**// Client.cpp**

#include <stdio.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <string.h>

#include <iostream>

#include <stdlib.h> /\* srand, rand \*/

#include <cstdlib>

#include <ctime>

#include <vector>

#define PORT 8080

using namespace std;

// function for string delimiter

vector<string> split(string s, string delimiter) {

size\_t pos\_start = 0, pos\_end, delim\_len = delimiter.length();

string token;

vector<string> res;

while ((pos\_end = s.find (delimiter, pos\_start)) != string::npos) {

token = s.substr (pos\_start, pos\_end - pos\_start);

pos\_start = pos\_end + delim\_len;

res.push\_back (token);

}

res.push\_back (s.substr (pos\_start));

return res;

}

int main(int argc, char const \*argv[])

{

srand((unsigned int)time(NULL)); // avoid always same output of rand()

float client\_local\_clock = rand() % 10; // range from 0 to 9

printf("Client starts. Client pid is %d \n", getpid());

printf("Client local clock is %f \n\n", client\_local\_clock);

int client\_socket\_fd, valread;

char client\_read\_buffer[1024] = {0};

struct sockaddr\_in server\_addr;

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

// server\_addr.sin\_addr.s\_addr = inet\_addr(argv[1]); // hardcode to 127.0.0.1

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

// Creating socket file descriptor (IPv4, TCP, IP)

if ((client\_socket\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) < 0)

{

printf("\n Client: Socket creation error \n");

return -1;

}

// Converting IPv4 and IPv6 addresses from text to binary form,

// from character string src into a network

// address structure in the af address family, then copies the

// network address structure to dst.

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

{

printf("\nClient: Invalid address/ Address not supported \n");

return -1;

}

// Connecting server, return 0 with success, return -1 with error

if (connect(client\_socket\_fd, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) < 0)

{

printf("\nClient: Connection Failed \n");

return -1;

}

char server\_ip[INET\_ADDRSTRLEN]="";

inet\_ntop(AF\_INET, &server\_addr.sin\_addr, server\_ip, INET\_ADDRSTRLEN);

printf("Client: connected server(%s:%d). \n", server\_ip, ntohs(server\_addr.sin\_port));

printf("\n\n");

// first round communicattion

// receiving form server

valread = read( client\_socket\_fd , client\_read\_buffer, 1024);

printf("Client: read: '%s'\n",client\_read\_buffer );

// convert char array to string

string recv\_msg = string(client\_read\_buffer);

// reply according to what client receive

if (strcmp(client\_read\_buffer, "Hello from server, please tell me your local clock value.") == 0) {

// prepare msg

string msg\_str = "Hello from client, my local clock value is " + to\_string(client\_local\_clock);

char msg\_char\_array[msg\_str.length() + 1];

strcpy(msg\_char\_array, msg\_str.c\_str());

// sending a message to server

send(client\_socket\_fd , &msg\_char\_array , strlen(msg\_char\_array) , 0 );

printf("Client: sent message: '%s'\n", msg\_char\_array);

}

//

// second round communicattion

//

// receiving form server

valread = read( client\_socket\_fd , client\_read\_buffer, 1024);

printf("Client: read: '%s'\n",client\_read\_buffer );

// convert char array to string

recv\_msg = string(client\_read\_buffer);

if (recv\_msg.find("From server, your clock adjustment offset is") != string::npos){ // if latter is a substring of former

string substr\_after\_lastbutone\_space;

string substr\_after\_last\_space;

vector<string> split\_str = split(recv\_msg, " ");

substr\_after\_lastbutone\_space = split\_str[ split\_str.size() - 2 ];

substr\_after\_last\_space = split\_str[ split\_str.size() - 1 ];

cout << "Client: received local clock adjustment offset (string) is " << substr\_after\_lastbutone\_space << " " << substr\_after\_last\_space << endl;

float substr\_after\_last\_space\_f = stof(substr\_after\_last\_space);

cout << "Client: received local clock adjustment offset (float) is " << substr\_after\_lastbutone\_space << " " << substr\_after\_last\_space\_f << endl;

char oper\_char\_array[substr\_after\_lastbutone\_space.length() + 1];

strcpy(oper\_char\_array, substr\_after\_lastbutone\_space.c\_str());

if (strcmp(oper\_char\_array, "add") == 0 ){

client\_local\_clock += substr\_after\_last\_space\_f;

}else if (strcmp(oper\_char\_array, "minus") == 0 ){

client\_local\_clock -= substr\_after\_last\_space\_f;

}

printf("Client local clock is %f \n\n", client\_local\_clock);

}

close(client\_socket\_fd);

return 0;

}