Anexo3: Código de la aplicación ClientePFC

```
/**
 * @clientepfc.c
 *
 * Sergio Bellido Sánchez
 *
 * Proyecto Final de Carrera
 *
 */

#include <stdio.h>
#include <netdb.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>

#include "constantes.h"
#include "funciones_cliente.h"

/*Aplicación cliente que se encarga de enviar los archivos necesarios para su ejecución al
servidor
* la forma de utilizar la aplicación sería:
* ./clientepfc LOGIN_USER
*/
int main(int argc, char *argv[])
{
        int descrip_local;
        int cuantos;
        int origen;
        int destino;
        int leidos;
        int k=1, h=0;

        char **cadenas;
        char buffer_fichero[100];
        char directorio [50]= "./../src/fp2/poo/";
        char concateno[100];
        char concateno1[100];
        char lomio[50];
        char respuesta[2];

         /*Reserva dinámica de memoria*/
        cadenas = (char **)malloc(sizeof(char *) * 50);
        *cadenas = (char *)malloc(sizeof(char));

        /*Inicialización a cero de las cadenas*/
        memset(concateno, '\0', strlen(concateno));
        memset(concateno1, '\0', strlen(concateno1));
        memset(lomio, '\0', strlen(lomio));
        memset(respuesta, '\0', strlen(respuesta));

        /*Se muestra la pantalla de inicio de la interfaz gráfica*/
        pantalla();

        /*Primera Comprobación: ¿Se ha escrito nombre usuario?*/
        if(argc!=2)
                error("\n\n\tNúmero de argumentos érroneos\n");

        /*Tomamos el nombre de los ficheros definidos en constantes.h*/
        cadenas[0] = argv[1];
        cadenas[1] = FICHERO1;
        cadenas[2] = FICHERO2;
        cadenas[3] = FICHERO3;
        cadenas[4] = FICHERO4;
        cadenas[5] = FICHERO5;
        cadenas[6] = FICHERO6;
        cadenas[7] = FICHERO7;
```

```
    /*Se concatenan cadenas para establecer la ruta hasta los ficheros*/
    strcat(lomio, "makefile");
    strcat(lomio, cadenas[0]);
    cadenas[8] = lomio;

    strcat(concateno, directorio);

    /*primeramente se hace una comprobación de que los archivos especificados
    * existen, si no fuera así, se cierra la aplicación
    */
    comprobacion(concateno, cadenas);

    /*creamos el socket mediante una función que devuelve el descriptor*/
    descrip_local=crea_socket();

    printf("\n\n\n\tSubiendo archivos para su procesamiento...\n");

     /*
      *Primeramente se envía el nombre de grupo, para que el servidor lo trate
      *individualmente
      */
    if((cuantos = send(descrip_local, cadenas[0], strlen(cadenas[0]), 0)) < 0)
            perror("\n send user");
    close(descrip_local);

    memset(concateno, '\0', strlen(concateno));
    strcat(concateno, "./../src/fp2/poo/");


    h=1;
    k=1;
/**
 * Primera rutina, que envía en orden todos los ficheros, empezando por su nombre
 * seguido de su contenido, haciendo uso de la conexión establecida con la herramienta
 * servidora e-Assessment. El número total de ficheros: TOTAL_ARGS
 */
    while(k < TOTAL_ARGS)
    {
            /*volvemos a crear un socket para el resto de ficheros*/
            descrip_local=crea_socket(destino);
            /*el primer fichero tiene un nombre y ruta diferentes*/
            if(h==1) {
                    strcat(concateno1, cadenas[1]);
                    strcat(concateno1, cadenas[0]);
                    strcat(concateno1, ".java");
                    if((cuantos = send(descrip_local,concateno1, strlen(concateno1), 0)) <0)
                    {
                            printf("fallo al enviar nombre de archivo %s\n", cadenas[h]);
                            exit(1);
                    }
                    memset(concateno1, '\0', strlen(concateno1));
            }
            /*enviamos el nombre del fichero si no es el primero*/
            else {
                    if((cuantos = send(descrip_local,cadenas[h], strlen(cadenas[h]), 0)) <0)
                    {
                            printf("fallo al enviar nombre de archivo %s\n", cadenas[h]);
                            exit(1);
                    }
            }
            memset(respuesta, '\0', strlen(respuesta));

            /*recivimos confirmación de recepción*/
            if((cuantos = recv(descrip_local, respuesta, 1, 0)) < 1)
                    error("\nError al recibir información de creación de fichero");

            /*El primer archivo en tomar será Principal.java*/
            if(h==1) {
                    strcat(concateno1, concateno);
                    strcat(concateno1, "utilidades/");
                    strcat(concateno1, cadenas[1]);
                    strcat(concateno1, cadenas[1]);
                    strcat(concateno1, ".java");
                    strcat(concateno1, cadenas[h]);
```

```
                }
                else {
                        /*El último archivo será el makefile del usuario*/
                        if(h==8) {
                                strcat(concateno1, "./../makefile");
                                strcat(concateno1, cadenas[0]);
                        }
                        /*el resto de archivos serán los alojados en pfpooxxx*/
                        else {
                                strcat(concateno1, "./../src/fp2/poo/pfpoo");
                                strcat(concateno1, cadenas[0]);
                                strcat(concateno1, "/");
                                strcat(concateno1, cadenas[h]);
                        }
                }
                /**
                * Abrimos el fichero para proceder a su lectura.
                * Tantos bytes leamos, tantos enviamos
                */
                origen = open(concateno1, O_RDONLY);
                memset(concateno1, '\0', strlen(concateno1));
                if(origen < 0)
                {
                        printf("\nError al leer archivo %s", cadenas[h]);
                        exit(1);
                }
                do
                {
                        /*leemos el fichero especificado y enviamos al servidor su contenido*/
                        if((leidos = read(origen, buffer_fichero, 100)) < 0)
                        {
                                printf("\n error al leer archivo origen %s", cadenas[h]);
                                exit(1);
                        }
                        if(leidos > 0)
                        {
                                cuantos = send(descrip_local, buffer_fichero, leidos, 0);
                                if(cuantos < 0)
                                        error("\nError al enviar contenido fichero");
                        }
                }
                while(leidos > 0);
                close(origen);
                memset(buffer_fichero,'\0',strlen(buffer_fichero));

                h++;
                k++;
                close(descrip_local);
        }
        /*Damos unos segundos al servidor para que pueda procesar los datos*/
        sleep(3);
        for(;;)
        {
                /**
                 * Creamos un nuevo socket y quedamos a la espera de la respuesta
                 * del servidor. Si todo ha sido correcto seguimos con el proceso
                 */
                descrip_local=crea_socket(destino);
                if((cuantos = recv(descrip_local, respuesta, 1, 0)) < 0)
                        error("\nError en confirmación de éxito");
                if((*respuesta)!='O')
                        printf("error respuesta");
                pantalla_opciones(descrip_local);
        }
        return 1;
}
```

```
/**
 * @constantes.h
 *
 * Sergio Bellido Sánchez
 *
 * Proyecto Final de Carrera
 *
 */


#ifndef CONSTANTES_H
#define CONSTANTES_H

#define TOTAL_ARGS 9
#define FICHERO1 "DatosDeEntrada"
#define FICHERO2 "Contacto.java"
#define FICHERO3 "Domicilio.java"
#define FICHERO4 "Telefono.java"
#define FICHERO5 "Persona.java"
#define FICHERO6 "CorreoElectronico.java"
#define FICHERO7 "Agenda.java"

#endif
```

```
/**
 * @funciones_cliente.h
 *
 * Sergio Bellido Sánchez
 *
 * Proyecto Final de Carrera
 *
 */


#ifndef FUNCIONES_CLIENTES_H
#define FUNCIONES_CLIENTES_H

void error(char *err);
void pantalla_opciones(int descrip_local);
void pantalla();
void opciones();
void comprobacion(char *concatenado, char **archivos);
int crea_socket();

#endif
```

```c
/**
 * @funciones_cliente.c
 *
 * Sergio Bellido Sánchez
 *
 * Proyecto Final de Carrera
 *
 */


#include <stdio.h>
#include <netdb.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>

#include "funciones_cliente.h"
#include "constantes.h"

/**
 * Método que muestra por pantalla la interfaz de usuario
 */
void pantalla()
{
        printf("\n\n\n\n\n\n\n\n\n\n\n\n");
        printf("\t*****************************************\n");
        printf("\t*        Sergio Bellido Sánchez         *\n");
        printf("\t*      Proyecto Final de Carrera        *\n");
        printf("\t*                                         *\n");
        printf("\t*        Ingeniería Telemática          *\n");
        printf("\t*    Programación Orientada a Objetos     *\n");
        printf("\t*                                         *\n");
        printf("\t*        Herramienta de Depuración      *\n");
        printf("\t*****************************************\n");

}

/**
 * Método que presenta al alumno por pantalla las opciones que puede
 * seleccionar para interactuar con la máquina servidora
 */
void opciones()
{
        printf("\n\n\n\n\tSeleccione una de las siguientes opciones:\n");
        printf("\t\t1.Compilar código\n");
        printf("\t\t2.Ejecutar pruebas\n");
        printf("\n\n\t\t0.Salir de la aplicación\n");
}

/*
 * Método que realiza la comprobación de existencia de archivos
 * Es un método importante, porque si no se encuentra algún archivo
 * el proceso cliente debe ser parado antes de que se comunique con
 * el servidor
 */
void comprobacion(char *concatenado, char **archivos)
{
        char concateno1[100];
        int k;
        int origen;
        k=1;
        while(k<TOTAL_ARGS) {
                if(k==1) {
                        strcat(concateno1, concatenado);
                        strcat(concateno1, "utilidades/");
                        strcat(concateno1, archivos[k]);
                        strcat(concateno1, cadenas[0]);
                        strcat(concateno1, ".java");
                        origen = open(concateno1, O_RDONLY);
                        if(origen < 0) {
```

```
                                printf("\n\tEl archivo %s no se encuentra en %s\n", archivos[k],
                                                                                concateno1);
                                exit(1);
                        }
                        close(origen);
                }
                else {
                        if(k==8) {
                                strcat(concateno1, "./../makefile");
                                strcat(concateno1, archivos[0]);
                                origen = open(concateno1, O_RDONLY);
                                if(origen < 0) {
                                        printf("\n\tEl archivo %s no se encuentra en %s\n",
                                                                        archivos[k], concateno1);
                                        exit(1);
                                }
                                close(origen);
                        }
                        else {
                                strcat(concateno1, concatenado);
                                strcat(concateno1, "pfpoo");
                                strcat(concateno1, archivos[0]);
                                strcat(concateno1, "/");
                                strcat(concateno1, archivos[k]);
                                origen = open(concateno1, O_RDONLY);
                                if(origen < 0) {
                                        printf("\n\tEl archivo %s no se encuentra en %s\n",
                                                                        archivos[k], concateno1);
                                        exit(1);
                                }
                                close(origen);
                        }
                }
                memset(concateno1, '\0', strlen(concateno1));
                k++;
        }
}

/**
 * Método que trabaja con las opciones que el usuario ha elegido de cara a
 * interactuar con el servidor, dependiendo de la opción elegida la rutina
 * tomará distinto camino
 */
void pantalla_opciones(int descrip_local)
{

        char cad[100];
        char cad_a[100];
        char concatenacion[100];
        char buffer[100];
        char almacen[10000];
        char respuesta[2];
        int j=0, i=0;
        int cuantos;
        int origen;
        int leidos;
        int escritos;
        int destino;

         /*Limpiamos las cadenas de caracteres espúreos*/
        memset(cad, '\0', strlen(cad));
        memset(cad_a, '\0', strlen(cad_a));
        memset(concatenacion, '\0', strlen(concatenacion));

        /*Presentamos la interfaz gráfica al usuario*/
        pantalla();
        opciones();

        /*Tomamos los datos que ha elegido el usuario*/
        do {
                cad[j]=getchar();
                j++;
        } while(cad[j-1]!='\n');
        cad[j]='\0';
```

```
 /*Si la respuesta es 1, el usuario quiere compilar su código*/
if(cad[0] == '1') {
        printf("\n\n\tEspere mientras se compila su código, puede tardar varios
                                                                minutos...\n\n");

        /*mandamos la respuesta al servidor*/
        if((cuantos = send(descrip_local, cad, strlen(cad), 0)) < 0)
                error("\n\tError al enviar opción 1\n\n");
        sleep(3);

        /*
         * Recibimos la respuesta del servidor, si todo ha sido correcto
         * se avisa al alumno de tal hecho. En caso de fallo en el código,
         * se recibe el archivo de erroes generados en el servidor, y se
         * le pasa al usuario para que pueda trabajar con él y corregir
         * los errores que se hayan cometido
         */
        if((cuantos = recv(descrip_local, respuesta, 1, 0)) < 1)
                error("\n\terror al recibir la respuesta de compilación\n\n");
        if(respuesta[0]=='O') {
                close(descrip_local);
                printf("\n\n\n\n#####################################\n");
                printf("\n\n\tOperación realizada con éxito\n");
                printf("\n\n\tHa superado la prueba con éxito.\n");
                printf("\n\n#####################################\n\n\n\n\n");
                exit(1);
        } else {
                destino = open("salidacompile", O_CREAT | O_WRONLY | O_TRUNC, 0664);
                if(destino ==-1)
                        error("Error abrir salida compilación compilexxx\n\n");
                if((cuantos = recv(descrip_local, almacen, 10000, 0)) < 1)
                        error("Error al recibir el archivo de compilación\n\n");
                escritos = write(destino, almacen, cuantos);
                if(escritos < 0)
                        error("error al escribir en el archivo de compilación\n\n");
                close(destino);
                pantalla();

                printf("\n\\n\n\n\tIncorrecto \n");
                printf("\n\tError al compilar el código, Revise los ficheros de error
                          que se han instalado en su directorio de ejecución\n\n\n");
                close(descrip_local);
                exit(1);
        }
}

/**
 * Si la respuesta es 2, el alumno quiere ejecutar algunas de las pruebas
 * Deberá indicar primeramente el nombre del archivo de entrada que quiere
 * así como el nombre del archivo principal que contiene el método main() y
 * que ha sido diseñado para pasar la prueba específica
 */
else if(cad[0] == '2') {
        printf("\n\tIndique el nombre del argumento a enviar\n");
        do {
                cad[j]=getchar();
                cad_a[i]=cad[j];
                j++;
                i++;
        } while(cad[j-1]!='\n');
        cad[j]='\0';
        cad_a[i-1]='\0';

        if((cuantos = send(descrip_local, cad, strlen(cad), 0)) < 0)
                error("\n\tError al enviar opción 2\n\n");
        if((cuantos = recv(descrip_local, respuesta, 1, 0)) < 1)
                error("\n\tError al recibir respuesta sobre ejecución\n\n");

        strcat(concatenacion, "./../src/fp2/poo/datos/");
        strcat(concatenacion, cad_a);

        origen = open(concatenacion, O_RDONLY);
        if(origen < 0)
```

```
                error("error al abrir datos entrada\n");

        printf("\n\n\tEspere mientras se ejecuta su código,
                                        puede tardar varios segundos...\n\n");

        do {
                if((leidos = read(origen, buffer, 100)) < 0)
                        error("\n\tError al leer los datos de entrada para la
                                                                ejecución\n");
                cuantos = send(descrip_local, buffer, leidos, 0);
                if(cuantos < 0)
                        error("\n\tError al enviar datos de entrada\n\n");
        } while(leidos > 0);
        if((cuantos = recv(descrip_local, respuesta, 1, 0)) < 1)
                error("\n\tError al recibir respuesta ejecución");
        close(origen);

        memset(concatenacion, '\0', strlen(concatenacion));
        memset(cad, '\0', strlen(cad));
        memset(cad_a, '\0', strlen(cad_a));
        printf("\n\tIndique el nombre del archivo principal a enviar\n");
        printf("\n\tRecuerde, si envía agenda05, ahora debe escribir
                                                        Principal05.java\n");
        j=0;
        i=0;
        do {
                cad[j]=getchar();
                cad_a[i]=cad[j];
                j++;
                i++;
        } while(cad[j-1]!='\n');
        cad[j]='\0';
        cad_a[i-1]='\0';

        if((cuantos = send(descrip_local, cad, strlen(cad), 0)) < 0)
                error("\n\tError al enviar opción 2\n\n");
        if((cuantos = recv(descrip_local, respuesta, 1, 0)) < 1)
                error("\n\tError al recibir respuesta sobre ejecución\n\n");

        strcat(concatenacion, "./../src/fp2/poo/principal/");
        strcat(concatenacion, cad_a);

        origen = open(concatenacion, O_RDONLY);
        if(origen < 0)
                error("error al abrir datos entrada\n");
        printf("\n\n\tEspere mientras se ejecuta su código, puede tardar varios
                                                                segundos...\n\n");

        do {
                if((leidos = read(origen, buffer, 100)) < 0)
                        error("\n\tError al leer los datos de entrada para la
                                                                ejecución\n");
                cuantos = send(descrip_local, buffer, leidos, 0);
                if(cuantos < 0)
                        error("\n\tError al enviar datos de entrada\n\n");
                if(cuantos < 100)
                        break;
        } while(leidos > 0);

/*
 * Quedamos a la espera de respuesta por parte del servidor, y en caso de ser
 * un OK, se muestra al alumno un mensaje de éxito. En caso contrario, se recibirá
 * un archivo, que dependiendo dónde estuvo el fallo será un archivo generado durante
 * la compilación, o un archivo generado por el comando diff de comparación
 * de ficheros
 */
        if((cuantos = recv(descrip_local, respuesta, 1, 0)) < 1)
                error("\n\tError al recibir respuesta ejecución");
        close(origen);

        if(respuesta[0]=='O') {
                close(descrip_local);
                printf("\n\n\n\n\n\n#####################################\n");
                printf("\n\n\tOperación realizada con Éxito\n");
```

```
                        printf("\n\tHa superado la prueba con éxito.\n");
                        printf("\n\n########################################\n\n\n\n\n");
                        exit(1);
                }
                else {
                        destino = open("salidadiff", O_CREAT | O_WRONLY | O_TRUNC, 0664);
                        if(destino ==-1)
                                error("\n\tError abrir salida ejecución, archivo diff\n\n");
                        if((cuantos = recv(descrip_local, almacen, 10000, 0)) < 1)
                                error("\n\tError al recibir los datos del fichero de salida de
                                                                errores de la ejecución\n\n");
                        escritos = write(destino, almacen, cuantos);
                        if(escritos < 0)
                                error("\n\tError al escribir en el fichero de errores de la
                                                                        ejecución\n\n");
                        close(destino);

                        pantalla();
                        printf("\n\n\n\n\n\tIncorrecto\n");
                        printf("\n\tError al ejecutar las pruebas, Revise los ficheros de error
                                                que se han instalado en su directorio\n\n\n");
                        close(descrip_local);
                        exit(1);
                }

    /*Si no se eligió ni 1 ni 2, entonces es que el alumno quiere salir de la aplicación*/
        } else {
                printf("\n\n\n\n\tFinal de la aplicación\n\n\n\n\n");
                cad[0]='0';
                if((cuantos = send(descrip_local, cad, strlen(cad), 0)) < 0)
                        error("\nError al enviar elección\n\n");
                close(descrip_local);
                exit(1);
        }
}
/*Método para crear el socket para la conexión con server y devolvemos su descriptor*/
int crea_socket()
{
        struct sockaddr_in destino;
        struct servent *puerto;
        struct hostent *direc=NULL;
        int descrito;

        /*Indicamos que vamos a utilizar direcciones de internet*/
        destino.sin_family = AF_INET;

        /*Se pasa la dirección de la máquina destino*/
        int dir = inet_aton("10.0.2.15", &(destino.sin_addr));

        /*Le asignamos el puerto de escucha*/
        if((puerto = getservbyname("pfc","tcp")) == NULL)
        {
                printf("error funcion tomar puerto");
                exit(1);
        }
        destino.sin_port = puerto->s_port;
        memset((&destino.sin_zero), '\0', 8);

        /*creamos el socket y conectamos con el servidor*/
        if((descrito=socket(AF_INET,SOCK_STREAM, 0))<0)
                error("socked");
        if(connect(descrito, (struct sockaddr *)&destino, sizeof(struct sockaddr)))
                error("connect");
        return descrito;
}
/*
 * Método creado para mostrar por pantallas mensajes de error internos del programa
 * y salir del mismo en su caso
 */
void error(char *err)
{
        perror(err);
        exit(1);
}
```