INSTITUO EDUCACIÓN SECUNDARIA ÁGORA

GRADO SUPERIOR DE ADMINISTRACIÓN DE SISTEMAS INFORMÁTICOS

PROYECTO:
COMPARTICIÓN DE LIBROS DIGITALES

1.- MANUAL DEL PROGRAMADOR

PROYECTO FIN DE CURSO
REALIZADO POR: TAMARA RUFO POZO
ASIGNATURA: DFSI
Cáceres a 1 de Marzo de 2011

Tabla de contenido

Apéndice A

Manual del programador

A.1. Introducción

El propósito de este manual del programador es dar a conocer al lector todos los listados del programa realizado. Para ello se tratará de forma amena y concisa un repaso de todos los *ficheros Include, ejecutables...*, con el fin de que cualquier desarrollador pueda modificar a su gusto algunos de los valores y parámetros de las funciones expuestas.

A.2. Arquitectura

La herramienta se encuentra dividida en cuatro partes bien diferenciadas. Por un lado tenemos el programa servidor, el cual podemos considerar la herramienta principal, por otra parte encontramos el programa cliente, que se ocupará del registro y acceso de los usuarios al servidor, por otra el programa cola, que se va a ocupar de recoger de una cola los libros enviados por el proveedor y enviarlos al servidor, mediante sockets, y por último encontramos el programa proveedor, el cual se va a ocupar de proporcionar los libros al servidor.

A.2.1. Estructura Servidor

Está aplicación permite crear y borrar temas y subtemas, también permite borrar libros. Esta aplicación almacenará en el lugar adecuado cada libro que recibe de un proveedor. Los libros se almacenan en el servidor en una estructura de directorios que coincide con los temas, subtemas… de los libros.

Como acabamos de mencionar la herramienta principal se arrancará en el computador que haga las funciones de punto de control. Los archivos que necesitamos para ello son:

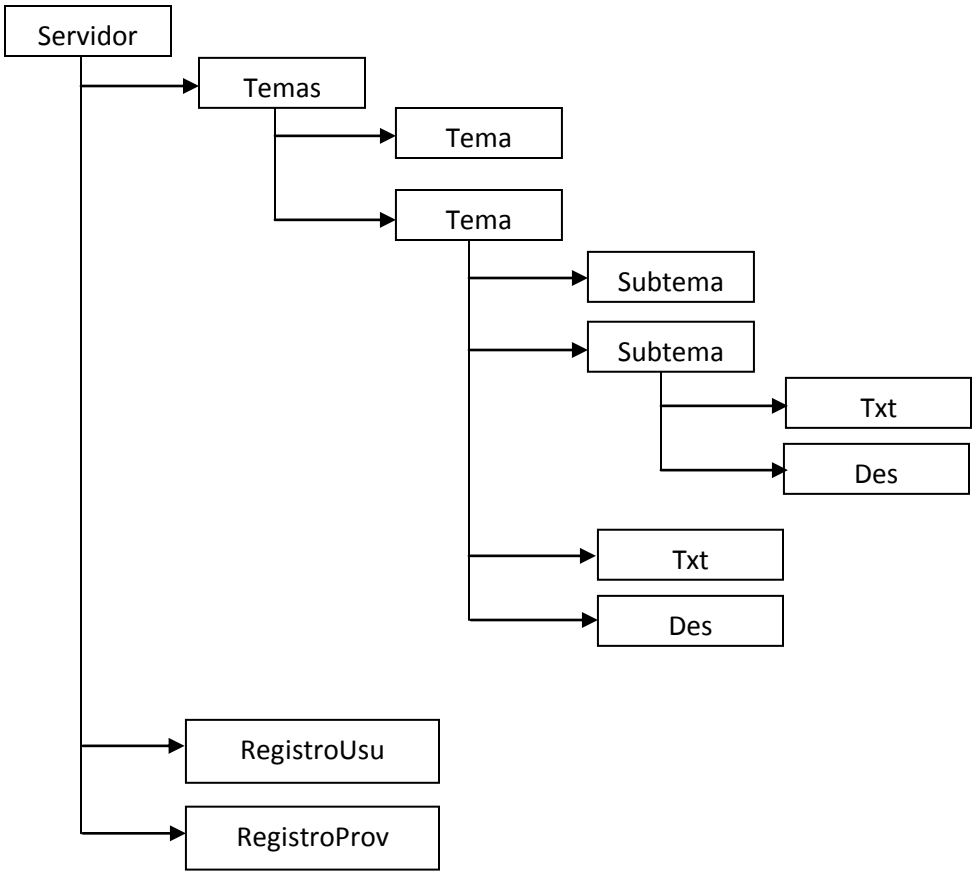
- *servidor.c*: Contiene el código fuente.
- *milibreria.h*: Librería común o todos los archivos *".c"*, y contiene todas las librerías necesarias para la compilación de los códigos fuente.
- *servidor*: Archivo ejecutable de la aplicación.

A.2.1.1 Funciones

- ***int main(void);*** Función principal.
- ***void crearToS(void);*** Crear Temas y Subtemas.
- ***void creartema(char ruta[256]);*** Crear tema.
- ***void crearsubtema(char ruta[256]);*** Crear subtema.
- ***void borrarToS(void);*** Borrar Temas y Subtemas.
- ***void borrartema(char ruta[256]);*** Borrar tema.
- ***void borrarsubtema(char ruta[256]);*** Borrar subtema.
- ***int listar(char ruta[256]);*** Listar contenido del directorio.
- ***void blibro(void);*** Borrar libro.
- ***void BuscaYborra(char tema[20],char ruta[255]);*** Función para buscar y borrar un libro en concreto.
- ***void crear (char ruta[256]);*** Función llamada por creartema y crearsub para crearlos.
- ***void borrar(char ruta[256]);*** Función llamada por borrartema y borrarsubtema para borrarlos.
- ***void busqueda(char tema[20]);*** Función llamada por "*blibro*" para encontrar los libros disponibles en el servidor.
- ***void funcionsocketP(void);*** Función llamada por main, para gestionar los mensajes de socket que llegan del proveedor, con respecto a los usuarios.
- ***void funcionsocketU(void);*** Función llamada por main, para gestionar los mensajes de socket que llegan del cliente, con respecto a los usuarios.
- ***void funcionsocketLibrosU(void);*** Función llamada por main, para gestionar los mensajes de socket que llegan del cliente, con respecto a la descarga de libros.
- ***void funcionsocketLibrosP(void);*** Función llamada por main, para gestionar los mensajes de socket que llegan del proveedor, con respecto al envio de libros.
- ***int compruebausu(char usuario[20],char contra[10]);*** Función llamada por la función "*funcionsocketU*" para comprobar si el usuario existe en el sistema.
- ***int compruebauprov(char proveedor[20],char contra[10],char usuario[20],char contrau[10]);*** Función llamada por la función "*funcionsocketP*" para comprobar si un usuario de un proveedor existe en el sistema.
- ***int compruebaprov(char proveedor[20],char contra[10]);*** Función llamada por la función "*funcionsocketP*" para comprobar si un proveedor existe en el sistema.
- ***void borrarusu(char usuario[20],char contra[10]);*** Función llamada por la función "*funcionsocketU*" para eliminar a un usuario del sistema.
- ***void borraruprov(char proveedor[20],char contra[10],char usuario[20],char contrau[10]);*** Función llamada por la función "*funcionsockeP*" para eliminar a un usuario del proveedor del sistema.
- ***void borrarprov(char proveedor[20],char contra[10]);*** Función llamada por la función "*funcionsockeP*" para eliminar a un proveedor del sistema.
- ***void guardaLibro(COMANDOLIBROPROV clibro);*** Función llamada por la función "*funcionsocketLibrosP*" para guardar un libro en el sistema.
- ***void partir(char *original, char * parte1, char * parte2);*** Función llamada por la función "*funcionsocketLibrosP*" para separar directorios de una ruta.

- ***int crearListalibros (char lista[500],char ruta[255]);*** Función llamada por la función "*funcionsocketLibroU*" para mandar un listado de los libros que hay en el sistema, al usuario.
- ***int buscaLibro(char libro[50]);*** Función llamada por la función "*funcionsocketLibrosU*" para buscar el libro que queremos descargar del sistema.
- ***char* substr (char* cadena, int comienzo, int longitud);*** Función llamada por la función "*funcionsocketLibrosP*" para separar el nombre del archivo de su extensión.
- ***void listaservidor (void);*** Función llamada por la función "*blibro*" para ver un listado de libros disponibles en el servidor.

A.2.1.2 Estructura Directorios



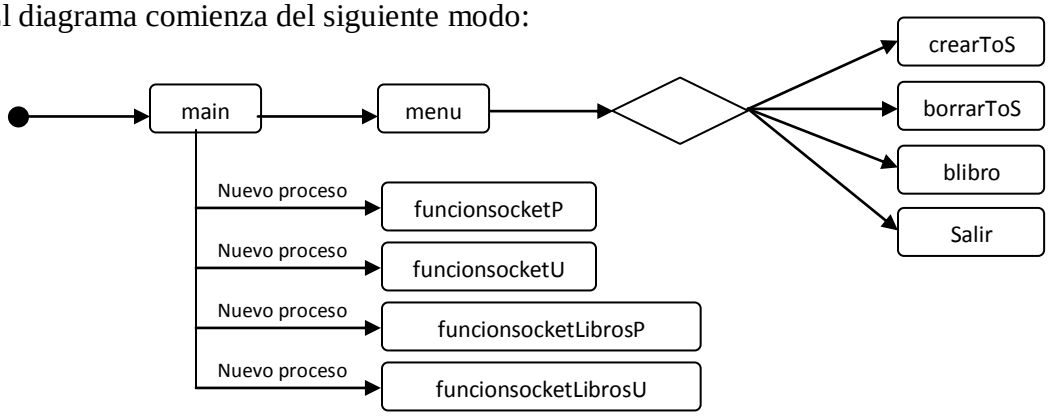
A.2.1.3 Funcionamiento Servidor

En primer lugar la función *"main()"* creará la estructura de directorios básica, compuesta por la carpeta contenedora de todas las demás, llamada "Servidor", y dentro de ésta,

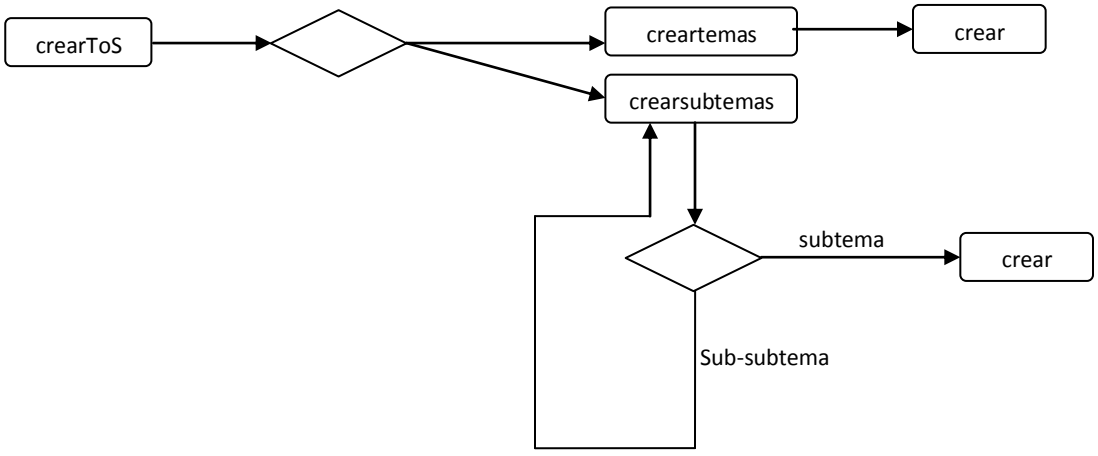
"Temas", "Usuarios" y "Proveedores". A continuación muestro con un diagrama como va avanzando el programa por las funciones programadas. La leyenda para este tipo de diagramas será la siguiente:

●	Comienzo del programa
→	Transiciones de una función a otra.
▭	Uso de una función
◇	Decisión de una opción

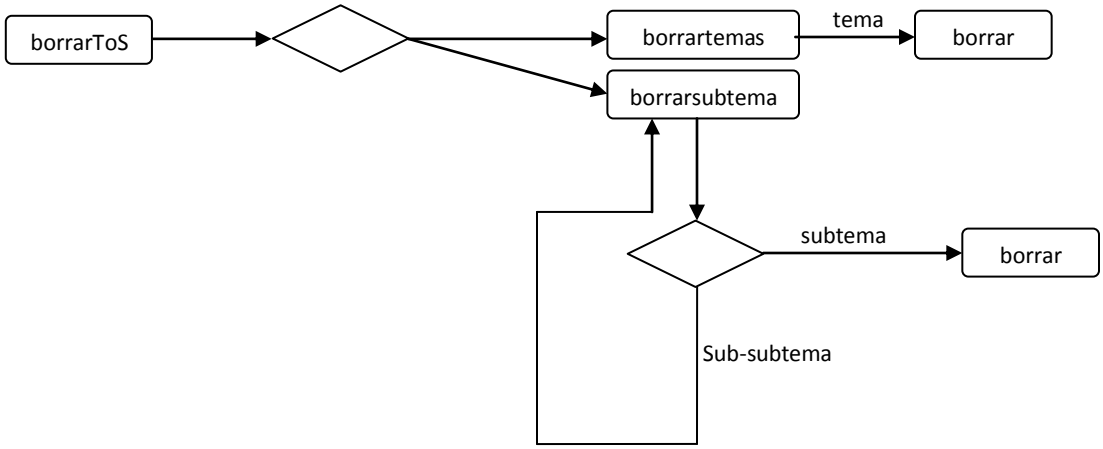
El diagrama comienza del siguiente modo:



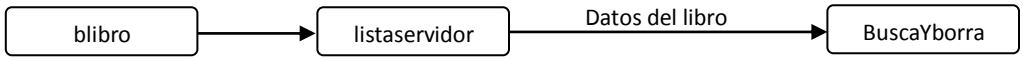
1. Si elegimos la opción "Crear Tema" o Subtema el diagrama continúa de este modo:



2. Si elegimos la opción “Borrar Tema o Subtema” el diagrama continúa del siguiente modo:



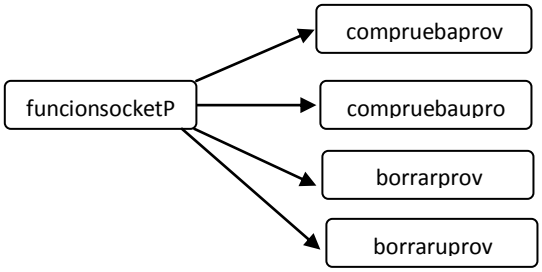
3. Si elegimos la opción “Borrar libro” el diagrama continúa del siguiente modo:



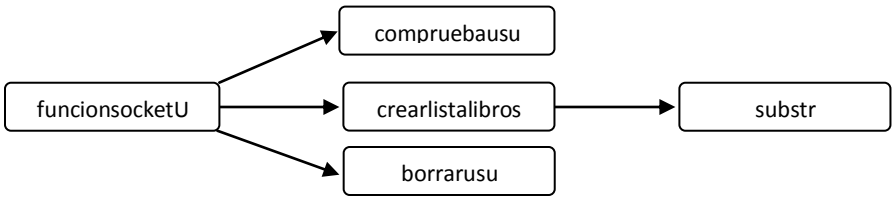
Para mayor claridad de los diagramas, e ignorado la función *“listar”*, la cual se ocupa de irnos mostrando el contenido de todos los directorios.

Funciones de Sockets:

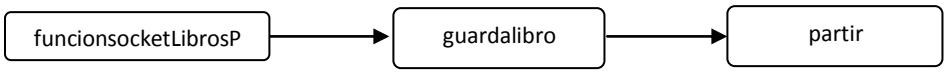
1. Función de Socket que va a recoger los datos de los usuarios del proveedor.



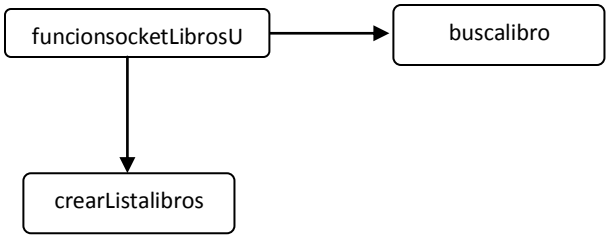
2. Función de Socket que va recoger los datos de los usuarios del cliente.



3. Función de Socket que va a recoger los datos de los libros que envía el proveedor al servidor.



4. Función de Socket que va a recoger los datos de los libros que va a pedir el cliente para descargarse.



A.2.2. Estructura Cliente

Podrá descargarse un libro del sistema y almacenar hasta 5 de ellos. La aplicación permitirá ver la descripción de un libro, leerlo y borrarlo. Los libros descargados se almacenan en la carpeta 'librosNombreAlumno' donde NombreAlumno es el nombre del usuario con el que se accede.

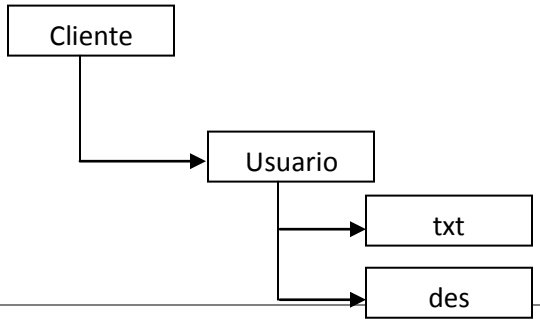
Esta aplicación se arrancará en el cualquier equipo que pueda acceder al servidor, o en su defecto en el mismo equipo del servidor. Los archivos que necesitamos para ello son:

- *cliente.c*: Contiene el código fuente.
- *milibreria.h*: Librería común o todos los archivos *".c"*, y contiene todas las librerías necesarias para la compilación de los códigos fuente.
- *cliente*: Archivo ejecutable de la aplicación.

A.2.2.1 Funciones

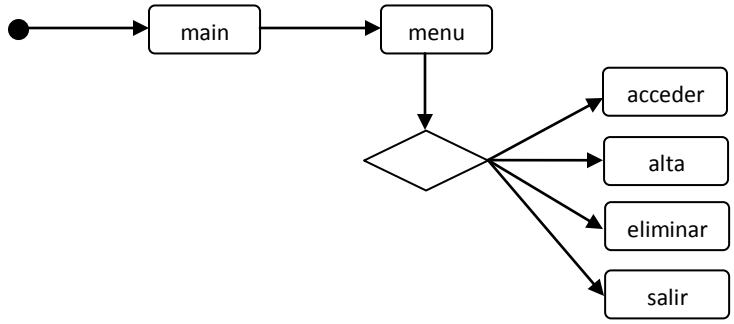
- ***main().***Función principal.
- ***void menu (char servidor[20]);*** Función menu principal.
- ***void acceder(char servidor[20]);*** Acceder usuario.
- ***void submenu(char servidor[20],char usuario[20],char contra[10]);*** Submenu de usuario.
- ***void disponibles(char ruta[256]);*** Función para comprobar los libros que están disponibles en el servidor.
- ***void leerlibro(void);*** Función para leer un libro que tenemos en local.
- ***void eliminarlibro(char usuario[20]);*** Función que va a eliminar un libro que tenemos en local.
- ***void alta(char servidor[20]);*** Función que va a dar de alta a un usuario en el servidor.
- ***void eliminar(char servidor[20]);*** Función que va a dar de baja a un usuario de del servidor.
- ***void borrar (char ruta[256]);*** Función que va a borrar la estructura de directorios que tiene el usuario, cuando se dé de baja en el sistema.
- ***int comandoservidor(char servidor[20],char comando[20],char usuario[20],char contra[10]);*** Función que va a ser llamada cuando ejecutemos la opción dar de alta, para enviar un mensaje al servidor, para que haga lo pertinente.
- ***int comandoservidorDescargar(char servidor[20],char comando[20],char libro[50],char usuario[20]);*** Función que va a ser llamada cuando ejecutemos la opción disponibles, para enviar un mensaje al servidor, para descargarnos un libro.
- ***int comandoservidorLista(char servidor[20],char comando[20],char lista[500]);*** Función que va a ser llamada cuando ejecutemos la opción disponibles, para enviar un mensaje al servidor, para que nos mande una lista de los libros disponibles en el servidor.
- ***int comprobardir(char usuario[20]);*** Función que nos va a comprobar cuántos libros tenemos en nuestro directorio.
- ***int listar (char ruta[256]);*** Función que nos va a listar el contenido de nuestro directorio.
- ***void partir(char *original, char * parte1, char * parte2);*** Función para separar directorios de una ruta.
- ***void guardaLibro(COMANDOLIBROUSU clibro,char usuario[20]);*** Función que va a guardar un libro en el sistema.

A.2.2.2 Estructura Directorios

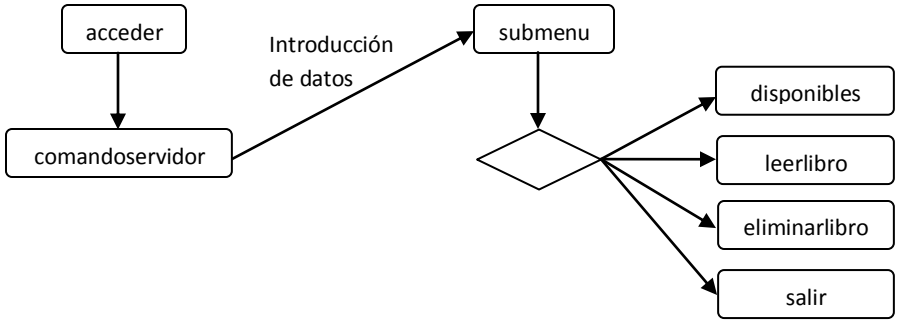


A.2.2.3 Funcionamiento Cliente

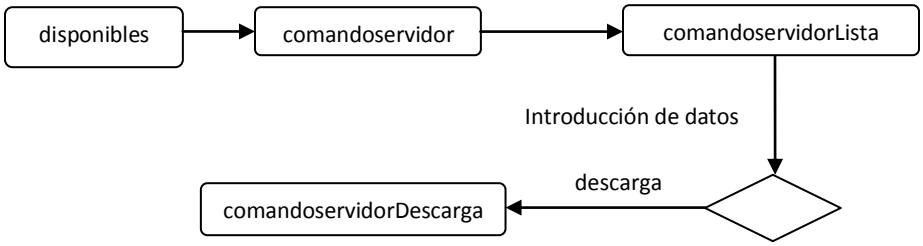
En primer lugar la función *"main()"* creará la carpeta "Cliente", en la que irá incluida nuestra estructura de directorios. A continuación se muestra con un diagrama como va avanzando el programa por las funciones programadas.



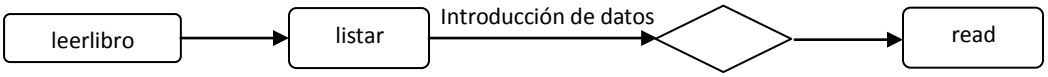
1. Si elegimos la opción de acceder, el diagrama continuaría del siguiente modo:



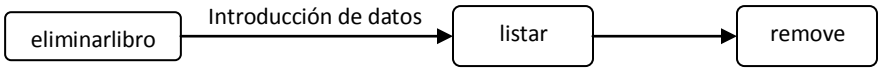
1.1. Si elegimos la opción visualizar los libros disponibles en el sistema, el diagrama continuaría del siguiente modo:



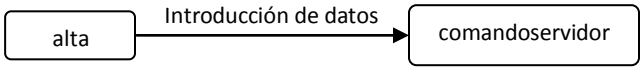
1.2. Si elegimos la opción de leer uno de nuestros libros, el diagrama continuaría del siguiente modo:



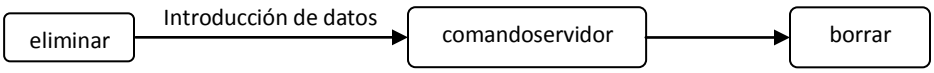
1.3. Si elegimos la opción de eliminar uno de nuestros libros, el diagrama continuaría del siguiente modo:



2. Si elegimos la opción dar de alta un usuario, el diagrama continuaría del siguiente modo:



3. Si elegimos la opción eliminar usuario, el diagrama continuaría del siguiente modo:



A.2.3. Estructura Proveedor

Cada proveedor dispondrá de una aplicación que podrá ejecutar los distintos usuarios dentro de un mismo sistema, un proceso se encargará de mostrar la cantidad total de libros aportados a nuestro sistema por ese proveedor y tendrá, tantas instancias como usuarios, de otro programa o proceso que se encargará de la entrada de datos de un nuevo libro. Estos se añadirán a una cola de mensajes. El proceso que muestra la cantidad total de los libros aportados enviará cada 10 minutos los libros que hay en la cola al servidor.

Esta aplicación se arrancará en el cualquier equipo que pueda acceder al servidor, o en su defecto en el mismo equipo del servidor. Los archivos que necesitamos para ello son:

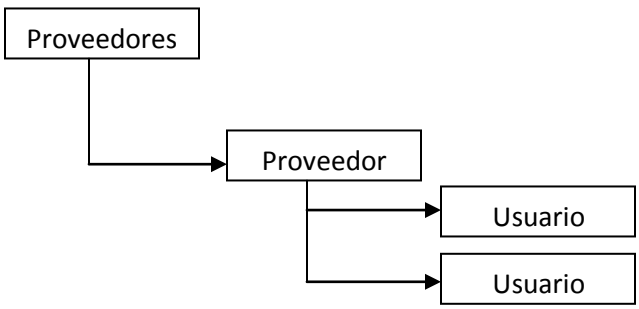
- *proveedor.c*: Contiene el código fuente.

- *milibreria.h*: Librería común o todos los archivos *".c"*, y contiene todas las librerías necesarias para la compilación de los códigos fuente.
- *proveedor*: Archivo ejecutable de la aplicación.

A.2.3.1 Funciones

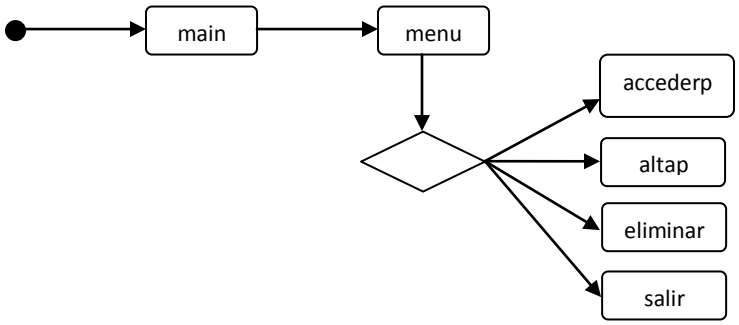
- ***void menu (char servidor[20]);*** menú principal.
- ***void accederp(char servidor[20]);*** Función que será llamada por el menú principal y hará una llamada al servidor para acceder como proveedor.
- ***void accederuprov(char servidor[20],char proveedor[20],char contra[10]);*** Función que será llamada por el submenú de proveedor y hará una llamada al servidor para acceder como usuario de proveedor.
- ***int comandoservidorp(char servidor[20],char comando[20],char proveedor[20],char contra[10],char usuario[20],char contrau[10]);*** Función que será llamada cuando haya que hacer una comprobación al servidor.
- ***void submenu(char servidor[20],char prov[20],char contra[10]);*** menu de provee dor.
- ***void altaprov(char servidor[20]);*** Función de alta de proveedor.
- ***void altauprov(char servidor[20],char prov[20], char contra[10]);*** Función de alta de usuario.
- ***void eliminarprov(char servidor[20]);*** Función de eliminar proveedor.
- ***void eliminaruprov(char servidor[20],char proveedor[20],char contra[10]);*** Función de eliminar usuario de un proveedor.
- ***void borrarDir(char ruta[256]);*** borrar cualquier directorio.
- ***void intro(char prov[20],char usuario[20]);*** Función para introducir libro, el cual va a ser enviado a una cola de mensajes.

A.2.3.2 Estructura Directorios

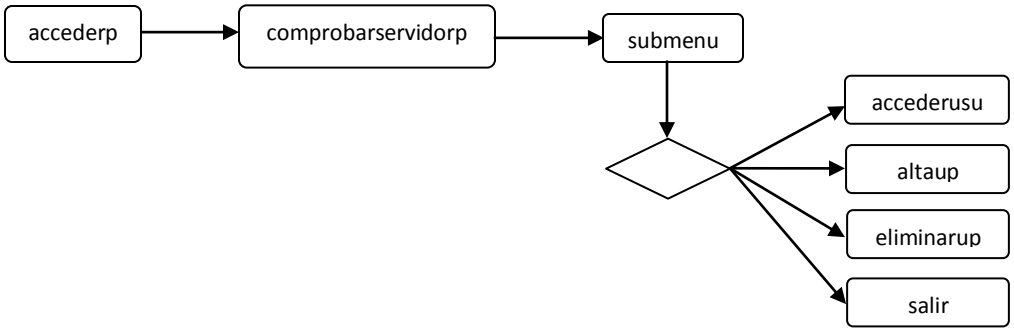


A.2.3.3 Funcionamiento Proveedor

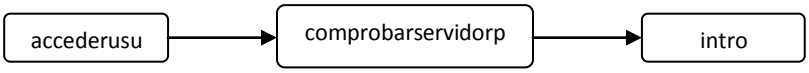
En primer lugar la función *"main()"* creará la carpeta "Proveedores", en la que irá incluida nuestra estructura de directorios. A continuación se muestra con un diagrama como va avanzando el programa por las funciones programadas.



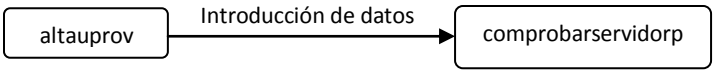
1. Si elegimos la opción acceder con un proveedor, el diagrama continuaría del siguiente modo:



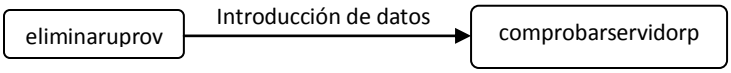
1.1. Si elegimos la opción acceder con un usuario, el diagrama continuaría del siguiente modo:



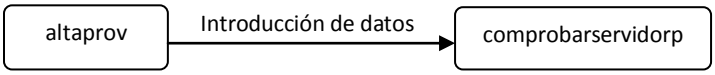
1.2. Si elegimos la opción "altaprov", el diagrama continuaría del siguiente modo:



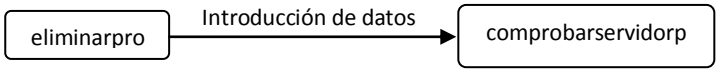
1.3. Si elegimos la opción "eliminarprov", el diagrama continuaría del siguiente modo:



2. Si elegimos la opción dar de alta a un proveedor, el diagrama continuaría del siguiente modo:



3. Si elegimos la opción eliminiar a un proveedor, el diagrama continuaría del siguiente modo:



A.2.4 Estructura Cola

Este programa se va a ejecutar conjuntamente con el proveedor, ya que se va a ocupar de leer la cola de mensajes, y al mismo tiempo, los mensajes que reciba los mandará a través de socket al proveedor.

Esta aplicación se arrancará en el cualquier en el mismo equipo donde se ejecute el proveedor. Los archivos que necesitamos para ello son:

- *cola.c*: Contiene el código fuente.
- *milibreria.h*: Librería común o todos los archivos *".c"*, y contiene todas las librerías necesarias para la compilación de los códigos fuente.
- *cola*: Archivo ejecutable de la aplicación.

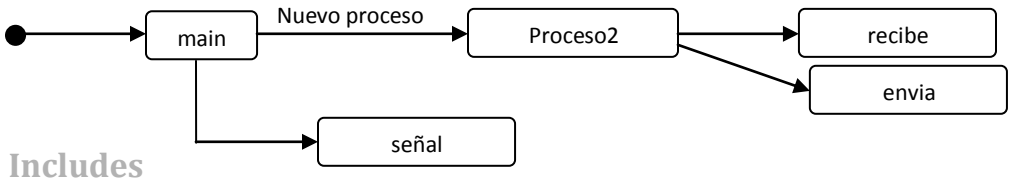
A.2.4.1 Funciones

- ***void proceso2(char servidor[20]);*** Función que recoge los mensajes de la cola y los manda por sockets.
- ***void minutos(int s); Función*** para controlar el tiempo de espera.

A.2.4.2 Funcionamiento Proveedor

En primer lugar la función *"main()"* creará un proceso nuevo, para que se ocupe

constantemente de estar escuchado la cola de mensajes. A continuación se muestra con un diagrama como va avanzando el programa por las funciones programadas.



A.2.5 Includes

Se declaran en un archivo llamado milibreria.h mediante la sentencia *#include<nombre>*. Son las siguientes:

- stdio.h
- stdlib.h
- unistd.h
- dirent.h
- string.h
- sys/stat.h
- sys/ipc.h
- sys/shm.h
- sys/wait.h
- signal.h
- time.h
- sys/types.h
- fcntl.h
- curses.h
- sys/msg.h
- sys/socket.h
- netinet/in.h
- arpa/inet.h
- strings.h
- netdb.h

A.2.6 Estructuras utilizadas

typedef struct{
char nombre[20];
char contra[10];
}USUARIO; Estructura para gestionar usuarios.

typedef struct{
int valor;
}RESPUESTA; Estructura para gestionar las respuestas del servidor.

typedef struct{
char proveedor[20];
char contra[10];
char nombre[20];
char contrau[10];
}PROV; Estructura para gestionar usuarios proveedores y proveedores.

typedef struct{
int enviados;
}MEMO; Estructura para gestionar la memoria compartida.

typedef struct{
char comando[20];
char nombre[20];
char contrau[10];
}COMANDOUSU; Estructura para gestionar la comprobación de usuarios en el servidor.

typedef struct{
char comando[20];
char proveedor[20];
char contra[10];
char nombre[20];
char contrau[10];
}COMANDOPROV; Estructura para gestionar la comprobación de usuarios proveedores y usuarios de proveedores en el servidor.

typedef struct{
char titulo[50];
char autor[50];
char adq[10];
char prov[20];
char tema[50];
int pag;
int isbn;
int desc;
}DESC; Estructura para gestionar la descripción de un libro.

typedef struct{
char texto[40];
}LIBRO; Estructura para gestionar el texto de un libro.

typedef struct {
long tipo;
char comando[20];
char titulo[50];
char autor[50];

char adq[10];
char prov[20];
char tema[50];
int pag;
int isbn;
int desc;
char texto[40];
}COMANDOLIBROPROV; Estructura para gestionar el envió de un libro al servidor.

typedef struct {
char comando[20];
char lista[500];
}COMANDOLISTALIBROUSU; Estructura para gestionar las listas de libros que se le van a ofrecer al cliente.

typedef struct {
char comando[20];
char titulo[50];
char autor[50];
char adq[10];
char prov[20];
char tema[50];
int pag;
int isbn;
int desc;
char texto[40];
}COMANDOLIBROUSU; Estructura para gestionar los libros que se van a enviar al cliente para su descarga.

A.3 Código de la herramienta

En este apartado se van a presentar los códigos fuentes del proyecto, para su estudio.

A.3.1. Código Servidor

```
#include "milibreria.h"
void menu(void);
void crearToS(void);
void creartema(char ruta[256]);
void crearsubtema(char ruta[256]);
void borrarToS(void);
void borrartema(char ruta[256]);
void borrarsubtema(char ruta[256]);
void blibro(void);
void buscaYborra(char tema[20], char ruta[255]);
void funcionsocketP(void);
void funcionsocketU(void);
void funcionsocketLibrosU(void);
```

```
void funcionsocketLibrosP(void);
int compruebausu(char usuario[20],char contra[10]);
int compruebauprov(char proveedor[20],char contra[10],char
usuario[20],char contrau[10]);
int compruebaprov(char proveedor[20],char contra[10]);
int listar(char ruta[256]);
void crear (char ruta[256]);
void borrar(char ruta[256]);
void borrarusu(char usuario[20],char contra[10]);
void borraruprov(char proveedor[20],char contra[10],char usuario[20],char
contrau[10]);
void borrarprov(char proveedor[20],char contra[10]);
void guardaLibro(COMANDOLIBROPROV clibro);
void partir(char *original, char * parte1, char * parte2);
int crearListalibros (char lista[500],char ruta[255]);
COMANDOLIBROUSU buscaLibro(char libro[50],char ruta[255]);
char* substr (char* cadena, int comienzo, int longitud);
void listaservidor (char ruta[255]);

/* Parametros de entrada: void.
 * Función: principal
 * Parametro de salida: 0
 * */
int main()
{
      int i=1,j=1,k=1,l=1,fu,fp;
      mkdir("./Servidor",0700);
      mkdir("./Servidor/Temas",0700);
      fu=open("./Servidor/registrousu",O_CREAT|O_RDWR|O_APPEND,0777);
      fp=open("./Servidor/registroprov",O_CREAT|O_RDWR|O_APPEND,0777);
      close(fu);
      close(fp);
      i=fork();
      switch(i)
      {
            case -1:
                        printf("Error al hacer el fork\n");
                        sleep(2);
                        break;
            case 0:
                        funcionsocketP();//conexion del socket proveedor
                        break;
            default:
                        if(i!=0)
                        {
                              j=fork();
                              switch(j)
                              {
                                    case -1:
                                                printf("Error");
                                                break;
                                    case 0:
                                                funcionsocketU();//socket
                                                comprobacion de usuarios
                                                break;
                                    default:
                                                if(j!=0)
```

```
{
	k=fork();
	switch(k)
	{
		case -1:
				printf("Error");
				break;
		case 0:
				funcionsocketLibrosP();
					//socket comprobacion de usuarios
				break;
		default:
				if(k!=0)
				{
					l=fork();
					switch(l)
					{
						case -1:
								printf("Error");
								break;
								case 0:
								funcionsocketLibrosU();
								break;
						default:
								menu();//menu principal
                                                                }
                                                        }

                                        }
                                }
                }
		}
	}
	return 0;
}
/* Parametros de entrada: void
 * Función: Recibe los mensajes del proveedor y guarda el libro en local
 * Parametro de salida: void
 * */
void funcionsocketLibrosP(void)
{
	COMANDOLIBROPROV clibro;
	struct sockaddr_in dir, dircli;
	int sd,ss;
	int lon;

	sd=socket(AF_INET,SOCK_STREAM,0);
	if(sd<0)
	{
		printf("Error en el socket\n");
		exit(1);
	}
	//inicializamos la estructura
	bzero(&dir,sizeof(dir));
	dir.sin_family=AF_INET;
	dir.sin_port=htons(1041);
	dir.sin_addr.s_addr=INADDR_ANY;
```

```
	if(bind(sd,(struct sockaddr *)&dir,sizeof(dir))<0)
	{
		printf("Error en el bin socketLibrosP\n");
		exit(1);
	}

	listen(sd,3);
	while(1)
	{
		lon=sizeof(dircli);
		ss=accept(sd,(struct sockaddr *)&dircli,(socklen_t *)&lon);
		if(ss==-1)
		{
			printf("Error en la conexion\n");
		}
		else
		{

	recv(ss,&clibro,sizeof(COMANDOLIBROPROV),MSG_WAITALL);//recibimos
                                                              un mensaje
			//printf("Libro recibido\n");
			guardaLibro(clibro);
			close(ss);
		}
	}
}
/* Parametros de entrada: void
 * Función: Recibe los datos del usuario, comprueba la accion que
 * quiere realizar:LOGIN,ALTA,BAJA.
 * Parametro de salida: 0 si el resultado de la comprobacion es correcto,
 * 1 si es erroneo.
 * */
void funcionsocketU(void)
{
	COMANDOUSU cusu;
	USUARIO usu;
	RESPUESTA v;
	struct sockaddr_in dir, dircli;
	int sd,ss,d;
	int lon,res;

	sd=socket(AF_INET,SOCK_STREAM,0);
	if(sd<0)
	{
		printf("Error en el socket\n");
		exit(1);
	}
	//inicializamos la estructura
	bzero(&dir,sizeof(dir));
	dir.sin_family=AF_INET;
	dir.sin_port=htons(1050);
	dir.sin_addr.s_addr=INADDR_ANY;

	if(bind(sd,(struct sockaddr *)&dir,sizeof(dir))<0)
	{
		printf("Error en el bin socketU\n");
```

```
	exit(1);
}

listen(sd,3);
while(1)
{
	lon=sizeof(dircli);
	ss=accept(sd,(struct sockaddr *)&dircli,(socklen_t *)&lon);
	if(ss==-1)
	{
		printf("Error en la conexion\n");
	}
	else
	{
		recv(ss,&cusu,sizeof(COMANDOUSU),MSG_WAITALL);
		if (strcmp(cusu.comando,"LOGIN")==0)
		{
			res=compruebausu(cusu.nombre,cusu.contrau);
			if(res==0)
			{
				v.valor=0;
				send(ss,&v,sizeof(RESPUESTA),0);
			}
			else
			{
				v.valor=1;
				send(ss,&v,sizeof(RESPUESTA),0);
			}
			close(ss);
		}
		else if (strcmp(cusu.comando,"ALTA")==0)
		{
			res=compruebausu(cusu.nombre,cusu.contrau);
			//printf("res=%d\n",res);
			if(res==0)
			{
				v.valor=1;
				send(ss,&v,sizeof(RESPUESTA),0);
			}
			else
			{
				strcpy(usu.nombre,cusu.nombre);
				strcpy(usu.contra,cusu.contrau);

		d=open("./Servidor/registrousu",O_WRONLY|O_APPEND);
				write(d,&usu,sizeof(USUARIO));
				close(d);
				v.valor=0;
				send(ss,&v,sizeof(RESPUESTA),0);
			}
		}
		else if (strcmp(cusu.comando,"BAJA")==0)
		{
			res=compruebausu(cusu.nombre,cusu.contrau);
			if(res==0)
			{
				borrarusu(cusu.nombre,cusu.contrau);
```

```
                              v.valor=0;
                              send(ss,&v,sizeof(RESPUESTA),0);
                        }
                        else
                        {
                              v.valor=1;
                              send(ss,&v,sizeof(RESPUESTA),0);
                        }
                  }
                  else
                  {
                        v.valor=1;
                        send(ss,&v,sizeof(RESPUESTA),0);
                  }
            }
      }
}
/* Parametros de entrada: void
 * Función: Recibe los datos del proveedor, y del usuario del proveedor
 * comprueba la accion que quiere
                        realizar:LOGIN,ALTA,BAJA,LOGINU,ALTAU,BAJAU.
 * Parametro de salida: 0 si el resultado de la comprobacion es correcto,
 * 1 si es erroneo.
 * */
void funcionsocketP(void)
{
      COMANDOPROV cprov;
      PROV prov;
      RESPUESTA v;
      struct sockaddr_in dir, dircli;
      int sd,ss,d;
      int lon,res;

      sd=socket(AF_INET,SOCK_STREAM,0);
      if(sd<0)
      {
            printf("Error en el socket\n");
            exit(1);
      }
      //inicializamos la estructura
      bzero(&dir,sizeof(dir));
      dir.sin_family=AF_INET;
      dir.sin_port=htons(1040);
      dir.sin_addr.s_addr=INADDR_ANY;

      if(bind(sd,(struct sockaddr *)&dir,sizeof(dir))<0)
      {
            printf("Error en el bind socketP\n");
            exit(1);
      }

      listen(sd,3);
      while(1)
      {
            lon=sizeof(dircli);
            ss=accept(sd,(struct sockaddr *)&dircli,(socklen_t *)&lon);
            if(ss==-1)
```

```
        {
              printf("Error en la conexion\n");
        }
        else
        {
              recv(ss,&cprov,sizeof(COMANDOPROV),MSG_WAITALL);

              if ((strcmp(cprov.comando,"LOGINP"))==0)
              {
                    //printf("COMANDO: LOGINP \n");
                    res=compruebaprov(cprov.proveedor,cprov.contra);
                    if(res==0)
                    {
                          v.valor=0;
                          send(ss,&v,sizeof(RESPUESTA),0);
                    }
                    else
                    {
                          v.valor=1;
                          send(ss,&v,sizeof(RESPUESTA),0);
                    }
                    close(ss);
              }
              else if ((strcmp(cprov.comando,"LOGINU"))==0)
              {
                    //printf("COMANDO: LOGINU \n");

res=compruebauprov(cprov.proveedor,cprov.contra,cprov.nombre,cprov.
                                                      contrau);

                    if(res==0)
                    {
                          v.valor=0;
                          send(ss,&v,sizeof(RESPUESTA),0);
                    }
                    else
                    {
                          v.valor=1;
                          send(ss,&v,sizeof(RESPUESTA),0);
                    }
                    close(ss);
              }
              else if ((strcmp(cprov.comando,"ALTAP"))==0)
              {
                    //printf("COMANDO: ALTAP \n");
                    res=compruebaprov(cprov.proveedor,cprov.contra);
                    //printf("valor de res en Altap %d\n",res);
                    if(res==0)
                    {
                          v.valor=1;
                          send(ss,&v,sizeof(RESPUESTA),0);
                    }
                    else
                    {
                          strcpy(prov.proveedor,cprov.proveedor);
                          strcpy(prov.contra,cprov.contra);
```

```
d=open("./Servidor/registroprov",O_WRONLY|O_APPEND);
                        write(d,&prov,sizeof(PROV));
                        close(d);
                        v.valor=0;
                        send(ss,&v,sizeof(RESPUESTA),0);
                  }
            }
            else if ((strcmp(cprov.comando,"BAJAP"))==0)
            {
                  //printf("COMANDO: BAJAP \n");
                  res=compruebaprov(cprov.proveedor,cprov.contra);
                  if(res==0)
                  {
                        borrarprov(cprov.proveedor,cprov.contra);
                        v.valor=0;
                        send(ss,&v,sizeof(RESPUESTA),0);
                  }
                  else
                  {
                        v.valor=1;
                        send(ss,&v,sizeof(RESPUESTA),0);
                  }
            }
            else if ((strcmp(cprov.comando,"ALTAUP"))==0)
            {
                  //printf("COMANDO: ALTAUP \n");

res=compruebauprov(cprov.proveedor,cprov.contra,cprov.nombre,cprov.
                                                      contrau);
                  //printf("valor de res en Altaup %d\n",res);
                  if(res==0)
                  {
                        v.valor=1;
                        send(ss,&v,sizeof(RESPUESTA),0);
                  }
                  else
                  {
                        strcpy(prov.proveedor,cprov.proveedor);
                        strcpy(prov.contra,cprov.contra);
                        strcpy(prov.nombre,cprov.nombre);
                        strcpy(prov.contrau,cprov.contrau);

d=open("./Servidor/registroprov",O_WRONLY|O_APPEND);
                        write(d,&prov,sizeof(PROV));
                        close(d);
                        v.valor=0;
                        send(ss,&v,sizeof(RESPUESTA),0);
                  }
            }
            else if ((strcmp(cprov.comando,"BAJAUP"))==0)
            {
                  //printf("COMANDO: BAJAUP \n");

res=compruebauprov(cprov.proveedor,cprov.contra,cprov.nombre,cprov.
                                                      contrau);
                  if(res==0)
```

```
                        {

      borraruprov(cprov.proveedor,cprov.contra,cprov.nombre,cprov.contrau
                                                                        );
                              v.valor=0;
                              send(ss,&v,sizeof(RESPUESTA),0);
                        }
                        else
                        {
                              v.valor=1;
                              send(ss,&v,sizeof(RESPUESTA),0);
                        }
                  }
                  else
                  {
                        v.valor=1;
                        send(ss,&v,sizeof(RESPUESTA),0);
                  }
            }
      }
}
/* Parametros de entrada: usuario,contra.
 * Para pasarle el nombre el usuario y contraseÃ±a del usuario.
 * Función: esta funciÃ³n comprueba si el usuario esta dado de alta.
 * Parametro de salida: 0 si no existe, 1 si existe.
 * */
int compruebausu(char usuario[20],char contra[10])
{
      USUARIO usu;
      int d;
      d=open("./Servidor/registrousu",O_RDONLY);
      while(read(d,&usu,sizeof(USUARIO))!=0)
      {
            if((strcmp(usu.nombre,usuario)==0) &&
(strcmp(usu.contra,contra)==0))
            {
                  printf("El usuario existe en el servidor.");
                  close(d);
                  return 0;
            }
      }
      printf("El usuario no existe en el servidor.");
      close(d);
      return 1;
}
/* Parametros de entrada: proveedor,contra,usuario,contrau.
 * Para pasarle el nombre del proveedor y contraseÃ±a, al que pertenece
 * y el usuario y contraseÃ±a del usuario.
 * Función: esta funciÃ³n comprueba si el usuario proveedor esta dado de
alta.
 * Parametro de salida: 0 si no existe, 1 si existe.
 * */
int compruebauprov(char proveedor[20],char contra[10],char
usuario[20],char contrau[10])
{
      PROV prov;
      int d;
```

```
	d=open("./Servidor/registroprov",O_RDONLY);
	while(read(d,&prov,sizeof(PROV))!=0)
	{
		if ((strcmp(prov.proveedor,proveedor)==0) &&
(strcmp(prov.contra,contra)==0) && (strcmp(prov.nombre,usuario)==0) &&
					(strcmp(prov.contrau,contrau)==0))
		{
			//printf("El usuario existe en el servidor para este
								proveedor.");
			close(d);
			return 0;
		}
	}
	//printf("El usuario no existe en el servidor para este
						proveedor.");
	close(d);
	return 1;
}
/* Parametros de entrada: proveedor,contra.
 * Para pasarle el nombre del proveedor y contraseÃ±a.
 * Función: esta funciÃ³n comprueba si el proveedor esta dado de alta.
 * Parametro de salida: 0 si no existe, 1 si existe.
 * */
int compruebaprov(char proveedor[20],char contra[10])
{
	PROV prov;
	int d;
	d=open("./Servidor/registroprov",O_RDONLY);
	while(read(d,&prov,sizeof(PROV))!=0)
	{
		if ((strcmp(prov.proveedor,proveedor)==0) &&
					(strcmp(prov.contra,contra)==0))
		{
			//printf("Existe el proveedor en el servidor.\n");
			close(d);
			return 0;
		}
	}
	//printf("No existe el proveedor en el servidor.\n");
	close(d);
	return 1;
}
/* Parametros de entrada: void
 * Función: esta funciÃ³n nos muestra el menu principal
 * Parametro de salida: void
 * */
void menu (void)
{
	char op;
	do
		{
			system("clear");
			printf("SERVIDOR\n");
			printf("==========\n");
			printf("A. Crear tema o subtema\n");
			printf("B. Borrar tema o subtema\n");
			printf("C. Borrar libro\n");
```

```
                printf("S. Cerrar servidor\n");
                printf("\n");
                printf("Introduce una opcion: ");
                scanf(" %c",&op);
                switch(op)
                {
                      case 'A':
                      case 'a':
                                  system("clear");
                                  crearToS();
                                  break;
                      case 'B':
                      case 'b':
                                  system("clear");
                                  borrarToS();
                                  break;
                      case 'C':
                      case 'c':
                                  system("clear");
                                  blibro();
                                  break;
                      case 'S':
                      case 's':
                                  system("clear");
                                  printf("Hasta la proxima...");
                                  sleep(2);
                                  break;
                      default:
                                  printf("La opcion escogida no es
                                                correcta...\n");
                }
          }
      while(op!='S' || op!='s');
}
/* Parametros de entrada: void
 * FunciÃ³n: esta funciÃ³n nos muestra el menu para elegir si queremos
 * crear un tema o subtema.
 * Parametro de salida: void
 * */
void crearToS (void)
{
      char ruta[256];
      int op;
      strcpy(ruta,"./Servidor/Temas/");
      printf("CREAR TEMA O SUBTEMA\n");
      printf("====================\n");
      printf("\n");
      printf("¿Que quieres crear?\n");
      printf("1. Tema\n");
      printf("2. Subtema\n");
      printf("\n");
      printf("Introduce una opcion: ");
      scanf(" %d",&op);

      switch(op)
      {
            case 1:
```

```
                        system("clear");
                        creartema(ruta);
                        break;
            case 2:
                        system("clear");
                        crearsubtema(ruta);
                        break;
            default:
                        printf("Opcion incorrecta;");

      }
}
/* Parametros de entrada: ruta. Para indicarle su situaciÃ³n.
 * Función: esta funciÃ³n nos crea un tema.
 * Parametro de salida: void
 * */
void creartema(char ruta[256])
{
      char tema[20];
      int l;
      printf("CREAR TEMA\n");
      printf("==========\n");
      printf("\n");
      printf("Los temas existentes son:\n");
      l=listar(ruta);
      if(l==1)printf("No se encontraron temas disponibles\n");
      printf("\n");
      printf("Introduce el nombre del tema: ");
      scanf("%s",tema);
      strcat(ruta,tema);
      crear(ruta);
}
/* Parametros de entrada: ruta. Para indicarle su situaciÃ³n.
 * Función: esta funciÃ³n nos crea un subtema.
 * Parametro de salida: void
 * */
void crearsubtema(char ruta[256])
{
      char tema[20],subtema[20];
      int op,l;
      printf("CREAR SUBTEMA\n");
      printf("=============\n");
      printf("\n");
      printf("Los temas existentes son:\n");
      l=listar(ruta);
      if(l==1)
      {
            printf("No se encontraron temas disponibles\n");
            printf("Introduce el nombre del subtema: ");
            scanf(" %s",tema);
            strcat(ruta,tema);
            crear(ruta);
      }
      else
      {
            printf("\n");
            printf("Introduce el nombre del tema principal: ");
```

```
            scanf(" %s",tema);
            system("clear");
            printf("CREAR SUBTEMA\n");
            printf("=============\n");
            printf("\n");
            printf("Los Subtemas existentes son:\n");
            strcat(ruta,tema);
            strcat(ruta,"/");
            l=listar(ruta);
            if(l==1)
            {
                  printf("No se encontraron subtemas disponibles\n");
                  printf("Introduce el nombre del subtema: ");
                  scanf("%s",tema);
                  strcat(ruta,tema);
                  crear(ruta);
            }
            else
            {
                  printf("¿Quieres crear un subtema '1' o un Sub-subtema
                                                            '2' ?: ");
                  scanf(" %d",&op);
                  if(op==2) crearsubtema(ruta);
                  else
                  {
                        printf("\n");
                        printf("Introduce el nombre del subtema: ");
                        scanf(" %s",subtema);
                        strcat(ruta,subtema);
                        strcat(ruta,"/");
                        crear(ruta);
                  }
            }
      }
}
/* Parametros de entrada: void
 * Función: esta funciÃ³n nos muestra el menu para elegir si queremos
 * borrar un tema o subtema.
 * Parametro de salida: void
 * */
void borrarToS(void)
{
      char ruta[256];
      int op;
      strcpy(ruta,"./Servidor/Temas/");
      printf("BORRAR TEMA O SUBTEMA\n");
      printf("=====================\n");
      printf("\n");
      printf("¿Que quieres borrar?\n");
      printf("1. Tema\n");
      printf("2. Subtema\n");
      printf("\n");
      printf("Introduce una opcion: ");
      scanf(" %d",&op);
      switch(op)
      {
            case 1:
```

```
                        borrartema(ruta);
                        break;
            case 2:
                        borrarsubtema(ruta);
                        break;
            default:
                        printf("Opcion incorrecta;");
      }
}
/* Parametros de entrada: ruta. Para indicarle su situaciÃ³n.
 * Función: esta funciÃ³n nos borra un tema.
 * Parametro de salida: void
 * */
void borrartema(char ruta[256])
{
      int l;
      char tema[20];
      printf("BORRAR TEMA\n");
      printf("===========\n");
      printf("\n");
      printf("Los temas existentes son:\n");
      l=listar(ruta);
      if(l==1)printf("No hay temas que borrar\n");
      else
      {
            printf("Introduce el nombre del tema a eliminar: ");
            scanf(" %s",tema);
            strcat(ruta,tema);
            strcat(ruta,"/");
            borrar(ruta);
      }
}
/* Parametros de entrada: ruta. Para indicarle su situaciÃ³n.
 * Función: esta funciÃ³n nos borra un subtema.
 * Parametro de salida: void
 * */
void borrarsubtema(char ruta[256])
{
      int l,op;
      char tema[20],subtema[20];
      printf("BORRAR SUBTEMA\n");
      printf("==============\n");
      printf("\n");
      printf("Los temas existentes son:\n");
      l=listar(ruta);
      if(l==1)
      {
            printf("No se encontraron temas disponibles\n");
      }
      else
      {
            printf("\n");
            printf("Introduce el nombre del tema principal: ");
            scanf(" %s",tema);
            system("clear");
            printf("BORRAR SUBTEMA\n");
            printf("==============\n");
```

```
            printf("\n");
            printf("Los Subtemas existentes son:\n");
            strcat(ruta,tema);
            strcat(ruta,"/");
            l=listar(ruta);
            if(l==1)
            {
                  printf("No se encontraron subtemas disponibles\n");
            }
            else
            {
                  printf("¿Quieres borrar un subtema '1' o un Sub-
                                                subtema '2' ?: ");
                  scanf(" %d",&op);
                  if(op==2) borrarsubtema(ruta);
                  else
                  {
                        printf("\n");
                        printf("Introduce el nombre del subtema: ");
                        scanf(" %s",subtema);
                        strcat(ruta,subtema);
                        strcat(ruta,"/");
                        borrar(ruta);
                  }
            }
      }
}
/* Parametros de entrada: ruta. Para indicarle su situaciÃ³n.
 * FunciÃ³n: esta funciÃ³n nos mostrarÃ¡ un listado de los temas que
 * tenemos disponibles en el servidor.
 * Parametro de salida: 0 si ha encontrado directorios, 1 si no.
 * */
int listar (char ruta[256])
{
      int cont=1;
      DIR *d;
      struct dirent *entrada;
      d=opendir(ruta);
      while((entrada=readdir(d))!=NULL)
            {
                  if(entrada->d_type==DT_DIR)
                        {
                              if(entrada->d_name[0]!='.')
                              {
                                    printf("%d. %s\n",cont,entrada-
                                                      >d_name);
                                    cont++;
                              }
                        }
            }
      closedir(d);
      if(cont==1)return 1;
      else return 0;
}
/* Parametros de entrada: void.
 * Función: esta funciÃ³n nos muestra los libros existentes en el
 * servidor nos pide el nombre del que queremos eliminar y lo elimina.
```

```
 * Parametro de salida: void.
 * */
void blibro(void)
{
	char tema[20],ruta[255];

	printf("BORRAR LIBRO\n");
	printf("============\n");
	printf("Los libros existentes en el servidor son:\n");
	printf("\n");
	strcpy(ruta,"./Servidor/Temas/");
	listaservidor(ruta);
	printf("\n");
	printf("Introduce el nombre del libro: ");
	scanf(" %s",tema);
	strcpy(ruta,"./Servidor/Temas/");
	buscaYborra(tema,ruta);//busqueda del titulo
}
/* Parametros de entrada: tema,ruta. Para indicarle nombre del libro y
 * su situación.
 * Función: esta funciÃ³n busca el libro y si lo encuentra lo borra.
 * Parametro de salida: void.
 * */
void buscaYborra(char tema[20],char ruta[255])
{
	char tmp[255],libro[20],des[20],op;
	strcpy(tmp,ruta);
	strcpy(libro,tema);
	strcat(libro,".txt");
	strcpy(des,tema);
	strcat(des,".des");
	DIR *d;
	struct dirent *entrada;
	//printf("Entra en busqueda con parametro=%s y
                                        ruta=%s\n",tema,ruta);
	d=opendir(ruta);
	while((entrada=readdir(d)))
		{
			if((entrada->d_type)==DT_REG)
				{
					if((strcmp(entrada->d_name,libro))==0)
					{
						printf("¿Desea borrar %s ? S o N:
			",substr(entrada->d_name,0,strlen(entrada->d_name)-4));
						scanf(" %c",&op);
						if(op=='S' || op=='s')
						{
							strcpy(tmp,ruta);
							strcat(tmp,libro);
							remove(tmp);
							printf("Libro.txt borrado
                                                %s\n",tmp);
							strcpy(tmp,ruta);
							strcat(tmp,des);
							remove(tmp);
							printf("Descriptor.des borrado
                                                %s\n",tmp);
```

```
                                        sleep(2);
                                  }
                            }
                      }
                      else if((entrada->d_type)==DT_DIR)
                      {
                            if((strcmp(entrada-
               >d_name,".")!=0)&&(strcmp(entrada->d_name,"..")!=0))
                            {
                                  strcpy(tmp,ruta);
                                  strcat(tmp,entrada->d_name);
                                  strcat(tmp,"/");
                                  //printf("Entra en nuevo directorio:
                                                        %s\n",tmp);
                                  buscaYborra(tema,tmp);
                            }
                            else
                            {
                                  strcpy(tmp,ruta);
                            }
                      }
                }
      closedir(d);
}
/* Parametros de entrada: ruta. Para indicarle su situaciÃ³n.
 * Función: esta funciÃ³n nos crea un directorio.
 * Parametro de salida: void.
 * */
void crear (char ruta[256])
{
      if(mkdir(ruta,0700)!=0)
       {
            printf("\nEl tema ya estaba creado\n");
            sleep(3);
       }
       else
       {
            printf("\nEl tema se ha creado con exito\n");
            sleep(3);
       }
}
/* Parametros de entrada: ruta. Para indicarle su situaciÃ³n.
 * Función: esta funciÃ³n nos borra un directorio.
 * Parametro de salida: void.
 * */
void borrar(char directorio[256])
{
      struct dirent *entrada;
      DIR *dir;
      char tmp[200];
      dir=opendir(directorio);
      if (dir==NULL)
          printf("Error en directorio %s\n",directorio);
      else
      while((entrada=readdir(dir)))
      {
            if((entrada->d_type)!=DT_DIR)
```

```
            {
                  strcpy(tmp,directorio);
                  strcat(tmp,entrada->d_name);
                  printf("Borrando fichero %s\n",tmp);
                  remove(tmp);
            }
            else
            {
                  if((strcmp(entrada->d_name,".")!=0)&&(strcmp(entrada-
                                                      >d_name,"..")!=0))
                  {
                        strcpy(tmp,directorio);

                        strcat(tmp,entrada->d_name);
                        strcat(tmp,"/");
                        printf("Entrando en directorio %s\n",tmp);
                        borrar(tmp);
                        printf("Borrando directorio %s\n",tmp);
                        rmdir(tmp);
                  }
            }
      }
      closedir(dir);
      rmdir(directorio);
}
/* Parametros de entrada: usuario,contra. Para indicarle el usuario y
 * contraseña.
 * Función: esta funciÃ³n nos elimina un usuario del registro del
servidor.
 * Parametro de salida: void.
 * */
void borrarusu(char usuario[20],char contra[10])
{
      USUARIO usu;
      int d,d2;
      d=open("./Servidor/registrousu",O_RDONLY);
      d2=open("./Servidor/registrousu2",O_CREAT|O_RDWR|O_APPEND,0777);
      while((read(d,&usu,sizeof(USUARIO))!=0) &&
                                    (strcmp(usu.nombre,usuario))!=0)
      {
            write(d2,&usu,sizeof(USUARIO));
      }
      close(d2);
      close(d);
      remove("./Servidor/registrousu");
      rename("./Servidor/registrousu2","./Servidor/registrousu");
}
/* Parametros de entrada: proveedor,contra,usuario,contrau. Para
 * indicarle el proveedor y contraseÃ±a, usuario y contraseÃ±a.
 * Función: esta funciÃ³n nos elimina un proveedor del registro del
servidor.
 * Parametro de salida: void.
 * */
void borraruprov(char proveedor[20],char contra[10],char usuario[20],char
contrau[10])
{
      PROV prov;
```

```
	int d,d2;
	d=open("./Servidor/registroprov",O_RDONLY);
	d2=open("./Servidor/registroprov2",O_CREAT|O_RDWR|O_APPEND,0777);
	while((read(d,&prov,sizeof(PROV))!=0) &&
(strcmp(prov.proveedor,proveedor)) && (strcmp(prov.nombre,usuario))!=0)
	{
		write(d2,&prov,sizeof(PROV));
	}
	close(d2);
	close(d);
	remove("./Servidor/registroprov");
	rename("./Servidor/registroprov2","./Servidor/registroprov");
}
/* Parametros de entrada: proveedor,contra. Para indicarle el proveedor y
 * contraseña.
 * Función: esta funciÃ³n nos elimina un proveedor del registro del
servidor.
 * Parametro de salida: void.
 * */
void borrarprov(char proveedor[20],char contra[10])
{
	PROV prov;
	int d,d2;
	d=open("./Servidor/registroprov",O_RDONLY);
	d2=open("./Servidor/registroprov2",O_CREAT|O_RDWR|O_APPEND,0777);
	while((read(d,&prov,sizeof(PROV))!=0) &&
				(strcmp(prov.proveedor,proveedor))!=0)
	{
		write(d2,&prov,sizeof(PROV));
	}
	close(d2);
	close(d);
	remove("./Servidor/registroprov");
	rename("./Servidor/registroprov2","./Servidor/registroprov");
}
/* Parametros de entrada: clibro-Estructura COMANDOLIBROPROV.
 * Contiene la informacion del libro.
 * Función: esta función nos guarda el libro recibido del proveedor,
 * en el servidor.
 * Parametro de salida: void.
 * */
void guardaLibro(COMANDOLIBROPROV clibro)
{
	DESC des;
	LIBRO libro;
	char ruta[255],ruta2[255],tema[255],subtema[255];
	int ft,fd,creado=1;
	strcpy(ruta,"./Servidor/Temas/");
	strcpy(subtema,clibro.tema);
	strcat(subtema,"/");
	while(creado!=0)
	{
		partir(subtema,tema,subtema);
		printf("valor de tema: %s\n",tema);
		printf("valor de subtema: %s\n",subtema);
		strcat(ruta,tema);
		strcat(ruta,"/");
```

```
            mkdir(ruta,0766);
            strcpy(tema,"");
            if(strcmp(subtema,"\0")==0)
            {
                  creado=0;
            }
      }

      strcpy(des.titulo,clibro.titulo);//copiamos datos a la estructura
      strcpy(des.autor,clibro.autor);
      strcpy(des.adq,clibro.adq);
      strcpy(des.prov,clibro.prov);
      strcpy(des.tema,clibro.tema);
      des.pag=clibro.pag;
      des.isbn=clibro.isbn;
      des.desc=clibro.desc;

      strcat(ruta,clibro.titulo);
      strcat(ruta,".des");

      printf("Antes de crear libro ruta= %s\n",ruta);

      fd=open(ruta,O_CREAT|O_RDWR,0777);//creamos descripcion
      write(fd,&des,sizeof(DESC));
      //printf("Descriptor de libro creado\n");
      close(fd);

      strcpy(libro.texto,clibro.texto);//copiamos datos a la estructura
      printf("El valor de clibro.tema %s\n",clibro.tema);
      strcpy(ruta2,"./Servidor/Temas/");
      strcat(ruta2,clibro.tema);
      strcat(ruta2,"/");
      strcat(ruta2,clibro.titulo);
      strcat(ruta2,".txt");
      printf("Antes de crear libro ruta= %s\n",ruta2);
      ft=open(ruta2,O_CREAT|O_RDWR,0777);//creamos libro
      write(ft,&libro,sizeof(LIBRO));
      //printf("Libro creado\n");
      close(ft);
}
/* Parametros de entrada: void
 * Función: Recibe los mensajes del clientem diferenciando entre:
 * LISTA y DESCARGA, para enviar un mensaje con una lista de libros
 * disponibles o los datos del libro para su descarga.
 * Parametro de salida: void
 * */
void funcionsocketLibrosU(void)
{
      COMANDOLISTALIBROUSU clista;
      COMANDOLIBROUSU clibro;
      struct sockaddr_in dir, dircli;
      int sd,ss;
      int lon,res;
      char ruta[255];
      strcpy(ruta,"./Servidor/Temas/");
      sd=socket(AF_INET,SOCK_STREAM,0);
      if(sd<0)
```

```
{
      printf("Error en el socket\n");
      exit(1);
}
//inicializamos la estructura
bzero(&dir,sizeof(dir));
dir.sin_family=AF_INET;
dir.sin_port=htons(1051);
dir.sin_addr.s_addr=INADDR_ANY;

if(bind(sd,(struct sockaddr *)&dir,sizeof(dir))<0)
{
      printf("Error en el bin socketLibrosU\n");
      exit(1);
}

listen(sd,3);
while(1)
{
      lon=sizeof(dircli);
      ss=accept(sd,(struct sockaddr *)&dircli,(socklen_t *)&lon);
      if(ss==-1)
      {
            printf("Error en la conexion\n");
      }
      else
      {

recv(ss,&clista,sizeof(COMANDOLISTALIBROUSU),MSG_WAITALL);
            //printf("Peticion recibida.
                              comando=%s\n",clista.comando);
            if (strcmp(clista.comando,"LISTA")==0)
            {
                  strcpy(clista.lista,"");
                  res=crearListalibros(clista.lista,ruta);
                  if(res==0)
                  {
                        //printf("Respuesta de lista enviada\n");

send(ss,&clista,sizeof(COMANDOLISTALIBROUSU),0);
                  }
                  else
                  {
                        //printf("Respuesta de lista enviada con
                                          lista vacia\n");
                        strcpy(clista.lista,"");

send(ss,&clista,sizeof(COMANDOLISTALIBROUSU),0);
                  }
                  close(ss);
            }
            else if (strcmp(clista.comando,"DESCARGA")==0)
            {
                  strcpy(clibro.titulo,clista.lista);
                  clibro=buscaLibro(clibro.titulo,ruta);
                  //printf("Enviado libro en formato
                        COMANDOLIBROUSU %s\n",clibro.titulo);
```

```
                        send(ss,&clibro,sizeof(COMANDOLIBROUSU),0);

                }
                else
                {
                        //printf("Error: Comando desconocido.");
                        strcpy(clista.lista,"");
                        send(ss,&clista,sizeof(COMANDOLISTALIBROUSU),0);
                }
            }
      }
}
/* Parametros de entrada: original,parte1,parte2. Le pasamos la ruta
original,
 * y dos variables mas donde vamos a guardar las particiones.
 * Función: Esta funcion va a coger el contenido de la variable tema,
 * del libro, y va a dividir esta variable para poder crear directorio a
directorio.
 * Parametro de salida: void
 * */
void partir(char *original, char *parte1, char *parte2)
{
  int i=0;
  int j=0;
  int noencontrado=0;

  while (original[i]!='\0')
  {
     if (noencontrado==0)
     {
        if (original[i]=='/')
        {
           noencontrado=1;
           parte1[i]='\0';
        }
        else
        {
           parte1[i]=original[i];
        }
     }
     else
     {
        parte2[j++]=original[i];
        //noencontrado=0;
     }
     i++;
  }
  parte2[j++]='\0';
}
/* Parametros de entrada: lista,ruta. Para devolver a la funcion que
 * le llama su contenido e indicarle su localizacion.
 * Función: Esta función crea una lista de libros existentes en el
 * servidor, para enviarle un mensaje al cliente con el resultado.
 * Parametro de salida: 0 si hay algun resultado, 1 en caso contrario.
 * */
int crearListalibros (char lista[200],char ruta[255])
{
```

```
	int longitud,res;
	char tmp[255],*extension,*libro;
	DIR *d;
	struct dirent *entrada;
	//printf("\nEntra en %s\n",ruta);
	d=opendir(ruta);

	while((entrada=readdir(d)))
		{
			if((entrada->d_type)==DT_REG)
				{
					strcpy(tmp,ruta);
					strcat(tmp,"/");
					strcat(tmp,entrada->d_name);
					//printf("No es directorio: %s\n",tmp);
					longitud=strlen(entrada->d_name);
					extension=substr(entrada->d_name,longitud-
                                                        4,4);
					//printf("extension=%s\n",extension);
					if(strcmp(extension,".txt")==0)
					{
						libro=substr(entrada-
                                                >d_name,0,longitud-4);
						strcat(lista,libro);
						strcat(lista,"/");
						//printf("lista actual =
                                                        %s\n",lista);
					}
				}
				else
				{
					if((strcmp(entrada-
			>d_name,".")!=0)&&(strcmp(entrada->d_name,"..")!=0))
					{

						strcpy(tmp,ruta);
						strcat(tmp,entrada->d_name);
						strcat(tmp,"/");
						//printf("Entra en nuevo directorio:
                                                        %s\n",tmp);
						res=crearListalibros(lista,tmp);
						if(res!=0) return 1;
					}
					else
					{
						strcpy(tmp,ruta);
					}
				}
		}
		return 0;
}
/* Parametros de entrada: ruta. Para indicar su situacion.
 * Función: Esta funcion nos hace un listado de los libros que hay en el
servidor.
 * Parametro de salida: void
 * */
void listaservidor (char ruta[255])
```

```
{
	int longitud;
	char tmp[255],*extension,*libro;
	DIR *d;
	struct dirent *entrada;

	d=opendir(ruta);
	while((entrada=readdir(d)))
		{
			if((entrada->d_type)!=DT_DIR)
				{
					strcpy(tmp,ruta);
					strcat(tmp,entrada->d_name);
					//printf("No es directorio: %s\n",tmp);
					longitud=strlen(entrada->d_name);
					extension=substr(entrada->d_name,longitud-
                                                                4,4);
					//printf("extension=%s\n",extension);
					if(strcmp(extension,".txt")==0)
					{
						libro=substr(entrada-
                                                >d_name,0,longitud-4);
						printf("%s\n",libro);
					}
				}
				else
				{
					if((strcmp(entrada-
			>d_name,".")!=0)&&(strcmp(entrada->d_name,"..")!=0))
					{
						strcpy(tmp,ruta);
						strcat(tmp,entrada->d_name);
						strcat(tmp,"/");
						//printf("Entra en nuevo directorio:
                                                                %s\n",tmp);
						listaservidor(tmp);
					}
				}
		}
}
/* Parametros de entrada: seleccion,ruta. Para indicarle el nombre del
 * libro y su situacion.
 * Función: ESta función busca un libro en concreto en el servidor.
 * Parametro de salida: COMANDOLIBROUSU
 * */
COMANDOLIBROUSU buscaLibro(char seleccion[20],char ruta[255])
{
	char tmp[255],libro[20],des[20];
	int fl,fd;
	COMANDOLIBROUSU clibro;
	LIBRO lib;
	DESC desc;
	strcpy(tmp,ruta);
	strcpy(libro,seleccion);
	strcat(libro,".txt");
	strcpy(des,seleccion);
	strcat(des,".des");
```

```
	DIR *d;
	struct dirent *entrada;
	//printf("Entra en busqueda con parametro=%s y
				ruta=%s\n",tema,ruta);
	d=opendir(ruta);
	while((entrada=readdir(d)))
		{
			if((entrada->d_type)==DT_REG)
				{
					if((strcmp(entrada->d_name,libro))==0)
					{
						strcat(tmp,entrada->d_name);
						fl=open(tmp,O_RDONLY);//leemos libro
						read(fl,&lib,sizeof(LIBRO));
						strcpy(clibro.texto,lib.texto);
						close(fl);
					}
					else if((strcmp(entrada->d_name,des))==0)
					{
						strcat(tmp,entrada->d_name);
						fd=open(tmp,O_RDONLY);//leemos libro
						read(fd,&desc,sizeof(DESC));
						close(fd);

						strcpy(clibro.titulo,desc.titulo);
						strcpy(clibro.autor,desc.autor);
						strcpy(clibro.adq,desc.adq);
						strcpy(clibro.prov,desc.prov);
						strcpy(clibro.tema,desc.tema);
						clibro.pag=desc.pag;
						clibro.isbn=desc.pag;
						clibro.desc=desc.desc;
					}
				}
				else if((entrada->d_type)==DT_DIR)
				{
					if((strcmp(entrada-
			>d_name,".")!=0)&&(strcmp(entrada->d_name,"..")!=0))
					{
						strcpy(tmp,ruta);
						strcat(tmp,entrada->d_name);
						strcat(tmp,"/");
						clibro=buscaLibro(seleccion,tmp);
					}
				}
		}
		return clibro;
}
/* Parametros de entrada: cadena,comienzo,longitud. Para indicarle la
* cadena que queremos dividir,el comienzo para dividir y el donde
			queremos cortar.
 * Función: esta función nos divide una cadena, para separar los nombres
 * de los archivos de sus extensiones.
 * Parametro de salida: nuevo
 * */
char* substr (char* cadena, int comienzo, int longitud)
{
```

```
	char* nuevo;
	if (longitud==0)
	{
		longitud=strlen(cadena)-comienzo-1;
	}

	nuevo= (char*)malloc(sizeof(char) * longitud);
	strncpy(nuevo,cadena+comienzo,longitud);
	return nuevo;
}
```

A.3.2. Código Cliente

```
#include "milibreria.h"
void menu (char servidor[20]);
void acceder(char servidor[20]);
void submenu(char servidor[20],char usuario[20],char contra[10]);
void disponibles(char servidor[20],char usuario[20]);
void copiar(char ruta[256],char tema[20]);
void leerlibro(char usuario[20]);
void eliminarlibro(char usuario[20]);
void alta(char servidor[20]);
void eliminar(char servidor[20]);
void borrar (char ruta[256]);
int comandoservidor(char servidor[20],char comando[20],char
usuario[20],char contra[10]);
int comandoservidorDescargar(char servidor[20],char comando[20],char
libro[50],char usuario[20]);
int comandoservidorLista(char servidor[20],char comando[20],char
lista[500]);
int comprobardir(char usuario[20]);
int listar (char ruta[256]);
void partir(char *original, char *parte1, char *parte2);
void guardaLibro(COMANDOLIBROUSU clibro,char usuario[20]);

/* Parametros de entrada: argc,argv para darle la ip del servidor
 * Función: principal
 * Parametro de salida: 0
 * */
int main(int argc, char *argv[])
{
	if(argc<2)printf("Error en parametros\n");
	else
	{
		mkdir("./Cliente",0700);
		menu(argv[1]);
	}
	return 0;
}
/* Parametros de entrada: servidor, para pasarle la ip
 * Función: esta función nos muestra el menu principal
 * Parametro de salida: void
 * */
void menu (char servidor[20])
```

```
{
	char op;
	do
		{
			system("clear");
			printf("CLIENTES\n");
			printf("=========\n");
			printf("A. Acceder\n");
			printf("B. Dar de alta\n");
			printf("C. Eliminar usuario\n");
			printf("S. Cerrar\n");
			printf("\n");
			printf("Introduce una opcion: ");
			scanf(" %c",&op);
			switch(op)
			{
				case 'A':
				case 'a':
						system("clear");
						acceder(servidor);

						break;
				case 'B':
				case 'b':
						system("clear");
						alta(servidor);
						break;
				case 'C':
				case 'c':
						system("clear");
						eliminar(servidor);
						break;
				case 'S':
				case 's':
						system("clear");
						printf("Hasta la proxima...");
						sleep(5);
						break;
				default:
						printf("La opcion escogida no es
							correcta...\n");
			}
		}
	while(op!='S' || op!='s');
}
/* Parametros de entrada: servidor, para pasarle la ip
 * Función: esta función accede al servidor con el nombre y contraseña de
un cliente
 * Parametro de salida: void
 * */
void acceder(char servidor[20])
{
	char usuario[20],contra[20],comando[20];
	int res;
	printf("ACCEDER\n");
	printf("=======\n");
	printf("\n");
```

```
	printf("Introduce el usuario: ");
	scanf("%s",usuario);
	printf("\nIntroduce la contraseÃ±a: ");
	scanf("%s",contra);
	strcpy(comando,"LOGIN");
	res=comandoservidor(servidor,comando,usuario,contra);
	if(res==0)
	{
		printf("\n");
		printf("Bienvenido al sistema usuario %s\n",usuario);
		sleep(2);
		submenu(servidor,usuario,contra);
	}
	else
	{
		printf("\n");
		printf("Lo sentimos, compruebe su usuario o contraseña, no
                                                son correctos\n");
		printf("Recuerde que si no esta dado de alta puede
                                                    hacerlo.\n");
		sleep(2);
	}
}
/* Parametros de entrada:
servidor,comando,proveedor,contra,usuario,contrau.
 * Para pasarle la ip del servidor, el comando a ejecutar, el nombre y
 * contraseÃ±a del proveedor al que pertenece y el usuario y contraseÃ±a
del usuario.
 * Función: esta función manda por socket un mensaje al servidor con los
datos.
 * parametro de salida: 0 si se ha ejecutado correctamente, 1 en caso
contrario.
 * */
int comandoservidor(char servidor[20],char comando[20],char
usuario[20],char contra[10])
{
	COMANDOUSU cusu;
	RESPUESTA v;
	struct hostent *h;
	struct sockaddr_in d;
	int s;

	strcpy(cusu.comando,comando);
	strcpy(cusu.nombre,usuario);
	strcpy(cusu.contrau,contra);

	h=gethostbyname(servidor);
	if(h)
	{
		bzero(&d,sizeof(d));
		d.sin_family=AF_INET;
		d.sin_port=htons(1050);
		bcopy(h->h_addr,&d.sin_addr,4);

		s=socket(AF_INET,SOCK_STREAM,0);
		if(s<0)
		{
```

```
                printf("Error en el socket\n");
                return 1;
            }
            if(connect(s,(struct sockaddr *)&d,sizeof(d)))
            {
                printf("Error al conectar\n");return 1;
            }
            else
            {
                send(s,&cusu,sizeof(COMANDOUSU),0); //enviamos mensaje
                recv(s,&v,sizeof(RESPUESTA),MSG_WAITALL);//recibimos
                                                          respuesta
                close(s);
                if(v.valor==0)return 0;
                else return 1;
            }

      }
      else
      {
            //printf("Error al solucionar localhost\n");
            return 1;
      }
}
/* Parametros de entrada: servidor,comando,libro,usuario.
 * Para pasarle la ip del servidor, el comando a ejecutar, el libro y
 * el usuario.
 * Función: esta función recibe por socket un mensaje al servidor con los
datos.
 * parametro de salida: 0 si se ha ejecutado correctamente, 1 en caso
contrario.
 * */
int comandoservidorDescargar(char servidor[20],char comando[20],char
libro[50],char usuario[20])
{
      COMANDOLIBROUSU clibro;
      COMANDOLISTALIBROUSU clista;
      struct hostent *h;
      struct sockaddr_in d;
      int s;

      strcpy(clista.comando,comando);
      strcpy(clista.lista,libro);

      h=gethostbyname(servidor);
      if(h)
      {
            bzero(&d,sizeof(d));
            d.sin_family=AF_INET;
            d.sin_port=htons(1051);
            bcopy(h->h_addr,&d.sin_addr,4);

            s=socket(AF_INET,SOCK_STREAM,0);
            if(s<0)
            {
                printf("Error en el socket\n");
                return 1;
```

```
            }
            if(connect(s,(struct sockaddr *)&d,sizeof(d)))
            {
                  printf("Error al conectar\n");return 1;
            }
            else
            {
                  send(s,&clista,sizeof(COMANDOLISTALIBROUSU),0);
                                                      //enviamos mensaje

      recv(s,&clibro,sizeof(COMANDOLIBROUSU),MSG_WAITALL);//recibimos
                                                      respuesta
                  guardaLibro(clibro,usuario);
                  close(s);
                  return 0;
            }

      }
      else return 1;
}
/* Parametros de entrada: servidor,comando,lista.
 * Para pasarle la ip del servidor, el comando a ejecutar, y la lista.
 * Función: esta función recibe por socket un mensaje al servidor con
 * los datos de los libros que hay disponibles en el servidor.
 * parametro de salida: 0 si hay resultado, 1 en caso contrario.
 * */
int comandoservidorLista(char servidor[20],char comando[20],char
lista[255])
{
      COMANDOLISTALIBROUSU clista;
      struct hostent *h;
      struct sockaddr_in d;
      int s;

      strcpy(clista.comando,comando);

      h=gethostbyname(servidor);
      if(h)
      {
            bzero(&d,sizeof(d));
            d.sin_family=AF_INET;
            d.sin_port=htons(1051);
            bcopy(h->h_addr,&d.sin_addr,4);

            s=socket(AF_INET,SOCK_STREAM,0);
            if(s<0)
            {
                  printf("Error en el socket\n");
                  return 1;
            }
            if(connect(s,(struct sockaddr *)&d,sizeof(d)))
            {
                  printf("Error al conectar\n");return 1;
            }
            else
            {
                  //printf("Peticion de lista enviada\n");
```

```
                send(s,&clista,sizeof(COMANDOLISTALIBROUSU),0);
                                                //enviamos mensaje

      recv(s,&clista,sizeof(COMANDOLISTALIBROUSU),MSG_WAITALL);//recibimo
                                                        s respuesta
                //printf("Respuesta de lista recibida\n");
                close(s);
                strcpy(lista,clista.lista);
                //printf("lista=%s\n",lista);
                if(strcmp(lista,"")!=0)return 0;
                else return 1;
            }

      }
      else return 1;
}
/* Parametros de entrada: servidor,usuario,contra.
 * Para pasarle la ip del servidor, el usuario, y la contraseña.
 * Función: esta función nos mostrará un submenu cuando el usuario
 * se haya logeado.
 * parametro de salida: void.
 * */
void submenu(char servidor[20],char usuario[20],char contra[10])
{
      char op;
      do
            {
                system("clear");
                printf("USUARIO\n");
                printf("=======\n");
                printf("A. Libros disponibles en el servidor\n");
                printf("B. Leer uno de nuestros libros\n");
                printf("C. Borrar libro\n");
                printf("S. Salir\n");
                printf("\n");
                printf("Introduce una opcion: ");
                scanf(" %c",&op);
                switch(op)
                {
                      case 'A':
                      case 'a':
                                        system("clear");
                                        disponibles(servidor,usuario);
                                        break;
                      case 'B':
                      case 'b':
                                        system("clear");
                                        leerlibro(usuario);
                                        break;
                      case 'C':
                      case 'c':
                                        system("clear");
                                        eliminarlibro(usuario);
                                        break;
                      default:
                                        printf("Opcion
incorrecta...\n");
```

```
                }
            }
      while(op!='S' || op!='s');
}
/* Parametros de entrada: servidor,usuario.
 * Para pasarle la ip del servidor, el usuario.
 * Función: esta función envia al servidor una petición del listado de
 * libros disponibles y los muestra.
 * parametro de salida: void.
 * */
void disponibles(char servidor[20],char usuario[20])
{
      char
lista[500],comando[20],libro1[50],restolista[500],seleccion[20];
      int res,finaldelista=0,indice=1;

      strcpy(comando,"LISTA");
      res=comandoservidorLista(servidor,comando,lista);
      if (res==0)
      {
            strcpy(restolista,lista);
            system("clear");
            printf("LISTADO DE LIBROS EN EL SERVIDOR\n");
            printf("================================\n");
            while(finaldelista==0)
            {
                  partir(restolista,libro1,restolista);
                  printf("%d. %s\n",indice,libro1);
                  if(strcmp(restolista,"\0")==0)
                  {
                        finaldelista=1;
                  }
            }
            printf("\n");
            printf("Introduce el nombre del libro que quieres descargar:
                                                                  ");
            scanf(" %s",seleccion);

            //llamar al servidor y pedir ese libro
            strcpy(comando,"DESCARGA");

      res=comandoservidorDescargar(servidor,comando,seleccion,usuario);
            if(res==0)
            {
                  printf("Libro recibido.\n");
            }
            else
            {
                  printf("Error al recibir el libro solicitado.\n");
            }
      }
      else
      {
            printf("Hubo un error al recuperar la lista de libros del
servidor.\n");
            sleep(2);
```

```
	}
	printf("Los libros existentes son:\n");
}
/* Parametros de entrada: usuario. Para indicarle el directorio del
 * usuario que queremos comprobar.
 * Función: esta función nos comprueba si tenemos menos de 5 libros.
 * parametro de salida: 0 si el resultado es positivo, 1 en caso
contrario.
 * */
int comprobardir(char usuario[20])
{
	struct dirent *entrada;
	DIR *dir;
	int cont=0;
	char ruta[256];

	strcpy(ruta,"./Cliente/");
	strcat(ruta,usuario);
	strcat(ruta,"/");
	dir=opendir(ruta);
	while((entrada=readdir(dir)))
	{
		if(entrada->d_type==DT_REG)
		{
			cont++;
		}
	}
	closedir(dir);
	if(cont==0 || cont<10) return 0;
	else return 1;
}
/* Parametros de entrada: usuario. Para indicarle el directorio del
 * usuario que queremos utilizar.
 * Función: esta función muestra los libros que tenemos descargados y
 * nos da la opcion de elegir uno para leerlo.
 * parametro de salida: void.
 * */
void leerlibro(char usuario[20])
{
	DESC des;
	LIBRO libro;
	char ruta[256],titulo[20],nombre[20],aux[256];
	int l,fl,fd;
	strcpy(ruta,"./Cliente/");
	strcat(ruta,usuario);
	strcat(ruta,"/");
	printf("Los libros existentes son:\n");
	l=listar(ruta);
	if(l==1)printf("No hay ningun libro por el momento\n");
	else
	{
		printf("Introduce el nombre del libro que desea leer: ");
		scanf(" %s",titulo);
		strcpy(nombre,titulo);
		strcat(nombre,".des");
		strcpy(aux,ruta);
		strcat(ruta,nombre);
```

```
            fd=open(ruta,O_RDONLY);
            read(fd,&des,sizeof(DESC));
            printf("titulo: %s\n",des.titulo);
            printf("autor: %s\n",des.autor);
            printf("fecha adquisicion: %s\n",des.adq);
            printf("proveedor: %s\n",des.prov);
            printf("paginas: %d\n",des.pag);
            printf("ISBN: %d\n",des.isbn);
            close(fd);
            strcpy(ruta,aux);
            strcat(ruta,".txt");
            fl=open(ruta,O_RDONLY);
            read(fl,&libro,sizeof(LIBRO));
            printf("Texto: %s\n",libro.texto);
            close(fl);
      }
}
/* Parametros de entrada: usuario. Para indicarle el directorio del
 * usuario que queremos utilizar.
 * Función: esta función nos elimina un libro de los que tenemos en
local.
 * parametro de salida: void.
 * */
void eliminarlibro(char usuario[20])
{
    int l;
    char ruta[256],tema[20];
    strcpy(ruta,"./Cliente/");
    strcat(ruta,usuario);
    strcat(ruta,"/");
    printf("Tus libros son: \n");
    l=listar(ruta);
    if(l==1)printf("No hay ningun libro por el momento\n");
    else
    {
        printf("Introduce el nombre del libro que desea borrar: ");
        scanf(" %s",tema);
        strcpy(ruta,"./Cliente/");
        strcat(ruta,usuario);
        strcat(ruta,"/");
        strcat(ruta,tema);
        strcat(ruta,".txt");
        remove(ruta);
        strcpy(ruta,"./Cliente/");
        strcat(ruta,usuario);
        strcat(ruta,"/");
        strcat(ruta,tema);
        strcat(ruta,".des");
        remove(ruta);
    }
}
/* Parametros de entrada: directorio. Para indicarle el directorio a
 * eliminar.
 * Función: esta función nos elimina un directorio recursivamente.
 * parametro de salida: void.
 * */
void borrar(char directorio[256])
```

```
{
      struct dirent *entrada;
      DIR *dir;
      char tmp[200];
      dir=opendir(directorio);
      if (dir==NULL)
          printf("Error en directorio %s\n",directorio);
      else
      while((entrada=readdir(dir)))
      {
            if((entrada->d_type)!=DT_DIR)
            {
                  strcpy(tmp,directorio);
                  strcat(tmp,entrada->d_name);
                  printf("Borrando fichero %s\n",tmp);
                  remove(tmp);
            }
            else
            {
                  if((strcmp(entrada->d_name,".")!=0)&&(strcmp(entrada-
                                          >d_name,"..")!=0))
                  {
                        strcpy(tmp,directorio);

                        strcat(tmp,entrada->d_name);
                        strcat(tmp,"/");
                        printf("Entrando en directorio %s\n",tmp);
                        borrar(tmp);
                        printf("Borrando directorio %s\n",tmp);
                        rmdir(tmp);
                  }
            }
      }
      closedir(dir);
      rmdir(directorio);
}
/* Parametros de entrada: ruta. Para indicarle el directorio del
 * usuario que queremos utilizar.
 * Función: esta función nos muestra un listado de los libros
                                          descargados.
 * parametro de salida: 0 en caso de obtener resultado, 1 en caso
                                          contrario.
 * */
int listar (char ruta[256])
{
      int cont=1;
      DIR *d;
      struct dirent *entrada;
      d=opendir(ruta);
      while((entrada=readdir(d))!=NULL)
            {
                  if((entrada->d_type)==DT_REG)
                        {
                              printf("%d. %s\n",cont,entrada-
                                          >d_name);
                              cont++;
                        }
```

```
            }
      if(cont==1)return 1;
      else return 0;
}
/* Parametros de entrada: servidor. Para indicarle la ip del servidor.
 * FunciÃ³n: esta funciÃ³n nos da de alta un usuario en el sistema.
 * parametro de salida: void.
 * */
void alta(char servidor[20])
{
      char ruta[255],usuario[20],contra[10],comando[20];
      int res;
      printf("DAR DE ALTA\n");
      printf("==========\n");
      printf("\n");
      printf("Introduce el usuario: ");
      scanf(" %s",usuario);
      printf("\nIntroduce la contraseÃ±a: ");
      scanf(" %s",contra);
      strcpy(comando,"ALTA");
      res=comandoservidor(servidor,comando,usuario,contra);
      if(res==0)
      {
            strcpy(ruta,"./Cliente/");
            strcat(ruta,usuario);
            strcat(ruta,"/");
            res=mkdir(ruta,0766);
            printf("¡¡Felicidades!!\n");
            printf("El usuario se ha creado con exito\n");
            sleep(2);
      }
      else
      {
            printf("El usuario no ha podido ser creado.\n");
            sleep(2);
      }
}
/* Parametros de entrada: servidor. Para indicarle la ip del servidor.
 * Función: esta función nos da de baja un usuario en el sistema.
 * parametro de salida: void.
 * */
void eliminar(char servidor[20])
{
      char usuario[20],contra[10],comando[20],ruta[256];
      int res;
      printf("BORRAR USUARIO\n");
      printf("==============\n");
      printf("\n");
      printf("Introduce el usuario: ");
      scanf("%s",usuario);
      printf("\nIntroduce la contraseÃ±a: ");
      scanf("%s",contra);
      strcpy(comando,"BAJA");
      res=comandoservidor(servidor,comando,usuario,contra);
      if(res==0)
      {
            strcpy(ruta,"./Cliente/");
```

```
            strcat(ruta,usuario);
            borrar(ruta);
            printf("El usuario ha sido dado de baja correctamente.");
            sleep(2);
      }
      else
      {
            printf("El usuario no ha podido ser eliminado.\n");
            sleep(2);
      }
}
/* Parametros de entrada: original,parte1,parte2. Le pasamos la ruta
original,
 * y dos variables mas donde vamos a guardar las particiones.
 * Función: Esta funcion va a coger el contenido de la variable tema,
 * del libro, y va a dividir esta variable para poder crear directorio a
directorio.
 * Parametro de salida: void
 * */
void partir(char *original, char * parte1, char * parte2)
{
  int i = 0;
  int j = 0;
  int noencontrado = 0;

  while (original[i] != '\0')
  {
     if (noencontrado == 0)
     {
        if (original[i] == '/')
        {
           noencontrado = 1;
           parte1[i] = '\0';
        }
        else
        {
           parte1[i] = original[i];
        }
     }
     else
     {
        parte2[j++] = original[i];
     }
     i++;
  }
  parte2[j++]='\0';
}
/* Parametros de entrada: clibro-Estructura COMANDOLIBROPROV,usuario.
 * Contiene la informacion del libro, y el nombre del usuario.
 * Función: esta función nos guarda el libro recibido del servidor.
 * Parametro de salida: void.
 * */
void guardaLibro(COMANDOLIBROUSU clibro,char usuario[20])//guardar libro
en cliente
{
      DESC des;
      LIBRO libro;
```

```
	char ruta[255];
	int ft,fd,res;
	strcpy(ruta,"./Cliente/");
	strcat(ruta,usuario);
	strcat(ruta,"/");
	res=comprobardir(usuario);
	if(res==1)
	{
		printf("Lo sentimos debe eliminar algun libro.\n");
	}
	else
	{

		//printf("ruta=%s\n",ruta);
		strcpy(des.titulo,clibro.titulo);//copiamos datos a la
                                                 estructura
		strcpy(des.autor,clibro.autor);
		strcpy(des.adq,clibro.adq);
		strcpy(des.prov,clibro.prov);
		strcpy(des.tema,clibro.tema);
		des.pag=clibro.pag;
		des.isbn=clibro.isbn;
		des.desc=clibro.desc;
		strcat(ruta,clibro.titulo);
		strcat(ruta,".des");
		//printf("ruta=%s\n",ruta);
		fd=open(ruta,O_CREAT|O_RDWR|O_APPEND,0766);//creamos libro
		write(fd,&des,sizeof(DESC));
		//printf("Descriptor de libro creado\n");
		close(fd);

		strcpy(libro.texto,clibro.texto);//copiamos datos a la
                                                        estructura

		strcpy(ruta,"./Cliente/");
		strcat(ruta,usuario);
		strcat(ruta,"/");
		strcat(ruta,clibro.titulo);
		strcat(ruta,".txt");
		printf("Antes de crear libro ruta=%s\n",ruta);
		ft=open(ruta,O_CREAT|O_RDWR|O_APPEND,0766);//creamos
                                                        descripcion
		write(ft,&libro,sizeof(LIBRO));
		printf("Libro creado\n");
		close(ft);
	}
}
```

A.3.3. Código Proveedor

```
#include "milibreria.h"

void menu (char servidor[20]); //menu principal
void accederp(char servidor[20]); //acceder como proveedor
```

```
void accederuprov(char servidor[20],char proveedor[20],char
contra[10]);//acceder como usuario de proveedor
int comandoservidorp(char servidor[20],char comando[20],char
proveedor[20],char contra[10],char usuario[20],char contrau[10]);
void submenu(char servidor[20],char prov[20],char contra[10]);//menu de
provee dor
void altaprov(char servidor[20]);//alta de proveedor
void altauprov(char servidor[20],char prov[20], char contra[10]);//alta
de usuario
void eliminarprov(char servidor[20]);//eliminar proveedor
void eliminaruprov(char servidor[20],char proveedor[20],char
contra[10]);//eliminar usuario
void borrarDir(char ruta[256]);//borrar cualquier directorio
void enviar(void);//enviar libro al servidor
void intro(char prov[20],char usuario[20]);//introducir libro
MEMO *pn;//Memoria compartida

/* Parametros de entrada: argc,argv para darle la ip del servidor
 * Función: principal
 * Parametro de salida: 0
 * */
int main(int argc, char *argv[])
{
      int shmid,msgid;
      msgid=msgget(KEYC,IPC_CREAT|0666);//creamos una COLA
      shmid=shmget(KEYM,sizeof(MEMO),IPC_CREAT | 0666); //creamos MEMORIA
                                                          COMPARTIDA
      pn=(MEMO *)shmat(shmid,0,0);
      pn->enviados=0;
      mkdir("./Proveedores",0700);
      if(argc<2)printf("Error en parametros\n");
      else
      {
            menu(argv[1]);
      }
      close(msgid);//cerramos cola
      close(shmid);//cerramos memoria
      return 0;
}
/* Parametros de entrada: servidor, para pasarle la ip
 * Función: esta función nos muestra el menu principal
 * Parametro de salida: void
 * */
void menu (char servidor[20]) //menu principal
{
      char op;
      do
            {
                  system("clear");
                  printf("PROVEEDORES\n");
                  printf("===========\n");
                  printf("A. Acceder\n");
                  printf("B. Dar de alta un proveedor\n");
                  printf("C. Eliminar proveedor\n");
                  printf("S. Cerrar\n");
                  printf("\n");
                  printf("Introduce una opcion: ");
```

```
                  scanf(" %c",&op);

                  switch(op)
                  {
                        case 'A':
                        case 'a':
                                    system("clear");
                                    accederp(servidor);//acceder
                                                proveedor

                                    break;
                        case 'B':
                        case 'b':
                                    system("clear");
                                    altaprov(servidor); //alta proveedor

                                    break;
                        case 'C':
                        case 'c':
                                    system("clear");
                                    eliminarprov(servidor); //eliminar
                                                      proveedor
                                    break;
                        case 'S':
                        case 's':
                                    system("clear");
                                    printf("Hasta la proxima...");
                                    sleep(2);
                                    break;
                        default:
                                    printf("La opcion escogida no es
                                                correcta...\n");
                  }
            }
      while(op!='S' || op!='s');
}
/* Parametros de entrada: servidor, para pasarle la ip
 * Función: esta función accede al servidor con el nombre y contraseña de
un proveedor
 * Parametro de salida: void
 * */
void accederp(char servidor[20]) //acceder proveedor
{
      COMANDOPROV cprov;
      int res;
      printf("ACCEDER\n");
      printf("=======\n");
      printf("\n");
      printf("Introduce el nombre de proveedor: ");
      scanf("%s",cprov.proveedor);
      printf("\nIntroduce la contraseÃ±a de proveedor: ");
      scanf("%s",cprov.contra);
      strcpy(cprov.comando,"LOGINP");
      res=comandoservidorp(servidor,cprov.comando,cprov.proveedor,cprov.c
                                                      ontra,"","");
      if(res==0)
      {
```

```
            printf("\n");
            printf("Bienvenido al sistema proveedor:
                                                %s\n",cprov.proveedor);
            sleep(2);
            submenu(servidor,cprov.proveedor,cprov.contra); //menu para
                                          el usuario del proveedor
      }
      else
      {
            printf("\n");
            printf("Lo sentimos, compruebe su usuario o contraseña, no
                                                      son correctos\n");
            printf("Recuerde que si no esta dado de alta puede
                                                      hacerlo.\n");
            sleep(2);
      }
}
/* Parametros de entrada: servidor,proveedor,contra. Para pasarle la ip
 * del servidor, y el nombre y contraseÃ±a del proveedor al que pertenece
 * FunciÃ³n: esta funciÃ³n nos muestra el menu principal
 * Parametro de salida: void
 * */
void accederuprov(char servidor[20],char proveedor[20],char contra[10])
{
      COMANDOPROV cprov;
      int res;
      printf("ACCEDER\n");
      printf("=======\n");
      printf("\n");
      printf("Introduce el nombre de usuario: ");
      scanf("%s",cprov.nombre);
      printf("\nIntroduce la contraseÃ±a de usuario: ");
      scanf("%s",cprov.contrau);
      strcpy(cprov.comando,"LOGINU");
      res=comandoservidorp(servidor,cprov.comando,proveedor,contra,cprov.
                                                nombre,cprov.contrau);
      if(res==0)
      {
            printf("\n");
            printf("Bienvenido al sistema usuario: %s\n",cprov.nombre);
            intro(proveedor,cprov.nombre); //menu para el usuario del
                                                            proveedor
      }
      else
      {
            printf("\n");
            printf("Lo sentimos, compruebe su usuario o contraseña, no
                                                      son correctos\n");
            printf("Recuerde que si no esta dado de alta puede
                                                      hacerlo.\n");
            sleep(2);
      }
}
/* Parametros de entrada:
servidor,comando,proveedor,contra,usuario,contrau.
 * Para pasarle la ip del servidor, el comando a ejecutar, el nombre y
```

```
 * contraseña del proveedor al que pertenece y el usuario y contraseña
del usuario.
 * Función: esta función manda por socket un mensaje al servidor con los
datos.
 * parametro de salida: 0 si no existe, 1 si existe.
 * */
int comandoservidorp(char servidor[20],char comando[20],char
proveedor[20],char contra[10],char usuario[20],char contrau[10])
{
	COMANDOPROV cprov;
	RESPUESTA v;
	struct hostent *h;
	struct sockaddr_in d;
	int s;

	strcpy(cprov.comando,comando);
	strcpy(cprov.proveedor,proveedor);
	strcpy(cprov.contra,contra);
	strcpy(cprov.nombre,usuario);
	strcpy(cprov.contrau,contrau);

	h=gethostbyname(servidor);
	if(h)
	{
		bzero(&d,sizeof(d));
		d.sin_family=AF_INET;
		d.sin_port=htons(1040);
		bcopy(h->h_addr,&d.sin_addr,4);

		s=socket(AF_INET,SOCK_STREAM,0);
		if(s<0)
		{
			printf("Error en el socket\n");
			return 1;
		}
		if(connect(s,(struct sockaddr *)&d,sizeof(d)))
		{
			printf("Error al conectar\n");
			return 1;
		}
		else
		{
			send(s,&cprov,sizeof(COMANDOPROV),0); //enviamos
                                                              mensaje
			recv(s,&v,sizeof(RESPUESTA),MSG_WAITALL);//recibimos
                                                              respuesta
			//printf("Mensaje de respuesta recibido.
                                                 v.valor=%d\n",v.valor);
			if(v.valor==0)
			{
				return 0;
			}
			else
			{
				return 1;
			}
			close(s);
```

```
            }

      }
      else return 1;
}
/* Parametros de entrada: proveedor,contra. Para pasarle el nombre y
 * contraseÃ±a del proveedor .
 * Función: esta función comprueba si el proveedor esta dado de alta.
 * Parametro de salida: 0 si no existe, 1 si existe.
 * */
int comprueba(char usuario[20],char contra[20]) //comprueba si el usuario
existe
{
      PROV usu;
      int d;
      d=open("./Servidor/registroprov",O_RDONLY);
      while(read(d,&usu,sizeof(USUARIO))!=EOF || usu.nombre==usuario)
      {
            if(usu.nombre==usuario && usu.contra==contra)
            {
                  return 0;
            }
            else return 1;
      }
      close(d);
      if(d==0)return 0;
      else return 1;
}
/* Parametros de entrada: servidor,proveedor,contra.
 * Para pasarle la ip del servidor, y el nombre y contraseÃ±a del
proveedor.
 * FunciÃó: esta función que nos muestra un menu del proveedor.
 * Parametro de salida: void.
 * */
void submenu(char servidor[20],char prov[20],char contra[10])
{
      char op;
      do
            {
                  system("clear");
                  printf("PROVEEDOR: %s\n",prov);
                  printf("==========================\n");
                  printf("A. Acceder\n");
                  printf("B. Crear usuario\n");
                  printf("C. Borrar usuario\n");
                  printf("S. Salir\n");
                  printf("\n");
                  printf("Introduce una opcion: ");
                  scanf(" %c",&op);
                  switch(op)
                  {
                        case 'A':
                        case 'a':
                                          system("clear");
                                    accederuprov(servidor,prov,contra);
                                          //alta de usuario
                                          break;
```

```
                        case 'B':
                        case 'b':
                                    system("clear");
                              altauprov(servidor,prov,contra);
                                    //alta de usuario
                                    break;
                        case 'C':
                        case 'c':
                                    system("clear");
                              eliminaruprov(servidor,prov,contra);
                                    //eliminar usuario
                                    break;
                        case 'S':
                        case 's':
                                    system("clear");
                                    printf("Hasta la proxima...");
                                    sleep(2);
                                    break;
                        default:
                                    printf("La opcion escogida no
                                          es correcta...\n");
                  }
            }
      while(op!='S' || op!='s');
}
/* Parametros de entrada: prov,usuario,contra.
 * Para pasarle el nombre del proveedor al que pertenece y el usuario
 * y contraseÃ±a del usuario.
 * Función: esta función comprueba si el usuario esta dado de alta.
 * Parametro de salida: 0 si no existe, 1 si existe.
 * */
int compruebausu(char prov[20],char usuario[20],char contra[20])
{
      PROV usu;
      int d;
      d=open("./Servidor/registroprov",O_RDONLY);
      while(read(d,&usu,sizeof(PROV))!=EOF || usu.nombre==usuario)
      {
            if(usu.proveedor==prov && usu.nombre==usuario &&
                                    usu.contrau==contra)
            {
                  return 0;
            }
            else return 1;
      }
      close(d);
      if(d==0)return 0;
      else return 1;
}
/* Parametros de entrada: servidor. Para pasarle la ip del servidor.
 * Función: esta función recoge los datos del proveedor que queremos
 * dar de alta.
 * Parametro de salida: void.
 * */
void altaprov(char servidor[20])
{
      char ruta[256],proveedor[20],contra[10],comando[20];
```

```
	int res;
	printf("DAR DE ALTA\n");
	printf("===========\n");
	printf("\n");
	printf("Introduce el proveedor: ");
	scanf(" %s",proveedor);
	printf("\nIntroduce la contraseÃ±a del proveedor: ");
	scanf(" %s",contra);
	strcpy(comando,"ALTAP");
	printf("\nLlama a comandoservidorp\n");
	res=comandoservidorp(servidor,comando,proveedor,contra,"","");//com
                                                        prueba el proveedor
	printf("valor de res en Altaup %d\n",res);
	if(res==0)
	{
		strcpy(ruta,"./Proveedores/");
		strcat(ruta,proveedor);
		mkdir(ruta,0777);
		printf("¡¡Felicidades!!\n");
		printf("El proveedor se ha creado con exito.\n");
		sleep(2);
	}
	else
	{
		printf("Entra en respuesta 1\n");
		printf("El proveedor no ha podido ser creado.\n");
		sleep(2);
	}
}
/* Parametros de entrada: servidor,proveedor,contra. Para recoger los
 * datos del usuario que queremos dar de alta.
 * Función: esta función que recoge los datos del usuario que queremos
 * dar de alta.
 * Parametro de salida: void.
 * */
void altauprov(char servidor[20],char proveedor[20],char contra[20])
{
	char ruta[256],nombre[20],contrau[10],comando[20];
	int res;
	printf("DAR DE ALTA\n");
	printf("===========\n");
	printf("\n");
	printf("Introduce el usuario: ");
	scanf("%s",nombre);
	printf("\nIntroduce la contraseÃ±a: ");
	scanf("%s",contrau);
	strcpy(comando,"ALTAUP");
	res=comandoservidorp(servidor,comando,proveedor,contra,nombre,contr
                                        au);//comprueba el proveedor
	printf("valor de res en Altaup %d\n",res);
	if(res==0)
	{
		strcpy(ruta,"./Proveedores/");
		strcat(ruta,proveedor);
		strcat(ruta,"/");
		strcat(ruta,nombre);
		mkdir(ruta,0777);
```

```
            printf("¡¡Felicidades!!\n");
            printf("El usuario se ha creado con exito.\n");
            sleep(2);
      }
      else
      {
            printf("El proveedor no ha podido ser creado.\n");
            sleep(2);
      }
}
/* Parametros de entrada: servidor. Para pasarle la ip del servidor.
 * Función: esta función que recoge los datos del proveedor a eliminar
 * y los comprueba en el servidor.
 * parametro de salida: void.
 * */
void eliminarprov(char servidor[20])
{
      char ruta[256],comando[20];
      int res;
      PROV prov;
      printf("BORRAR PROVEEDOR\n");
      printf("================\n");
      printf("\n");
      printf("Introduce el proveedor a borrar: ");
      scanf(" %s",prov.proveedor);
      printf("\nIntroduce la contraseÃ±a del proveedor a borrar: ");
      scanf(" %s",prov.contra);
      strcpy(comando,"BAJAP");
      res =
comandoservidorp(servidor,comando,prov.proveedor,prov.contra,"","");//com
prueba usuario y contraseña
      if(res==0)
      {
            strcpy(ruta,"./Proveedores/");
            strcat(ruta,prov.proveedor);
            borrarDir(ruta);//borra sus carpetas
            printf("El proveedor ha sido eliminado\n");
            sleep(2);
      }
      else
      {
            printf("El proveedor no ha podido ser eliminado\n");
            sleep(2);
      }
}
/* Parametros de entrada: servidor,proveedor,contra. Para pasarle la ip
 * del servidor, y el nombre y contraseÃ±a del usuario.
 * Función: esta función recoge los datos del usuario que queremos
 * eliminar y lo comprueba en el servidor.
 * parametro de salida: void.
 * */
void eliminaruprov(char servidor[20],char proveedor[20], char contra[10])
{
      char ruta[256],comando[20];
      int res;
      PROV prov;
      strcpy(prov.proveedor,proveedor);
```

```
	strcpy(prov.contra,contra);
	printf("BORRAR USUARIO DE PROVEEDOR\n");
	printf("==========================\n");
	printf("\n");
	printf("Introduce el usuario a borrar: ");
	scanf(" %s",prov.nombre);
	printf("\nIntroduce la contraseÃ±a del usuario a borrar: ");
	scanf(" %s",prov.contrau);
	strcpy(comando,"BAJAP");
	res =
comandoservidorp(servidor,comando,prov.proveedor,prov.contra,prov.nombre,
prov.contrau);//comprueba usuario y contraseÃ±a
	if(res==0)
	{
		strcpy(ruta,"./Proveedores/");
		strcat(ruta,prov.proveedor);
		strcat(ruta,"/");
		strcat(ruta,prov.nombre);
		borrarDir(ruta);//borra sus carpetas
		printf("El usuario ha sido eliminado\n");
		sleep(2);
	}
	else
	{
		printf("El usuario no ha podido ser eliminado\n");
		sleep(2);
	}
}
/* Parametros de entrada: ruta.
 * Función: esta función elimina recursivamente cualquier directorio.
 * parametro de salida: void.
 * */
void borrarDir(char ruta[256])
{
	struct dirent *entrada;
	DIR *dir;
	char tmp[200];
	dir=opendir(ruta);
	if (dir==NULL)
	    printf("Error en directorio %s\n",ruta);
	else
	while((entrada=readdir(dir)))
	{
		if((entrada->d_type)!=DT_DIR)
		{
			strcpy(tmp,ruta);
			strcat(tmp,entrada->d_name);
			printf("Borrando fichero %s\n",tmp);
			remove(tmp);
		}
		else
		{
			if((strcmp(entrada->d_name,".")!=0)&&(strcmp(entrada-
>d_name,"..")!=0))
			{
				strcpy(tmp,ruta);
```

```
                        strcat(tmp,entrada->d_name);
                        strcat(tmp,"/");
                        printf("Entrando en directorio %s\n",tmp);
                        borrarDir(tmp);
                        printf("Borrando directorio %s\n",tmp);
                        rmdir(tmp);
                  }
            }
      }
      closedir(dir);
      rmdir(ruta);
}
/* Parametros de entrada: prov,usuario. Para pasarle el nombre del
 * proveedor y el del usuario.
 * Función: esta función recoge los datos del libro y los envia a una
 * cola de mensajes.
 * parametro de salida:void.
 * */
void intro(char prov[20],char usuario[20])
{
      int msgid;
      COMANDOLIBROPROV clibro;

      msgid=msgget(KEYC,0666);//nos enganchamos a la cola

      //introduccimos los datos del libro
      printf("INTRODUCIR LIBRO\n");
      printf("================\n");
      fflush(stdin);
      printf("\nIntroduce el titulo: ");
      scanf("%s",clibro.titulo);
      fflush(stdin);
      printf("\nIntroduce el autor: ");
      scanf("%s",clibro.autor);
      fflush(stdin);
      clibro.tipo=1;
      printf("\nIntroduce el numero de paginas: ");
      scanf(" %d",&clibro.pag);
      fflush(stdin);
      printf("\nIntroduce el ISBN: ");
      scanf(" %d",&clibro.isbn);
      fflush(stdin);
      printf("\nIntroduce el fecha de adquisicion: ");
      scanf("%s",clibro.adq);
      fflush(stdin);
      printf("\nIntroduce el proveedor: ");
      scanf("%s",clibro.prov);
      fflush(stdin);
      printf("\nIntroduce el tema: ");
      scanf("%s",clibro.tema);
      fflush(stdin);
      printf("\nIntroduce el texto del libro: \n");
      scanf("%s",clibro.texto);
      fflush(stdin);
      msgsnd(msgid,&clibro,sizeof(COMANDOLIBROPROV),0);
      pn->enviados++;
}
```

A.3.4. Código Cola

```
#include "milibreria.h"
void proceso2(char servidor[20]);
MEMO *pn;//Memoria compartida
int espera=1; //Espera de 10 minutos
void minutos(int s)
{
      espera=0;
}
/* Parametros de entrada: argc,argv para darle la ip del servidor
 * Función: principal
 * Parametro de salida: 0
 * */
int main(int argc, char *argv[])
{
      int shmid,msgid,proc2,bucle=0,vez=0;
      msgid=msgget(KEYC,IPC_CREAT|0666);//creamos una COLA
      shmid=shmget(KEYM,sizeof(MEMO),IPC_CREAT | 0666); //creamos MEMORIA
                                                              COMPARTIDA
      pn=(MEMO *)shmat(shmid,0,0);
      signal(SIGUSR1,minutos);
      printf("PUNTO 1\n");
      if(argc<2)printf("Error en parametros\n");
      else
      {
            proc2=fork();
            switch(proc2)
            {
                  case -1:
                              printf("Error en proceso lector\n");
                              break;
                  case 0:

                              while(bucle==0)
                              {
                                    if (espera==0)
                                    {
                                          vez++;
                                          printf("Entra proceso.
                                          Comprobación n°: %d\n",vez);
                                          printf("El total de libros
                        enviados al servidor es: %d\n",pn->enviados);
                                          sleep(2);
                                          proceso2(argv[1]);
                                          espera=1;
                                    }
                              }
                              break;
                  default:
                              while (bucle==0)
                              {
```

```
                                    kill(proc2,SIGUSR1);
                                    sleep(20);//Espera 20 segundos

                              }
            }
      }
      close(msgid);//cerramos cola
      close(shmid);//cerramos memoria
      return 0;
}
/* Parametros de entrada: servidor.Para darle la ip del servidor.
 * Función: Esta funcion nos recoge los mensajes de la cola y se los
 * manda al servidor mediante socket.
 * Parametro de salida: 0
 * */
void proceso2(char servidor[20]) //recoge de la cola
{
      int msgid;
      COMANDOLIBROPROV clibro;
      //variables socket
      struct hostent *h;
      struct sockaddr_in d;
      //struct in_addr dir;
      int n,s;
      msgid=msgget(KEYC,0666);//nos enganchamos a la cola
      //leemos de la cola
      if(-1==msgrcv(msgid,&clibro,sizeof(COMANDOLIBROPROV),1,IPC_NOWAIT))
      {
            printf("No hay ningun mensaje\n");
      }
      else
      {

            printf("Hay mensajes en la cola.\n");
            h=gethostbyname(servidor);
            if(h)
            {
                  bzero(&d,sizeof(d));
                  d.sin_family=AF_INET;
                  d.sin_port=htons(1041);
                  bcopy(h->h_addr,&d.sin_addr,4);
                  s=socket(AF_INET,SOCK_STREAM,0);
                  if(s<0)
                  {
                        printf("Error en el socket\n");
                        exit(1);
                  }
                  if(connect(s,(struct sockaddr *)&d,sizeof(d)))
                  {
                        printf("Error al conectar\n");
                  }
                  else
                  {

                        n=send(s,&clibro,sizeof(COMANDOLIBROPROV),0);
//enviamos mensaje
                  }
```

```
                    close(s);
                }
        }
}
```