Ex.No:5

**CONCURRENT SERVER**

ROLL NO.:1905007

NAME : R.ASWINRAJA

Date:15.09.2021

**AIM:**

To implement a “Real Estate Management System” using simple Concurrent Server program.

**PROBLEM STATEMENT:**

Write a simple Concurrent Server program to implement “Real Estate Management System”.

* + - * The server maintains the different city and the land location available for sale.
      * The server also maintains the description about the land (land area, amount per cent, nearby facilities, etc.).
      * The client1/2/3 requests the server to give the details based on the city name/location.
      * If any two clients from requests for the same land, the request is processed based on the mode of payment (ready cash / loan processing).
      * The confirmation of land booking between the server and client at the final stage should be made encrypted.

**PROGRAM:**

**1. SERVER CODE:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#define MAX 1024

#define PORT 4444

struct lands

{ char property[3][5][30];

};

int main()

{ int sockfd, ret;

struct sockaddr\_in serverAddr;

int newSocket;

struct sockaddr\_in newAddr;

socklen\_t addr\_size;

char buffer[MAX],buffer1[MAX],str[MAX];

pid\_t childpid;

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

if(sockfd < 0){

printf("[-]Error in connection.\n");

exit(1);

}

printf("[+]Server Socket is created.\n");

memset(&serverAddr, '\0', sizeof(serverAddr));

serverAddr.sin\_family = AF\_INET;

serverAddr.sin\_port = htons(PORT);

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

ret = bind(sockfd, (struct sockaddr\*)&serverAddr, sizeof(serverAddr));

if(ret < 0){

printf("[-]Error in binding.\n");

exit(1);

}

printf("[+]Bind to port %d\n", 4444);

if(listen(sockfd, 10) == 0)

printf("[+]Listening....\n");

else printf("[-]Error in binding.\n");

key\_t key;

int shmid;

struct lands \*ptr;

//creation of shared memory

shmid = shmget(key, sizeof(struct lands), 0644 | IPC\_CREAT);

ptr = shmat(shmid, NULL, 0);

//initialize the properties details.

strcpy(ptr->property[0][0],"Salem");

strcpy(ptr->property[0][1],"Steel plant");

strcpy(ptr->property[0][2],"16");

strcpy(ptr->property[0][3],"300.5");

strcpy(ptr->property[0][4],"unbooked");

strcpy(ptr->property[1][0],"Coimbatore");

strcpy(ptr->property[1][1],"Gandhipuram");

strcpy(ptr->property[1][2],"12.5");

strcpy(ptr->property[1][3],"250.156");

strcpy(ptr->property[1][4],"unbooked");

strcpy(ptr->property[2][0],"Salem");

strcpy(ptr->property[2][1],"fairlands");

strcpy(ptr->property[2][2],"24");

strcpy(ptr->property[2][3],"150.86");

strcpy(ptr->property[2][4],"unbooked");

char reg[3][22]={"Registration:success\n","Registration:pending\n","Registration:failure\n"};

//encryption

for(int i=0;i<3;i++)

for(int j=0; (j<23 && reg[i][j]!='\0'); j++)

reg[i][j] = reg[i][j] + 5;

int num=3;//number of lands available

while(1)

{ newSocket = accept(sockfd, (struct sockaddr\*)&newAddr, &addr\_size);

if(newSocket < 0) exit(1);

printf("Connection accepted from %s:%d\n", inet\_ntoa(newAddr.sin\_addr), ntohs(newAddr.sin\_port));

int ind=2;

if((childpid = fork()) == 0)

{

close(sockfd);

ptr = shmat(shmid, NULL, 0);

while(1)

{ bzero(buffer1,MAX);

recv(newSocket, buffer1, MAX, 0);//receive city name

if(strcmp(buffer1, "exit") == 0){

printf("Disconnected from %s:%d\n", inet\_ntoa(newAddr.sin\_addr), ntohs(newAddr.sin\_port));

break;

}

else

{

int count=0;

for(int ind=0;ind<num;ind++)

if(strcmp(ptr->property[ind][0],buffer1)==0)

count++;

write(newSocket,&count,sizeof(count));

for(int index=0;index<num;index++)

{ if(strcmp(ptr->property[index][0],buffer1)==0)

{ printf("Client(%s:%d): %s\n",inet\_ntoa(newAddr.sin\_addr), ntohs(newAddr.sin\_port),buffer1);

for(int ind=0;ind<5;ind++)

{strcpy(str,ptr->property[index][ind]);

//encryption

for(int i = 0; (i < MAX && str[i] != '\0'); i++)

str[i] = str[i] + 5;

send(newSocket, str, strlen(str), 0);

bzero(buffer, sizeof(buffer));

recv(newSocket, buffer, MAX, 0);

strcpy(str,buffer);

//decryption

for(int i = 0; (i < MAX && str[i] != '\0'); i++)

str[i] = str[i] - 5;

}//for end

if(strcmp(str,"cash")==0 )

{

if(strcmp(ptr->property[index][4],"booked")==0)

send(newSocket, reg[2], strlen(reg[2]), 0);

else

{

strcpy(ptr->property[index][4],"booked");

send(newSocket, reg[0], strlen(reg[0]), 0);

printf("Property (%s - %s) registered by %s:%d\n", ptr->property[index][0],ptr->property[index][1],inet\_ntoa(newAddr.sin\_addr), ntohs(newAddr.sin\_port));

}

}

else if(strcmp(str,"loan")==0)

{

if(strcmp(ptr->property[index][4],"booked")==0)

send(newSocket, reg[2], strlen(reg[2]),0);

else

{ strcpy(ptr->property[index][4],"pending");

send(newSocket, reg[1], strlen(reg[1]),0);

printf("Property (%s - %s) registration pending by %s:%d\n", ptr->property[index][0],ptr->property[index][1],inet\_ntoa(newAddr.sin\_addr), ntohs(newAddr.sin\_port));

}

}

else

send(newSocket, reg[2], strlen(reg[2]), 0);

bzero(buffer,sizeof(buffer));

int choice;

read(newSocket,&choice,sizeof(choice));

if(choice==0)

break;

}//if end

}//for end

}//else emd

}// inner while(1) end

}//if(chid end

}//outer while(1) end

close(newSocket);

return 0;

}

**2. CLIENT CODE:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#define MAX 1024

#define PORT 4444

int main()

{ int clientSocket, ret;

struct sockaddr\_in serverAddr;

char buffer[MAX],str[MAX];

char payment[MAX];

char arr[5][16]={"city\t","location","land area","amount per cent","status"};

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

if(clientSocket < 0)

{ printf("[-]Error in connection.\n"); exit(1); }

printf("[+]Client Socket is created.\n");

memset(&serverAddr, '\0', sizeof(serverAddr));

serverAddr.sin\_family = AF\_INET;

serverAddr.sin\_port = htons(PORT);

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

ret = connect(clientSocket, (struct sockaddr\*)&serverAddr, sizeof(serverAddr));

if(ret < 0) {printf("[-]Error in connection.\n"); exit(1); }

printf("[+]Connected to Server.\n");

while(1)

{ printf("\nEnter the city:\t");

scanf("%s",buffer);

send(clientSocket, buffer, strlen(buffer), 0);//send city name

if(strcmp(buffer, "exit") == 0)

{ close(clientSocket);

printf("[-]Disconnected from server.\n");

exit(1);

}

int num1;

read(clientSocket,&num1,sizeof(num1));

for(int flag=0;flag<num1;flag++)

{ printf("\n");

for(int ind=0;ind<4;ind++)

{ bzero(buffer,MAX);

recv(clientSocket, buffer, 1024, 0);

strcpy(str,buffer);

//decryption

for(int i=0; (i<1024 && str[i]!='\0');i++)

str[i] = str[i] - 5;

printf("%s\t: %s\n",arr[ind], str);

send(clientSocket, buffer, strlen(buffer), 0);

}//inner for loop end

bzero(buffer,MAX);

recv(clientSocket, buffer, 1024, 0);

strcpy(str,buffer);

for(int i=0; (i<1024 && str[i]!='\0'); i++)

str[i] = str[i] - 5;

printf("%s\t\t: %s\n",arr[4], str);

bzero(payment,MAX);

if(strcmp(str,"unbooked")==0 || strcmp(str,"pending")==0)

{ printf("Enter the payment method(cash\\loan\\none):");

scanf("%s",payment);

strcpy(str,payment);

//ecryption

for(int i=0; (i<1024 && str[i]!='\0');i++)

str[i] = str[i] + 5;

send(clientSocket, str, strlen(str), 0);

bzero(buffer,MAX);

recv(clientSocket, buffer, 1024, 0);

strcpy(str,buffer);

for(int i=0;(i<1024 && str[i]!='\0');i++)

str[i] = str[i] - 5;

printf("%s",str);

}

else

{ send(clientSocket, buffer, strlen(buffer), 0);

bzero(buffer,MAX);

recv(clientSocket, buffer, 1024, 0);

strcpy(str,buffer);

for(int i=0; (i<1024 && str[i]!='\0');i++)

str[i] = str[i] - 5;

printf("%s",str);

}

int choice;

printf("\nSee more property 1/0 :");

scanf("%d",&choice);

write(clientSocket,&choice,sizeof(choice));

if(choice==0) break;

}//outer for end

printf("No more property found in this location.\n");

}

return 0;

}

**OUTPUT:**

![Text

Description automatically generated]()**SERVER:**

**CLIENT – 1:**

![Text

Description automatically generated]()

![Text

Description automatically generated]()**CLIENT – 2:**

![Text

Description automatically generated]()**CLIENT – 3:**

![Text

Description automatically generated]()