

#include<sys/socket.h>

#include<netdb.h>

#include<string.h>

#include<stdlib.h>

#include<stdio.h>

#define MAX 10

#define PORT 43454

#define SA struct sockaddr

int main()

{

char buff[MAX];

int sockfd,len,n;

struct sockaddr\_in servaddr;

sockfd=socket(AF\_INET,SOCK\_DGRAM,0);

if(sockfd==-1)

{

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

exit(0);

}

else

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

bzero(&servaddr,sizeof(len));

servaddr.sin\_family=AF\_INET;

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

servaddr.sin\_port=htons(PORT);

len=sizeof(servaddr);

printf("\nEnter city name : ");

n=0;

scanf("%s",buff);

//while((buff[n++]=getchar())!='\n');

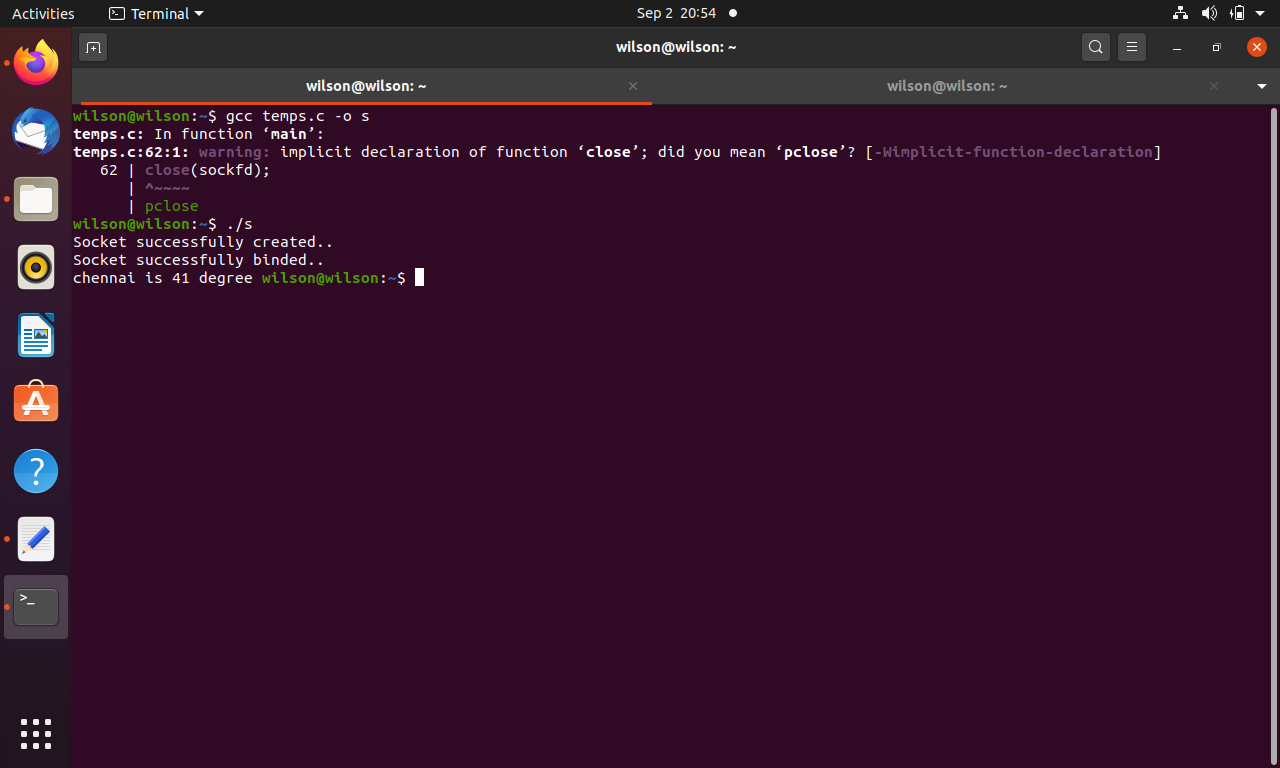
sendto(sockfd,buff,sizeof(buff),0,(SA \*)&servaddr,len);

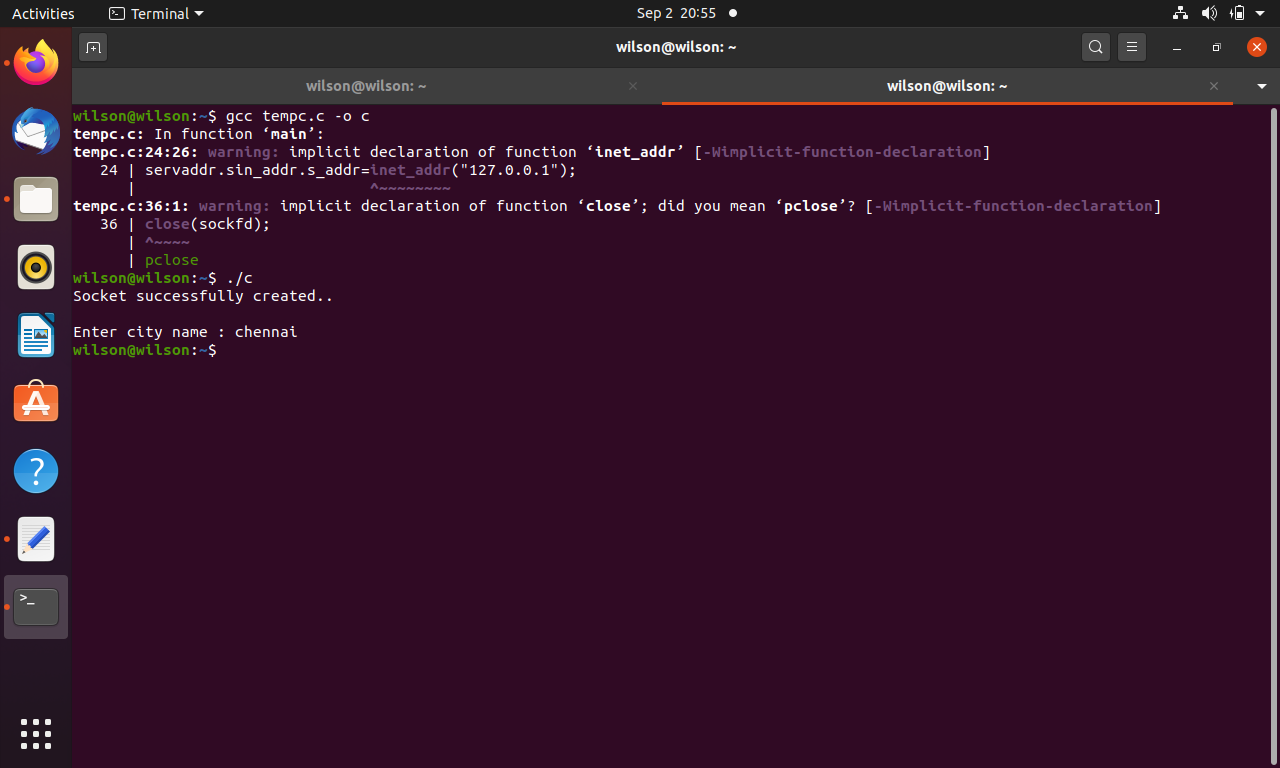
bzero(buff,sizeof(buff));

close(sockfd);

}

Output:





2. Establish an **unreliable connectionless** conversation between a client and server. The conversation terminates by receiving ‘BYE’ from any one of them. Generate a network encryption to protect the confidentiality of sensitive data in transmission. Use Caesar Cipher Technique for encryption:



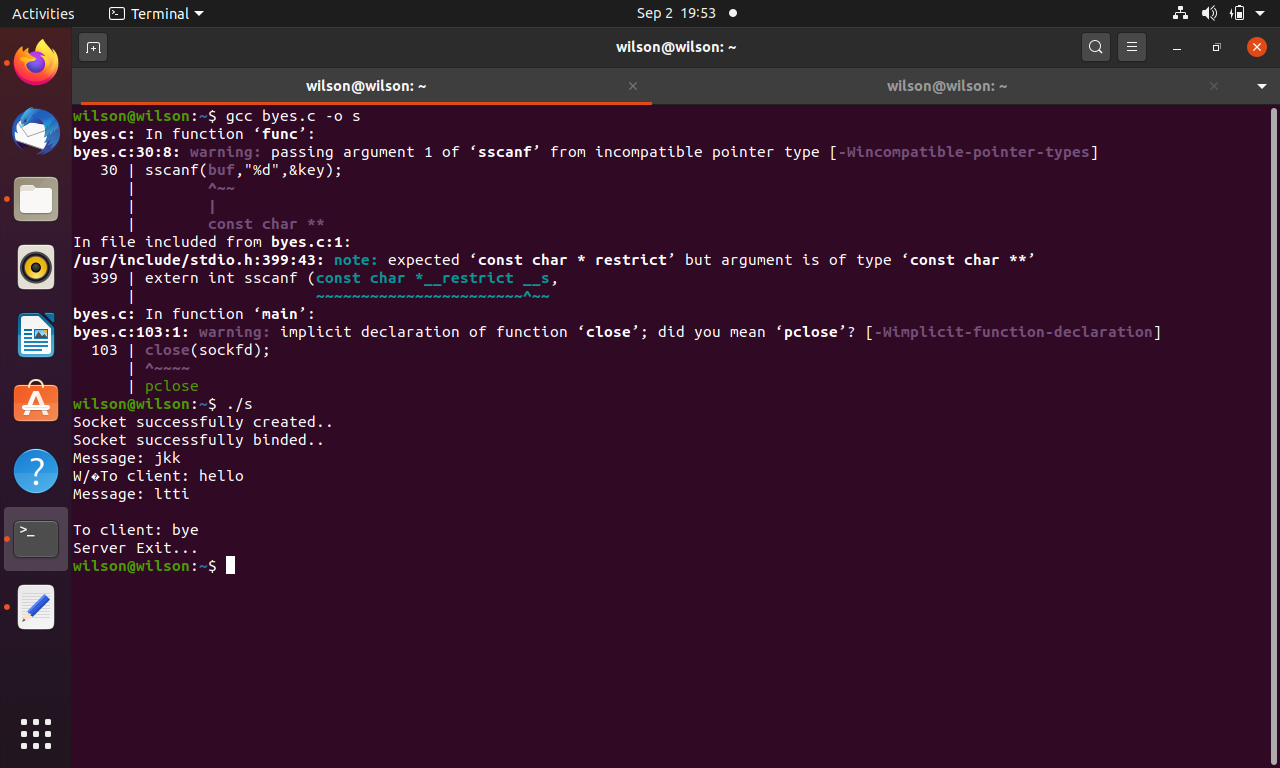
Server Code:

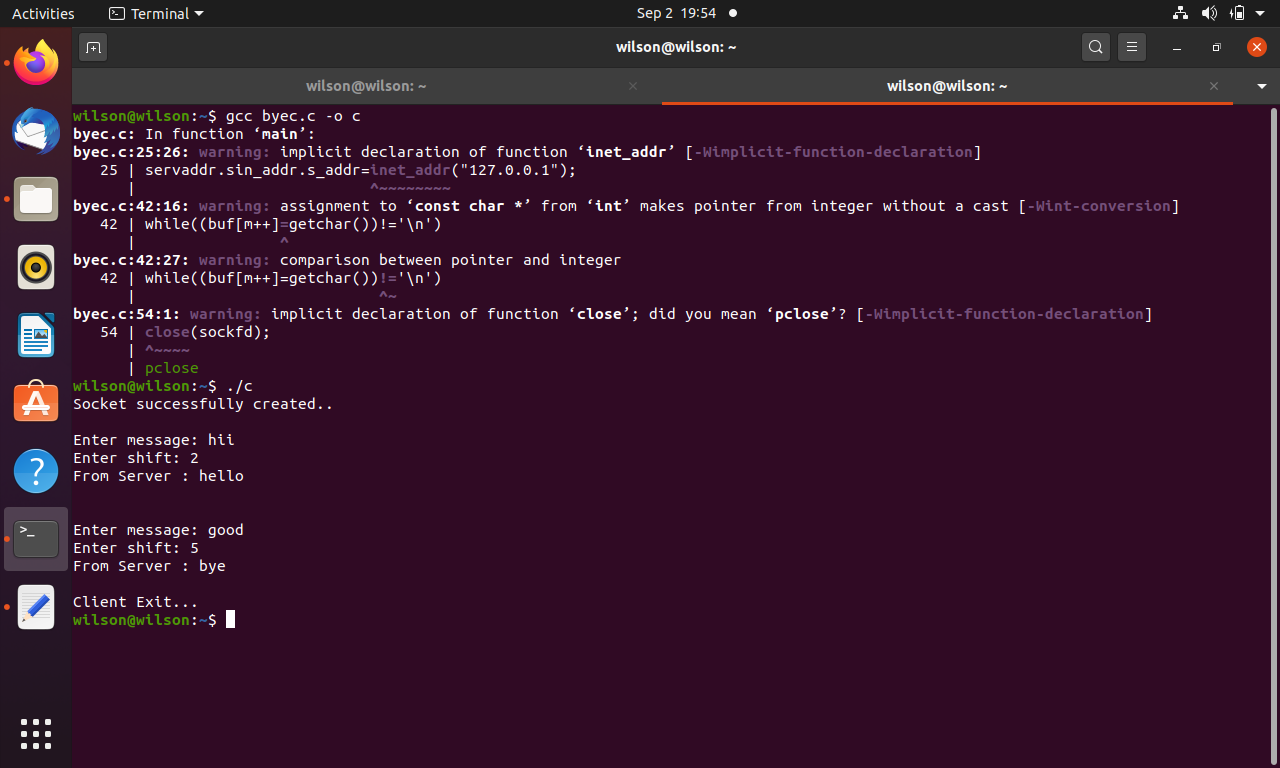
#include<stdio.h>  
#include<netinet/in.h>  
#include<sys/types.h>  
#include<sys/socket.h>  
#include<netdb.h>  
#include<string.h>  
#include<stdlib.h>  
#define MAX 10  
#define PORT 43454  
#define SA struct sockaddr  
void func(int sockfd)  
{  
char buff[MAX],ch;  
const char \*buf[2];  
int n,clen,i,key;  
struct sockaddr\_in cli;  
clen=sizeof(cli);  
for(;;)  
{  
bzero(buff,MAX);  
bzero(buf,2);  
recvfrom(sockfd,buff,sizeof(buff),0,(SA \*)&cli,&clen);  
if(strncmp("bye",buff,3)==0)  
{  
printf("Server Exit...\n");  
goto label;  
}  
  
recvfrom(sockfd,buf,sizeof(buf),0,(SA \*)&cli,&clen);  
sscanf(buf,"%d",&key);  
for(i = 0; buff[i] != '\0'; ++i)  
 {  
 ch = buff[i];  
 if(ch >= 'a' && ch <= 'z')  
 {  
 ch = ch + key;  
 if(ch > 'z')  
 {  
 ch = ch - 'z' + 'a' - 1;  
 }  
   
 buff[i] = ch;  
 }  
 else if(ch >= 'A' && ch <= 'Z')  
 {  
 ch = ch + key;  
   
 if(ch > 'Z')  
 {  
 ch = ch - 'Z' + 'A' - 1;  
 }  
   
 buff[i] = ch;  
 }  
 }  
 label: printf("Message: %s", buff);  
  
  
if(strncmp("bye",buff,3)==0)  
{  
printf("Server Exit...\n");  
break;  
}  
bzero(buff,MAX);  
n=0;  
printf("To client: ");  
while((buff[n++]=getchar())!='\n');  
sendto(sockfd,buff,sizeof(buff),0,(SA \*)&cli,clen);  
if(strncmp("bye",buff,3)==0)  
{  
printf("Server Exit...\n");  
break;  
}  
else{  
  
}  
}  
}  
int main()  
{  
int sockfd;  
struct sockaddr\_in servaddr;  
sockfd=socket(AF\_INET,SOCK\_DGRAM,0);  
if(sockfd==-1)  
{  
printf("socket creation failed...\n");  
exit(0);  
}  
else  
printf("Socket successfully created..\n");  
bzero(&servaddr,sizeof(servaddr));  
servaddr.sin\_family=AF\_INET;  
servaddr.sin\_addr.s\_addr=htonl(INADDR\_ANY);  
servaddr.sin\_port=htons(PORT);  
if((bind(sockfd,(SA \*)&servaddr,sizeof(servaddr)))!=0)  
{  
printf("socket bind failed...\n");  
exit(0);  
}  
else  
printf("Socket successfully binded..\n");  
func(sockfd);  
close(sockfd);  
}

Client Code:

#include<sys/socket.h>  
#include<netdb.h>  
#include<string.h>  
#include<stdlib.h>  
#include<stdio.h>  
#define MAX 10  
#define PORT 43454  
#define SA struct sockaddr  
int main()  
{  
char buff[MAX];  
const char \*buf[2];  
int sockfd,len,n,m;  
struct sockaddr\_in servaddr;  
sockfd=socket(AF\_INET,SOCK\_DGRAM,0);  
if(sockfd==-1)  
{  
printf("socket creation failed...\n");  
exit(0);  
}  
else  
printf("Socket successfully created..\n");  
bzero(&servaddr,sizeof(len));  
servaddr.sin\_family=AF\_INET;  
servaddr.sin\_addr.s\_addr=inet\_addr("127.0.0.1");  
servaddr.sin\_port=htons(PORT);  
len=sizeof(servaddr);  
for(;;)  
{  
printf("\nEnter message: ");  
n=0,m=0;  
while((buff[n++]=getchar())!='\n');  
sendto(sockfd,buff,sizeof(buff),0,(SA \*)&servaddr,len);  
  
if(strncmp("bye",buff,3)==0)  
{  
printf("Client Exit...\n");  
break;  
}  
else{  
printf("Enter shift: ");  
while((buf[m++]=getchar())!='\n')  
sendto(sockfd,buf,sizeof(buf),0,(SA \*)&servaddr,len);  
  
bzero(buff,sizeof(buff));  
recvfrom(sockfd,buff,sizeof(buff),0,(SA \*)&servaddr,&len);  
printf("From Server : %s\n",buff);  
if(strncmp("bye",buff,3)==0)  
{  
printf("Client Exit...\n");  
break;  
}}  
}  
close(sockfd);  
}

Output:

****

****