



# Proyecto Final

13.08.2021

---

Cortez Ibarra Derek

Jiménez Hernández Francisco Williams

Arquitectura Cliente-Servidor

Profesor: Román Zamitis Carlos Alberto

Grupo: 01

Universidad Nacional Autónoma de México

## Código Fuente

Cliente: client.c

```
// Importa librerías necesarias      You, 3 weeks ago • Creación del repositorio
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include "utils.c"

// Constante con el número de puerto por el cual se va conectar el cliente
#define PORT 3490

// Constante con el número máximo de bytes que podemos enviar
#define MAXDATASIZE 300

void main(int argc, char *argv[]){

    // Declaración de variables y estructuras necesarias
    int sockfd, numbytes;
    char * words[3];           // Para almacenar el comando dividido en 3
    char buf[MAXDATASIZE];     // Para almacenar el comando y respuesta del servidor
    struct hostent *he;        // Información del host
    struct sockaddr_in their_addr; // Información de direcciones de conectores (puerto)

    // Verifica si se proporcionó un argumento de línea de comandos
    if(argc != 2) {
        fprintf(stderr, "Client-Usage: %s host_servidor\n", argv[0]);
        exit(1);
    }

    // Obtiene la información del host
    if((he = gethostbyname(argv[1])) == NULL) {

        // Si no obtiene información del host, termina el programa
        perror("gethostbyname()");
        exit(1);
    }else {

        // Si obtiene información, muestra el host
        printf("Client-The remote host is: %s\n", argv[1]);
    }
}
```

```
// Crea un socket para envío y recepción de info. con direcciones IPv4
if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1){

    // Si no se puede crear, muestra el error y termina el programa
    perror("socket()");
    exit(1);
}else {

    // Si se crea, muestra un mensaje de éxito
    printf("Client-The socket() sockfd is OK...\n");
}

// Orden de bytes del host (4 bytes para direcciones IPv4 -> AF_INET)
their_addr.sin_family = AF_INET;

// Almacena en la estructura tipo sockaddr_in el puerto de comunicación
// y la dirección IP del host en el campo s_addr de la estructura interna sin_addr
printf("Server-Using %s and port %d...\n", argv[1], PORT);
their_addr.sin_port = htons(PORT);
their_addr.sin_addr = *((struct in_addr *)he->h_addr);

// Establece en 0 el campo sin_zero de la estructura sockaddr_in
memset(&(their_addr.sin_zero), '\0', 8);

// Intenta hacer una conexión a un socket (descriptor de archivo 'sockfd')
if(connect(sockfd, (struct sockaddr *)&their_addr, sizeof(struct sockaddr)) == -1) {

    // Si la conexión falla, muestra el error y termina el programa
    perror("connect()");
    exit(1);
}
else {

    // Si no falla, muestra un mensaje de éxito
    printf("Client-The connect() is OK...\n");
}

// Lee el mensaje del usuario
printf("Enter a command: ");
fgets(buf, MAXDATASIZE-1, stdin);

// Divide el comando
split_string(buf, words);
```

```

// Valida el comando
if (validate_command(words) == 1)
{
    // Si es válido, envía mensaje al servidor
    if ((numbytes = send(sockfd, buf, MAXDATASIZE-1, 0)) == -1) {

        // Si ocurre un error, se termina el programa
        perror("send()");
        exit(1);
    } else {

        // Si se envía con éxito, muestra un mensaje
        printf("Client-send() is OK...\n");
    }
} else {

    // Si no es válido, muestra un mensaje de error y termina el programa
    printf("Invalid command: Please enter a valid command.\n");
    exit(1);
}

/* Recibe un mensaje del servidor; almacena el mensaje en el buffer (buf) y
el tamaño del mensaje escrito en el buffer es retornado (numbytes) */
if((numbytes = recv(sockfd, buf, MAXDATASIZE-1, 0)) == -1) {

    // Si ocurre un error, se termina el programa
    perror("recv()");
    exit(1);
} else {

    // Si se recibe el mensaje sin problema, muestra un mensaje de éxito
    printf("Client-The recv() is OK...\n");
}

// Al final del mensaje recibido coloca el caracter de fin d cadena (\0)
//buf[numbytes] = '\0';

// Muestra el mensaje recibido y cierra el socket (Descriptor de archivo)
printf("Client-Received: %s\n", buf);
printf("Client-Closing sockfd\n");
close(sockfd);
}

```

## Servidor: server.c

```
// Importa librerías necesarias      You, 3 weeks ago • Creación del repositorio
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/wait.h>
#include <signal.h>
#include "utils.c"

// Constante con el número de puerto por el cuál va escuchar peticiones el servidor
#define MYPORT 3490

// Tamaño de la cola para conexiones pendientes
#define BACKLOG 10

// Constante con el número máximo de bytes que podemos enviar
#define MAXDATASIZE 300

// Función de captura de señal
void sigchld_handler(int s){
    while(wait(NULL) > 0);
}

int main(int argc, char *argv[ ]){

    // Declaración de variables y estructuras necesarias
    int sockfd, new_fd, numbytes;
    char buf[MAXDATASIZE];
    char * words[3];           // Para almacenar el comando dividido en 3
    char response[100];        // Almacena la respuesta del servidor
    int result;                 // Para almacenar el resultado del servidor
    char file_name[30];

    // Información de direcciones del servidor
    struct sockaddr_in my_addr;

    // Información de direcciones de conectores (puerto)
```

```

struct sockaddr_in their_addr;
int sin_size;
struct sigaction sa;
int yes = 1;

// Crea un socket
if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {

    // Si no se puede crear, muestra el error y termina el programa
    perror("Server-socket() error lol!");
    exit(1);
}else {

    // Si se crea, muestra un mensaje de éxito
    printf("Server-socket() sockfd is OK...\n");
}

/*
setsockopt() - Establece las opciones asociadas al socket sockfd:

Establece la opción SO_REUSEADDR en el nivel de protocolo SOL_SOCKET, al valor
apuntado por 'yes' para el socket asociado con el descriptor de archivo 'sockfd'.

- SO_REUSEADDR: Especifica que las reglas utilizadas en la validación de direcciones
proporcionadas a bind() deben permitir la reutilización de direcciones locales, si el
protocolo lo admite. Esta opción toma un valor int. Esta es una opción booleana.

SO_REUSEADDR se configura comúnmente en programas de servidor de red, ya que un
patrón de uso común es realizar un cambio de configuración, luego se requiere
reiniciar ese programa para que el cambio surta efecto. Sin SO_REUSEADDR, la llamada
bind() en la nueva instancia del programa reiniciado fallará si había conexiones
abiertas a la instancia anterior cuando la eliminó. Esas conexiones mantendrán el puerto
TCP en el estado TIME_WAIT durante 30-120 segundos.

- SOL_SOCKET es la propia capa de conexión. Se utiliza para opciones que son independientes
del protocolo.
*/
if (setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(int)) == -1) {

    // Si ocurre un error, lo comunica
    perror("Server-setsockopt() error lol!");
    exit(1);
}else {

    // Si no ocurre un error, emite un mensaje de éxito
    printf("Server-setsockopt is OK...\n");
}

```

```

// Orden de bytes del host (4 bytes para direcciones IPv4 -> AF_INET)
my_addr.sin_family = AF_INET;

/* Almacena en la estructura de tipo sockaddr_in llamada my_addr, el puerto de comunicación
y la dirección IP en el campo s_addr de la estructura interna sin_addr */

/*
  INADDR_ANY: en realidad es la IP especial 0.0.0.0 que representa
  "cualesquiera que sean las IPs de todas las interfaces de red de este ordenador".

  Al usar INADDR_ANY, por tanto, se consiguen dos cosas:

  1. Admitir datos que provengan de cualquiera de las interfaces de red que el servidor
  tenga.
  2. Aunque el servidor cambie su IP, el programa seguirá funcionando sin cambios.
*/
my_addr.sin_port = htons(MYPORT);
my_addr.sin_addr.s_addr = INADDR_ANY;
printf("Server-Using %s and port %d...\n", inet_ntoa(my_addr.sin_addr), MYPORT);

// Establece en 0 el campo sin_zero de la estructura sockaddr_in
memset(&(my_addr.sin_zero), '\0', 8);

// Intenta enlazar el servidor al socket 'sockfd' para escuchar peticiones por ahí
if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr)) == -1) {

    // Si el enlace falla, muestra el error y termina el programa
    perror("Server-bind() error");
    exit(1);
} else {

    // Si el enlace no falla, muestra un mensaje de éxito
    printf("Server-bind() is OK...\n");
}

// Escucha las conexiones de socket y limita la cola de conexiones entrantes
if (listen(sockfd, BACKLOG) == -1) {
    perror("Server-listen() error");
    exit(1);
}

printf("Server-listen() is OK...Listening...\n");

```

```

// Limpia todos los procesos muertos
sa.sa_handler = sigchld_handler;
sigemptyset(&sa.sa_mask);
sa.sa_flags = SA_RESTART;

/*
    sigaction - Examina y cambia una acción de señal

    int sigaction(int sig, const struct sigaction *restrict act, struct sigaction *restrict oact);

    La función sigaction() permite que el proceso de llamada examine y/o especifique la
    acción que se asociará con una señal específica. El argumento sig especifica la señal;
    los valores aceptables se definen en <signal.h>.

    La estructura sigaction, se utiliza para describir una acción a realizar.

    - SIGCHLD es la señal enviada a un proceso cuando uno de sus procesos hijos termina.
    - &sa es la estructura de tipo sigaction que especifica la acción a realizar.
*/
if (sigaction(SIGCHLD, &sa, NULL) == -1) {
    // Si falla muestra el error y termina el programa
    perror("Server-sigaction() error");
    exit(1);
} else {
    // Si no falla muestra un mensaje de éxito
    printf("Server-sigaction() is OK...\n");
}

// Ciclo infinito para aceptar peticiones
while(1) {

    // Almacena el tamaño de una estructura tipo sockaddr_in
    sin_size = sizeof(struct sockaddr_in);

    // Acepta una nueva conexión en un socket
    if ((new_fd = accept(sockfd, (struct sockaddr *)&their_addr, &sin_size)) == -1) {
        //Si ocurre un error, muestra un mensaje y el programa continua
        perror("Server-accept() error");
        continue;
    } else {
        // Si no ocurre error se muestran un par de mensajes de éxito
    }
}

```



```

    // Si no ocurre error, se muestran un par de mensajes de éxito
    printf("Server-accept() is OK...\n");
    printf("Server-new socket, new_fd is OK...\n");
    printf("Server: Got connection from %s\n", inet_ntoa(their_addr.sin_addr));
}

// Se crea un proceso hijo para que atienda al cliente
if(!fork()){
    // Aquí entra el proceso hijo únicamente

    // El proceso hijo cierra el socket 'sockfd' porque no lo necesita
    close(sockfd);

    // Recibe un mensaje del cliente
    if((numbytes = recv(new_fd, buf, MAXDATASIZE-1, 0)) == -1) {
        // Si ocurre un error, lo reporta y el programa continua
        perror("Server-recv() error lol!");
    } else {

        // Si se recibe el mensaje sin problema, muestra el mensaje del cliente
        printf("Server-recv() is OK...\n");
        buf[numbytes] = '\0';

        // Divide el mensaje
        split_string(buf, words);

        strcpy(file_name, words[1]);
        strcat(file_name, ".txt");

        printf("Server-Received: %s %s %s\n", words[0], words[1], words[2]);

        // Atiende la petición
        result = attend_request(words, file_name, response);

        if (result == 0) {
            printf("Server-insert() is OK...\n");

            // Envía un mensaje en un socket
            if(send(new_fd, "Successful insert!", MAXDATASIZE-1, 0) == -1) {
                // Si ocurre un error local, lo reporta y el programa continua
                perror("Server-send() error lol!\n");
            }
        } else if (result == 1){
            printf("Server-select() is OK...\n");

            // Envía un mensaje en un socket
            if(send(new_fd, response, MAXDATASIZE-1, 0) == -1) {

```

```
        if(send(new_fd, response, MAXDATASIZE-1, 0) == -1) {
            // Si ocurre un error local, lo reporta y el programa continua
            perror("Server-send() error lol!\n");
        }
    } else if (result == 2) {

        // Envía un mensaje en un socket
        if(send(new_fd, "Can't open file.", MAXDATASIZE-1, 0) == -1) {
            // Si ocurre un error local, lo reporta y el programa continua
            perror("Server-send() error lol!\n");
        }

        printf("Server-unexpected-error: Can't open file.\n");
    } else {
        printf("An unexpected error has occurred.\n");
    }

    printf("Server-select() has responded...\n");
}

// El proceso hijo ierra el socket de comunicación con el cliente y termina su ejecución
close(new_fd);
exit(0);
}else {

    // El proceso padre muestra un mensaje de éxito
    printf("Server-send is OK...!\n");
}

// El proceso padre cierra el socket de comunicación con el cliente, no lo necesita
close(new_fd);
printf("Server-new socket, new_fd closed successfully...\n");
}

// Termina el servidor
return 0;
}
```

## Utilities: utils.c

```
#include <stdio.h>      You, 3 weeks ago • INSERT: Terminado

int attend_request(char * command[], char * file_name, char * response)
{
    FILE *file;          // Variable apuntador al archivo
    char content[50];     // Variable para el contenido del archivo

    // Si el comando es INSERT
    if ((strcmp(command[0], "INSERT") == 0) || (strcmp(command[0], "insert") == 0))
    {
        // Copia el contenido del archivo
        strcpy(content, command[2]);

        // Crea el archivo con el número de cuenta como nombre y escribe el nombre del alumno en él
        file = fopen(file_name, "w");

        // Si el archivo no se abrió
        if (file == NULL)
        {
            return 2;
        }
        else
        // Si el archivo se abre correctamente
        {
            fprintf(file, "%s", content);

            fflush(stdout);
            fflush(file);

            fclose(file);

            return 0;
        }
    }
    // Si el comando es SELECT
    else if ((strcmp(command[0], "SELECT") == 0) || (strcmp(command[0], "select") == 0))
    {
        // Abre el archivo en modo lectura
        file = fopen(file_name, "r");

        // Si el archivo no se abrió
        if (file == NULL)
        {
            return 2;
        }
        // Si el archivo se abre correctamente
```

```

// Si el archivo se abre correctamente
else
{
    // Lee el contenido y lo almacena en el arreglo command
    fgets(response, 100, file);
    printf("Command[2]: %s\n", response);
    return 1;
}

}

// Si por alguna razón no es ninguno de los dos
else
{
    return 2;
}

}

// Valida el comando introducido por el usuario
int validate_command(char * command_pieces[])
{
    if (((strcmp(command_pieces[0], "INSERT") == 0) ||
        (strcmp(command_pieces[0], "SELECT") == 0) ||
        (strcmp(command_pieces[0], "insert") == 0) ||
        (strcmp(command_pieces[0], "select") == 0)) &&
        ((int*)command_pieces[1] > 0)) {
        return 1;
    } else {
        return 0;
    }
}

// Divide una cadena de caracteres en 3 piezas
void split_string(char * inputString, char * pieces[])
{
    int indexCtr = 0;
    int wordIndex = 0;
    int totalWords = 0;
    char words[3][50];

    // Loop through each character in the string
    for (indexCtr = 0; indexCtr <= strlen(inputString); indexCtr++)
    {
        // If current character is space or null
        if (inputString[indexCtr] == ' ' || inputString[indexCtr] == '\0')
        {
            // If the first two words have already been copied

```

```
// If the first two words have already been copied
if (totalWords >= 2)
{
    // If it's the end of the string
    if (inputString[indexCtr] == '\0')
    {
        // Append the null character to the current word
        words[totalWords][wordIndex] = '\0';
    }
    // If it's not the end of the string
    else
    {
        // Append a space character to the current word
        words[totalWords][wordIndex] = ' ';


        // Increment the index for the word
        wordIndex++;
    }
}
// The first two words are not copied yet
else
{
    // Append the null character to the current word
    words[totalWords][wordIndex] = '\0';

    // Append the entire word to the list of pieces
    pieces[totalWords] = words[totalWords];

    // Increment total words found
    totalWords++;

    // Reset the index to zero for the next word
    wordIndex = 0;
}
}
// Found a character of a word
else
{
    if (inputString[indexCtr] != '\n')
    {
        // Copy the character of the word
        words[totalWords][wordIndex] = inputString[indexCtr];

        // Increment the index for the word
        wordIndex++;
    }
}
```



```
// Append the entire word to the list of pieces
pieces[totalWords] = words[totalWords];
}
```

## Ejecución

### Servidor: Terminal

```
williams@DESKTOP-02RKFL3:/mnt/c/Users/Williams Jiménez/Documents/Educación/Universidad/Semestre 10 (2
021-2)/Arquitectura cliente servidor/Proyecto/cliente-servidor$ gcc server.c -o server
williams@DESKTOP-02RKFL3:/mnt/c/Users/Williams Jiménez/Documents/Educación/Universidad/Semestre 10 (2
021-2)/Arquitectura cliente servidor/Proyecto/cliente-servidor$ ./server
Server-socket() sockfd is OK...
Server-setsockopt is OK...
Server-Using 0.0.0.0 and port 3490...
Server-bind() is OK...
Server-listen() is OK...Listening...
Server-sigaction() is OK...
Server-accept() is OK...
Server-new socket, new_fd is OK...
Server: Got connection from 127.0.0.1
Server-send is OK...!
Server-new socket, new_fd closed successfully...
Server-recv() is OK...
Server-Received: INSERT 314182144 Jiménez Hernández Francisco Williams
Server-insert() is OK...
Server-select() has responded...
Server-accept() is OK...
Server-new socket, new_fd is OK...
Server: Got connection from 127.0.0.1
Server-send is OK...!
Server-new socket, new_fd closed successfully...
Server-recv() is OK...
Server-Received: INSERT 123456789 Cortez Ibarra Derek
Server-insert() is OK...
Server-select() has responded...
Server-accept() is OK...
Server-new socket, new_fd is OK...
Server: Got connection from 127.0.0.1
Server-send is OK...!
Server-new socket, new_fd closed successfully...
Server-recv() is OK...
Server-Received: SELECT 123456789
Command[2]: Cortez Ibarra Derek
Server-select() is OK...
```

## Cliente: Terminal

```
williams@DESKTOP-02RKFL3:/mnt/c/Users/Williams Jiménez/Documents/Educación/Universidad/Semestre 10 (2021-2)/Arquitectura cliente servidor/Proyecto/cliente-servidor$ gcc client.c -o client
williams@DESKTOP-02RKFL3:/mnt/c/Users/Williams Jiménez/Documents/Educación/Universidad/Semestre 10 (2021-2)/Arquitectura cliente servidor/Proyecto/cliente-servidor$ ./client 0.0.0.0
Client-The remote host is: 0.0.0.0
Client-The socket() sockfd is OK...
Server-Using 0.0.0.0 and port 3490...
Client-The connect() is OK...
Enter a command: INSERT 314182144 Jiménez Hernández Francisco Williams
Client-send() is OK...
Client-The recv() is OK...
Client-Received: Successful insert!
Client-Closing sockfd
williams@DESKTOP-02RKFL3:/mnt/c/Users/Williams Jiménez/Documents/Educación/Universidad/Semestre 10 (2021-2)/Arquitectura cliente servidor/Proyecto/cliente-servidor$ ./client 0.0.0.0
Client-The remote host is: 0.0.0.0
Client-The socket() sockfd is OK...
Server-Using 0.0.0.0 and port 3490...
Client-The connect() is OK...
Enter a command: INSERT 123456789 Cortez Ibarra Derek
Client-send() is OK...
Client-The recv() is OK...
Client-Received: Successful insert!
Client-Closing sockfd
williams@DESKTOP-02RKFL3:/mnt/c/Users/Williams Jiménez/Documents/Educación/Universidad/Semestre 10 (2021-2)/Arquitectura cliente servidor/Proyecto/cliente-servidor$ ./client 0.0.0.0
Client-The remote host is: 0.0.0.0
Client-The socket() sockfd is OK...
Server-Using 0.0.0.0 and port 3490...
Client-The connect() is OK...
Enter a command: SELECT 123456789
Client-send() is OK...
Client-The recv() is OK...
Client-Received: Cortez Ibarra Derek
Client-Closing sockfd
williams@DESKTOP-02RKFL3:/mnt/c/Users/Williams Jiménez/Documents/Educación/Universidad/Semestre 10 (2021-2)/Arquitectura cliente servidor/Proyecto/cliente-servidor$ ./client 0.0.0.0
Client-The remote host is: 0.0.0.0
Client-The socket() sockfd is OK...
Server-Using 0.0.0.0 and port 3490...
Client-The connect() is OK...
Enter a command: SELECT 314182144
Client-send() is OK...
Client-The recv() is OK...
Client-Received: Jiménez Hernández Francisco Williams
Client-Closing sockfd
williams@DESKTOP-02RKFL3:/mnt/c/Users/Williams Jiménez/Documents/Educación/Universidad/Semestre 10 (2021-2)/Arquitectura cliente servidor/Proyecto/cliente-servidor$
```



## Archivos

