

# Proyecto Final

Martínez Ostoa Néstor Iván  
#315618648  
Arquitectura Cliente Servidor - 2946

Agosto 2021

## 1. Objetivos

Realizar una arquitectura cliente servidor en lenguaje C para simular un manejador de base de datos. El proyecto deberá cumplir con lo siguiente:

- El cliente debe enviar un comando al servidor
- El servidor debe recibir las instrucciones del cliente y realizar las operaciones correspondientes
- El servidor deberá responderle de vuelta al cliente
- El servidor deberá ejecutar sus acciones sobre archivos

## 2. Descripción de la arquitectura

Como se mencionó en los objetivos, la idea de esta arquitectura será simular un manejador de base de datos en donde el cliente será responsable de ejecutar secuencias de SQL y el servidor de responder a dichas secuencias y manejar los archivos correspondientes de la base de datos. A continuación se especifican las responsabilidades de cada entidad.

### 2.1. Servidor

Los comandos que el servidor reciba se ejecutarán sobre archivos y éste deberá ser capaz de procesar de manera correcta comandos como el siguiente:

```
INSERT numcta apellido_paterno apellido_materno nombre(s)
```

donde **numcta** es el número de cuenta del alumno a insertar en la base de datos. El servidor deberá manejar las inserciones mediante archivos cuyo nombre será el valor de **numcta**. Se tendrá que tener un archivo por alumno. Una vez que se realice la inserción, el servidor deberá responder al cliente con un mensaje de éxito o error en caso de existir uno.

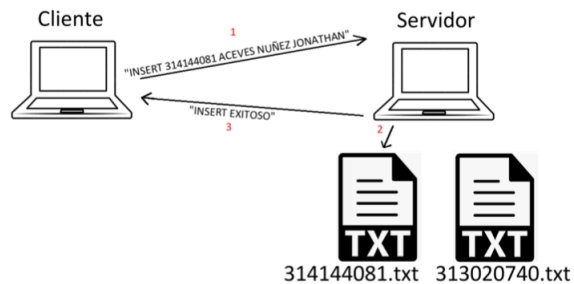


Figura 1: Ejemplo de inserción de elementos en la base de datos

## 2.2. Cliente

Por otro lado, el cliente a parte de poder insertar elementos a la base de datos, será capaz de hacer selecciones con base en un número de cuenta. El cliente, por medio del comando `select numcta` podrá obtener el contenido del archivo `numcta`. Cuando el servidor reciba esta instrucción, deberá responder con el contenido del archivo en caso de que exista. De lo contrario, deberá mandar un mensaje de error.

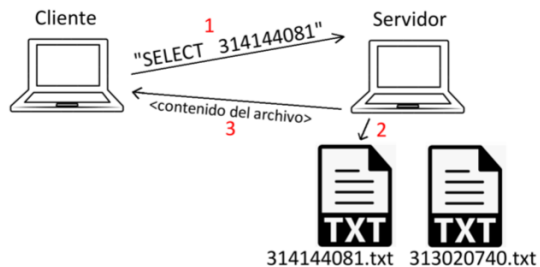


Figura 2: Ejemplo de selección de elementos dentro la base de datos

## 3. Desarrollo

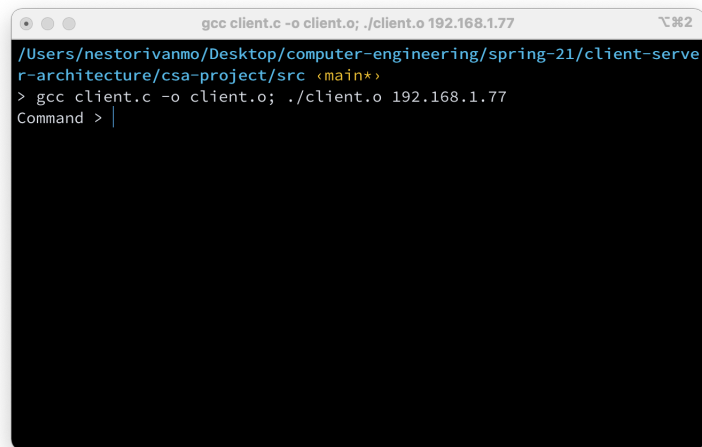
A continuación se muestra capturas de pantalla con las salidas de ejecución desde la terminal mostrando el funcionamiento del programa:

1. Compilamos y ejecutamos el servidor



```
gcc server.c -o server.o; ./server.o
/Users/nestorivanmo/Desktop/computer-engineering/spring-21/client-server-architecture/csa-project/src <main*>
> gcc server.c -o server.o; ./server.o
**SERVER** is ready to accept requests...
-----
```

2. Compilamos y ejecutamos el cliente con la dirección IP del host el cual queda a la espera de los comandos:



```
gcc client.c -o client.o; ./client.o 192.168.1.77
/Users/nestorivanmo/Desktop/computer-engineering/spring-21/client-server-architecture/csa-project/src <main*>
> gcc client.c -o client.o; ./client.o 192.168.1.77
Command > |
```

3. Inserción de un alumno

```
gcc client.c -o client.o; ./client.o 192.168.1.77
/Users/nestorivanmo/Desktop/computer-engineering/spring-21/client-serve
r-architecture/csa-project/src <main*>
> gcc client.c -o client.o; ./client.o 192.168.1.77
Command > insert 315618648 martinez ostoia nestor ivan
response -> insertion successful
Command > insert 3141440841 aceves nuñez jonathan
response -> insertion successful
Command > |
```

```
gcc server.c -o server.o; ./server.o
/Users/nestorivanmo/Desktop/computer-engineering/spring-21/client-serve
r-architecture/csa-project/src <main*>
> gcc server.c -o server.o; ./server.o
**SERVER** is ready to accept requests...
-----

server received -> insert 315618648 martinez ostoia nestor ivan

Processing insertion...
server received -> insert 3141440841 aceves nuñez jonathan

Processing insertion...
```

#### 4. Recuperación de los datos de un alumno

```
gcc client.c -o client.o; ./client.o 192.168.1.77 130 ↵
/Users/nestorivanmo/Desktop/computer-engineering/spring-21/client-serve
r-architecture/csa-project/src <main*>
> gcc client.c -o client.o; ./client.o 192.168.1.77
Command > select 315618648
response -> martinez ostoia nestor ivan

Command > select 3141440841
response -> aceves nuñez jonathan

Command > |
```

```
gcc server.c -o server.o; ./server.o 141 ↵
/Users/nestorivanmo/Desktop/computer-engineering/spring-21/client-serve
r-architecture/csa-project/src <main*>
> gcc server.c -o server.o; ./server.o
**SERVER** is ready to accept requests...
-----

server received -> select 315618648

Processing selection...
|315618648.txt|
server received -> select 3141440841

Processing selection...
|3141440841.txt|
|
```

## 4. Código

El código que se menciona a continuación también se puede encontrar en la siguiente URL

### 4.1. Servidor

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
```

```

#include <string.h>
#include <ctype.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 <dirent.h>

#define PORT 3490
#define BACKLOG 10
#define MAXDATASIZE 100
#define ENTITY "**SERVER**"
#define DB_FILES_PATH "db_files"

void handle_system_call(int result, char *system_call_msg, int verbose) {
    if (result == -1) {
        perror("error:");
        printf("%s: error at %s\n", ENTITY, system_call_msg);
        exit(1);
    }
    if (verbose == 1) {
        printf("%s: success at %s\n", ENTITY, system_call_msg);
    }
}

char *process_insertion(char instructions[][MAXDATASIZE]) {
    char *account_number = instructions[1];
    char *first_last_name = instructions[2];
    char *second_last_name = instructions[3];
    char *first_name = instructions[4];
    char *middle_name = instructions[5];

    FILE *fp;
    char file[100];
    sprintf(file, "%s.txt", account_number);
    fp = fopen(file, "w");

    if (fp == NULL) {
        return "insertion failed...";
    }

    fflush(stdin);
    fprintf(fp, "%s_%s_%s_%s", first_last_name, second_last_name, first_name, middle_name);
    fclose(fp);
}

```

```

    return "insertion_successful";
}

char* process_selection(char account_number[MAXDATASIZE]) {
    char num_cta[MAXDATASIZE];

    int j = 0;
    for (j = 0; j < MAXDATASIZE; j++)
        num_cta[j] = '\0';

    int i = 0;
    int digit_idx = 0;
    for (i = 0; i < MAXDATASIZE; i++) {
        if (isdigit(account_number[i])) {
            num_cta[digit_idx] = account_number[i];
            digit_idx++;
        }
    }

    int idx = 0;
    char extension[4] = ".txt";
    for (i = digit_idx; i < digit_idx + 4; i++) {
        num_cta[i] = extension[idx];
        idx++;
    }

    printf("|%s|\n", num_cta);

    FILE *fp;
    fp = fopen(num_cta, "r");
    if (fp == NULL) {
        return "error_reading_file";
    }

    char fln[MAXDATASIZE];
    char sln[MAXDATASIZE];
    char fn[MAXDATASIZE];
    char mn[MAXDATASIZE];
    fscanf(fp, "%s_%s_%s_%s", fln, sln, fn, mn);

    static char db_result[MAXDATASIZE];
    sprintf(db_result, "%s_%s_%s_%s\n", fln, sln, fn, mn);

    //printf("%s\n", db_result);

```

```

    return db_result;
}

char *process_request(char *buffer) {
    printf("\tserver_received -> %s\n", buffer);

    char instructions[10][MAXDATASIZE];
    int idx = 0;

    char *pch = strtok(buffer, " ");
    while (pch != NULL) {
        strcpy(instructions[idx], pch);
        pch = strtok(NULL, " ");
        idx++;
    }

    if (strcmp(instructions[0], "insert") == 0) {
        printf("\tProcessing_insertion...\n");
        return process_insertion(instructions);
    } else if (strcmp(instructions[0], "select") == 0) {
        printf("\tProcessing_selection...\n");
        return process_selection(instructions[1]);
    } else if (strcmp(instructions[0], "exit") == 10) {
        return "bye";
    } else {
        return strcat(instructions[0], "_not_recognized");
    }
}

void accept_requests(int server_socket_fd, int verbose) {
    char buffer[MAXDATASIZE];
    int bytes_received;
    char *response;

    //ACCEPT()
    int client_socket_fd;
    struct sockaddr_in client_ip_addr;
    unsigned int sockaddr_in_size = sizeof(client_ip_addr);
    client_socket_fd = accept(
        server_socket_fd, (struct sockaddr*)&client_ip_addr, &sockaddr_in_size
    );
    handle_system_call(client_socket_fd, "ACCEPT()", verbose);

    while (1) {
        //RECV()

```



```

        if((bytes_received = recv(client_socket_fd , buffer , MAXDATASIZE-1, 0)) == -1){
            perror("SERVER_recv()");
        } else {
            buffer[bytes_received] = '\0';
            if (strlen(buffer) > 1) {
                response = process_request(buffer);
            }
        }

        //SEND()
        if (send(client_socket_fd , response , strlen(response), 0) == -1) {
            printf("%s: SEND()_error\n", ENTITY);
        }
    }
}

int main(int argc , char *argv[ ]){
    int verbose = 0;
    int system_call_result;

    printf("%s_is_ready_to_accept_requests...\n");

    //SOCKET()
    int server_socket_fd;
    int domain = PF_INET;
    int type = SOCK_STREAM;
    int protocol = 0;
    server_socket_fd = socket(domain, type, protocol);
    handle_system_call(server_socket_fd , "SOCKET()", verbose);

    //BIND()
    int yes=1;
    if (setsockopt(server_socket_fd , SOL_SOCKET,SO_REUSEADDR, &yes , sizeof yes) == -1)
        perror("setsockopt");
        exit(1);
}
struct sockaddr_in server_ip_addr;
server_ip_addr.sin_family = AF_INET;
server_ip_addr.sin_port = htons(PORT);
server_ip_addr.sin_addr.s_addr = INADDR_ANY;
memset(&(server_ip_addr.sin_zero), '\0', 8);
system_call_result = bind(
    server_socket_fd , (struct sockaddr*)&server_ip_addr ,
    sizeof(struct sockaddr)
);
handle_system_call(system_call_result , "BIND()", verbose);

```

```

//LISTEN()
system_call_result = listen(server_socket_fd , BACKLOG);
handle_system_call(system_call_result , "LISTEN()", verbose);

//CLIENT SERVER INTERACTION
accept_requests(server_socket_fd , verbose);

//CLOSE()
close(server_socket_fd);

return 0;
}

```

## 4.2. Cliente

```

#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>

#define PORT 3490
#define MAXDATASIZE 1024
#define ENTITY "**CLIENT**"

void handle_system_call(int result , char *system_call_msg , int verbose) {
    if (result == -1) {
        printf("%s: _error_ at _%s\n", ENTITY, system_call_msg);
        exit(1);
    }
    if (verbose == 1) {
        printf("%s: _success_ at _%s\n", ENTITY, system_call_msg);
    }
}

void handle_client_input_parameters(int argc , char *ip_address , int verbose) {
    if (ip_address == NULL) {
        printf("%s: _missing_ ip _address\n", ENTITY);
        exit(1);
    }
}

```

```

    if (verbose == 1){
        printf("%s: _input_parameters_are_ok!\n", ENTITY);
    }
}

void verify_host_name(struct hostent *he, int verbose) {
    if (he == NULL) {
        printf("%s: _error_at_GETHOSTBYNAME()\n", ENTITY);
        exit(1);
    }
    if (verbose == 1){
        printf("%s: _success_at_GETHOSTBYNAME()\n", ENTITY);
    }
}

char *read_string(char *msg) {
    char *str;
    char buf[MAXDATASIZE];
    printf("%s", msg);
    fgets(buf, MAXDATASIZE, stdin);
    str = (char*) malloc(strlen(buf) + 1);
    if (str == NULL) {
        printf("%s: _error_out_of_memory\n", ENTITY);
        exit(1);
    }
    strcpy(str, buf);
    return str;
}

char *handle_server_msgs(char *buffer) {
    printf("\tresponse->_%s\n", buffer);
    return "";
}

void handle_connection_with_server(int client_socket_fd, int verbose) {
    char buffer[MAXDATASIZE];
    int bytes_received;

    char stop_command[] = "exit";
    char command[MAXDATASIZE];
    do {
        printf("Command->_");
        fflush(stdout);
        fgets(command, MAXDATASIZE, stdin);

        //SEND()

```

```

        if(send(client_socket_fd , command, strlen(command), 0) == -1) {
            printf("%s: _SEND() _error\n", ENTITY);
        }

        //RECV()
        if((bytes_received = recv(client_socket_fd , buffer , MAXDATASIZE-1, 0)) == -1){
            printf("%s: _RECV() _error\n", ENTITY);
        }
        buffer[bytes_received] = '\0';
        char *response = handle_server_msgs(buffer);

    } while (strcmp(command, stop_command) != 10);
}

int main(int argc , char *argv []){
    int verbose = 0;
    int system_call_result;

    //VERIFY CORRECT INPUT PARAMETERS
    char *client_input_ip_addr = argv[1];
    handle_client_input_parameters(argc , client_input_ip_addr , verbose);

    //HOSTBYNAME
    struct hostent *he;
    he = gethostbyname(client_input_ip_addr);
    verify_host_name(he , verbose);

    //SOCKET()
    int client_socket_fd;
    int domain = PF_INET;
    int type = SOCK_STREAM;
    int protocol = 0;
    client_socket_fd = socket(domain, type, protocol);
    handle_system_call(client_socket_fd , "SOCKET()", verbose);

    //CONNECT()
    struct sockaddr_in server_ip_addr;
    server_ip_addr.sin_family = AF_INET;
    server_ip_addr.sin_port = htons(PORT);
    server_ip_addr.sin_addr = *((struct in_addr *)he->h_addr);
    memset(&(server_ip_addr.sin_zero), '\0', 8);
    system_call_result = connect(
        client_socket_fd , (struct sockaddr*)&server_ip_addr ,
        sizeof(struct sockaddr)
    );
}

```

```

    handle_system_call(system_call_result , "CONNECT()", verbose);

    //connect with server
    handle_connection_with_server(client_socket_fd , verbose);

    //CLOSE()
    close(client_socket_fd);

    return 0;
}

```

## 5. Conclusiones

Con el desarrollo de este proyecto se pudieron cumplir de manera eficiente los objetivos planteados: programar una arquitectura cliente servidor utilizando sockets. Si bien lo más difícil de este proyecto, en teoría, era la programación de los sockets, lo que más tiempo me tomó resolver fue la lectura y escritura de archivos pues el manejo de cadenas de texto en C no es muy transparente. Independientemente de esto, el proyecto fungió como una parte importante en el entendimiento de las arquitecturas cliente-servidor así como la programación de sockets en UNIX.

## Referencias

- [1] Zamitiz C. (2021). Apuntes de la materia Arquitectura Cliente Servidor. Facultad de Ingeniería. UNAM
- [2] Hall, B. (2019). Beej's Guide to Network Programming. Independently Published.