//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// File: cliente.c

// Author: Javier Vivas Gomez

// Date: Abril de 2020

// Coms: Ejemplo de cliente mediante sockets

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <signal.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <pthread.h>

#define LENGTH 2048

/\* Variables Globales \*/

volatile sig\_atomic\_t flag = 0;

int sockfd = 0;

void fin\_cadena (char\* cadena, int length) {

int i;

for (i = 0; i < length; i++) {

if (cadena[i] == '\n') {

cadena[i] = '\0';

break;

}

}

}

void enviar() {

char message[LENGTH] = {};

char buffer[LENGTH] = {};

while(1) {

fgets(message, LENGTH, stdin);

fin\_cadena(message, LENGTH);

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

break;

}

else {

sprintf(buffer, "%s\n", message);

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

}

bzero(message, LENGTH);

bzero(buffer, LENGTH);

}

}

void recepcion() {

char msg[LENGTH] = {};

while (1) {

int recibido = recv(sockfd, msg, LENGTH, 0);

if (recibido > 0) {

printf("%s", msg);

}

else if (recibido == 0) {

break;

}

else {

// -1

}

memset(msg, 0, sizeof(msg));

}

}

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

struct sockaddr\_in ser\_addr; //Declaracion de la estructura de direcciones

pthread\_t hiloEnvio; //Declaracion del identificador del hilo de envio

pthread\_t hiloRecep; //Declaracion del identificador del hilo de recepcion

if(argc != 3){

printf("Usage: %s <direccion IP> <puerto>\n", argv[0]);

return EXIT\_FAILURE;

}

int puerto = atoi(argv[2]);

//Configuracion de la estructura de direcciones

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

ser\_addr.sin\_family = AF\_INET; //AF\_INET = IPv4

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

//Campo correspondiente a la direccion IP que se le asigna.

ser\_addr.sin\_port = htons(puerto);

// Conexion al servidor

int err = connect(sockfd, (struct sockaddr \*)&ser\_addr, sizeof(ser\_addr));

if (err == -1) {

printf("ERROR: connect\n");

return EXIT\_FAILURE;

}

printf("...Estableciendo conexion con el servidor...\n");

//Creacion y lanzamiento del hilo de envio

if(pthread\_create(&hiloEnvio, NULL, (void \*) enviar, NULL) != 0){

printf("ERROR: pthread\n");

return EXIT\_FAILURE;

}

//Creacion y lanzamiento del hilo de recepcion

if(pthread\_create(&hiloRecep, NULL, (void \*) recepcion, NULL) != 0){

printf("ERROR: pthread\n");

return EXIT\_FAILURE;

}

while (1){

if(flag){

printf("\nAdios\n");

break;

}

}

close(sockfd);

return EXIT\_SUCCESS;

}