Entrega No 3
Andres Mauricio Rondon - 2879687 (amrondonp)
Jaime Andres Vargas - 2879689 (jaavargasar)

Manual de uso:

1. Para compilar el programa, se ejecuta el siguiente comando en una terminal. Esta debe estar situada en el directorio donde se encuentran todos los archivos. (makefile , p3-dogClient.c, p3-dogServer.c).

   ```
   make
   ```

   Esto compilará los codigos fuentes del cliente y el servidor y generará los ejecutables.

2. Para ejecutar la aplicación, primero se debe ejecutar el servidor. El servidor es el único que posee acceso a los ficheros con la información.

   ```
   ./p3-dogServer
   ```

   El programa servidor no mostrará mensaje alguno. una vez hecho esto, ya podemos ejecutar el cliente

3. Para ejecutar el cliente, primero se saber la dirección ip del dispositivo en el cual se está ejecutando el servidor. Pero si en el mismo computador se están ejecutando los dos programas, esta direccion será 127.0.0.1. Se utiliza el siguiente comando:

   ```
   ./p3-dogClient [direccion ip]
   ```

   Si no se especifica direccion ip, es decir no se escribe nada después de el comando, este tomará por defecto la direccion local “ 127.0.0.1 “

   Nota: el puerto 6789 debe estar disponible en el servidor.

Cuando el cliente establece información con el servidor, aparecerá un menú con 5 diferentes opciones, las cuales permitirán al cliente ingresar un registro, consultar todos los registros,borrar un registro o buscar un registro dado el nombre.

Ingresar un registro: Para poder ingresar un nuevo registro, se debe oprimir la tecla 1. A continuación, se debe ingresar el nombre, luego la edad, la raza, la altura, el peso, luego se ingresa ‘M’ si es masculino o se ingresa ‘F’ si es femenino. Por último se debe confirmar si se desea ingresar el registro, si es así, se debe introducir la tecla ‘S’ para aceptar la confirmación o la tecla ‘N’ negar la confirmacion. Si se oprime la ‘S’ el sistema imprimirá un mensaje que diga

“cambios guardados” y se deberá ingresar cualquier tecla para imprimir de nuevo el menú, pero si se oprime la ‘N’ dira “cambios no guardados” y mostrará el menú por defecto.

Ver registro: Para poder ver todos los registros disponibles que están en el sistema se deberá oprimir la tecla 2. Este comando nos mostrará una tabla con dos columnas, en la izquierda imprimirá todos los los id de registros y en la derecha todos los nombres que se han ingresado previamente. Luego de esto se deberá ingresar el número de registro deseado que nos permite ver toda la información de un registro. Si se ingresa un registro válido imprimirá el nombre,la edad,la raza,la altura,el peso y el género y seguido a esto se deberá ingresar cualquier tecla para mostrar el menú. Pero si se ingresa un registro no válido imprimirá “No es un registro válido” y mostrará el menú por defecto.

Borrar registro: Para poder borrar un registro se debe oprimir la tecla 3. Este comando mostrará dos columnas , en la izquierda mostrará todos los registros y en la derecha todos los nombres ingresados previamente. Luego se deberá ingresar el número de registro a borrar, y seguido a esto aparecerá una confirmación de mensaje en la cual si se desea borrar el registro completo se debe oprimir la tecla ‘S’ , pero si no se quiere borrar se debe oprimir la tecla ‘N’. Si se oprime la ‘S’ saldrá un mensaje que diga “cambios guardados” y “registro borrado” luego aparecera el menu inicial. Pero si se oprime la ‘N’, mostrara un mensaje que diga “cambios no guardados” y se mostrará el menú por defecto.

Buscar registro: Para poder buscar un registro primero se debe oprimir la tecla 4,seguido a esto se debe ingresar el nombre de registro a buscar. Luego el sistema imprimirá dos columnas , en la izquierda todos los números de los registros encontrados y en la derecha todos los nombres que coinciden con el nombre ingresado previamente. Después se debe ingresar uno de estos registros y a continuación se imprimirá el nombre,la edad,la raza,la altura,el peso y el género, pero si se ingresa un nombre no válido , el sistema imprimirá “No se encontraron resultados”. Por último se deberá oprimir cualquier tecla para poder ver el menú por defecto.
Si se desea salir del sistema se deberá oprimir la tecla 5, seguido a esto cualquier tecla la cual hará que se termine de ejecutar el programa. El Servidor siempre está en ejecución esperando las solicitudes de los 32 posibles clientes y a su vez va respondiendo a estas solicitudes.

Funciones del cliente:

void create_registry();
Pregunta por la informacion del perro que se desea crear y envia el comando de menu al servidor.

int list_registry();
Le pregunta al servidor por todos los registros de perros que se encuentran en el sistema , escoge un registro y retorna este registro al servidor para poder ver su información detallada.

void search_registry(char * name);

Envia un nombre al servidor , para que este busque todos los registros que coincidan con el nombre, luego le pregunta al usuario cual de los registros encontrados quiere ver con detalle.
Parametros:

char * name; Apuntador al nombre que se desea buscar.

void erase_registry();
Lista todos los perros que se encuentran en el sistema y le pregunta al usuario que escoja uno de estos para enviarselo al servidor y que este pueda ser borrado.

void a_pause();
Pausa la ejecución del proceso.

int show_menu();
Muestra el menú principal y luego le pregunta al usuario por un comando a realizar. Retorna la opción del usuario si se encuentra entre las opciones o -1 si la opción no es válida.

char are_you_sure();
Pregunta al usuario si está seguro de que se realicen cambios en el sistema. Retorna una 'S' si el usuario está de acuerdo o una 'N' si no acepta la confirmacion.

void empty_buffer();
vacia el stdin buffer.

void send_to_server(void * buf, int bytes);
Envía datos a la ip del servidor.
Parametros:

void * buf; Apuntador a los datos que se desean enviar.

int bytes; Cuántos bytes se desean enviar.

void show_dog(struct Dog * dog);
Muestra en la consola, la información del perro.
Parametros:

Struct Dog * dog; El puntero del perro que se desea ver la información en detalle.

Funciones del servidor:

void create_registry(void * buf);
Agrega a la estructura de datos un nuevo perro, el cual ha sido ingresado previamente por el cliente.
Parametros:

void * buf; El puntero que contiene el tamaño y la información del perro.

void list_registry(int sock_cli);

Envía todos los registros de perros a el cliente, espera por la opción de registro que escoja el cliente y luego envía el dato en detalle del perro que el cliente desea consultar.

Parametros:

int sock_cli; Descriptor del socket del cliente.

int erase_registry(int sock_cli);

Envía todos los registros de los perros al cliente, espera por el id del perro que el cliente desea borrar y luego borra este registro. Retorna el número de registro borrado.

Parametros:

int sock_cli; el descriptor del socket del cliente.

char * search_registry(int sock_cli);

Recibe un string del registro que se desea buscar del cliente, envía todos los registros que coincidan con este string, luego espera por el registro que el cliente desea ver en detalle y finalmente envia la informacion del perro en detalle al cliente. Retorna el apuntador del nombre buscado.

Parametros:

int sock_cli; Descriptor del socket del cliente;

void charge_system();

Inicializa el sistema , busca si hay datos en el file “dataDogs.dat”. Si encuentra carga los datos a una estructura de perros, si no inicializa un sistema nuevo.

void save_system();

Guarda el estado actual del sistema en el file “dataDogs.dat”. Este sobrescribe la información que haya en el file.

void configurate_socket();

Crea el socket del servidor y envía el descriptor de este al cliente.

void *manage_client(void * args);

Una función que procesa hilos. Maneja la coneccion de un cliente, recibe y ejecuta los comandos que este procese, llama a las necesarias funciones y escribe un historial en el file “serverDogs.log”.

Parametros:

void * args; Puntero a un buffer que contiene el descriptor del socket client y el número del hilo que se está manejando.

void getDogResource();

Función que se sincroniza la entrada de los “lectores” al recurso, en este caso el archivo donde se guardan los datos (dataDog.dat). A la vez permite que varios lectores de forma simultánea, puedan adquirir el recurso y leer este al tiempo, sin importar el número de clientes. También

permite que si un cliente está leyendo, se bloquee la escritura, dado a que leer y escribir deben ser mutuamente excluyentes.

void freeDogResource();
Función que finaliza la sincronización entre lectores. Cuando el último lector a finalizado de utilizar el recurso esté lo libera, permitiendo que los clientes que quieran escribir en el archivo (dataDog.dat) puedan hacerlo sin ningun problema.

void sys_log(int client_index, int op, char * args);
Función que escribe datos a “serverDogs.log”
Parametros:

int client_index; El cliente que hace la solicitud (número del hilo que esta manejando)
int op; Comando que el cliente quiere ejecutar.
char * args; Información adicional para escribir en el file.

char * toString(int x);
Convierte un entero a un string. Retorna un puntero al string.
Parametros:

int x; El número a ser convertido a string;

Especificaciones funcionales:

1. Soporta y maneja la información correspondiente a perros tal como edad, nombre, raza, peso, género,altura.
2. Permite ingresar un nuevo registro a la base de datos
3. Permite consultar registros por nombre
4. Permite borrar registros
5. Permite consultar todos los registros y ver uno en particular

Especificaciones no funcionales:

6. Está diseñado bajo el modelo cliente servidor
7. Se permite un maximo de 32 clientes conectados en simultáneo con el servidor
8. Por defecto, el número máximo de perros es de 1’000.000 esto se puede modificando muy levemente el código fuente (una línea).

Diagramas de comunicación:

Diagrama de comunicación para la función ‘create_registry’

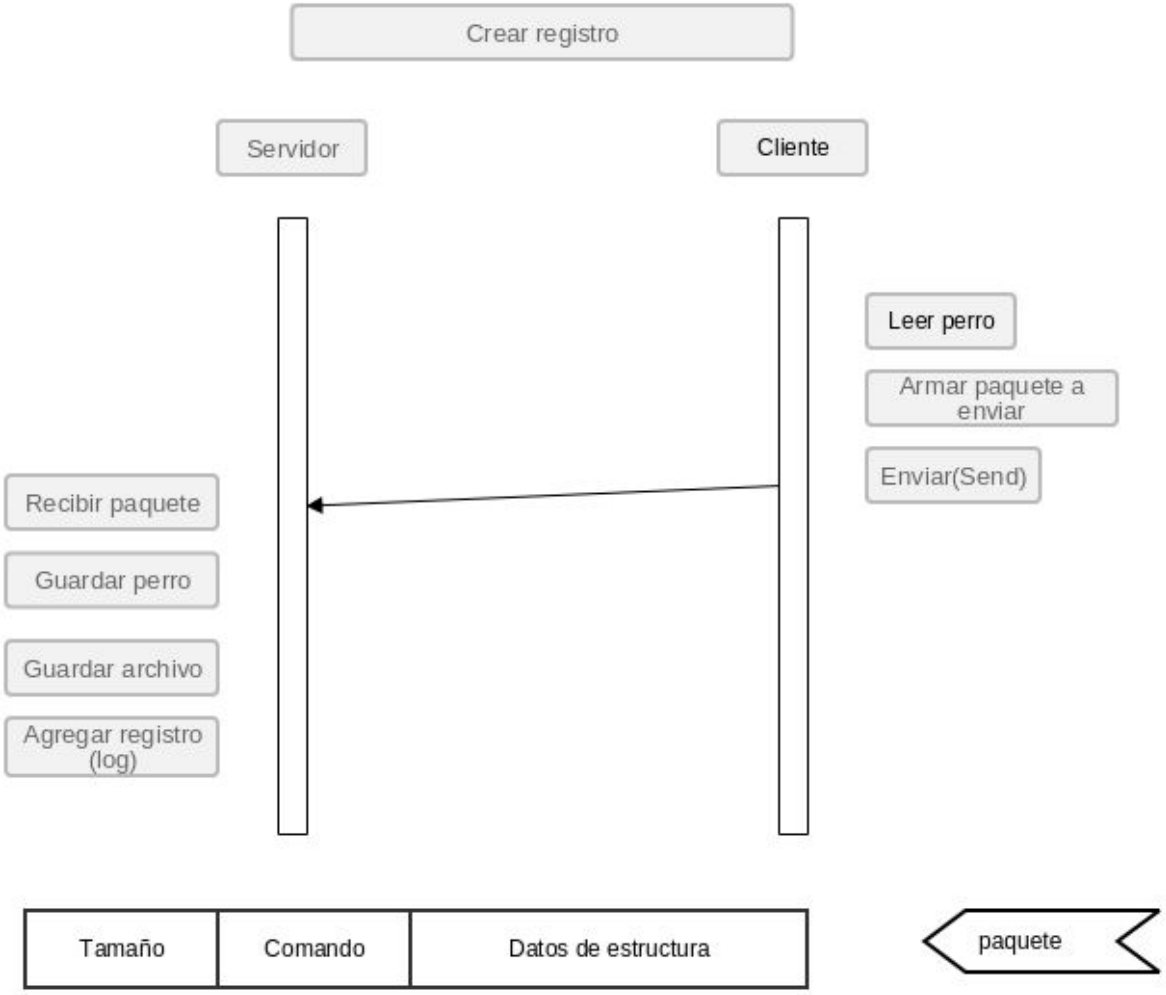


Diagrama de comunicación para la función list_registry

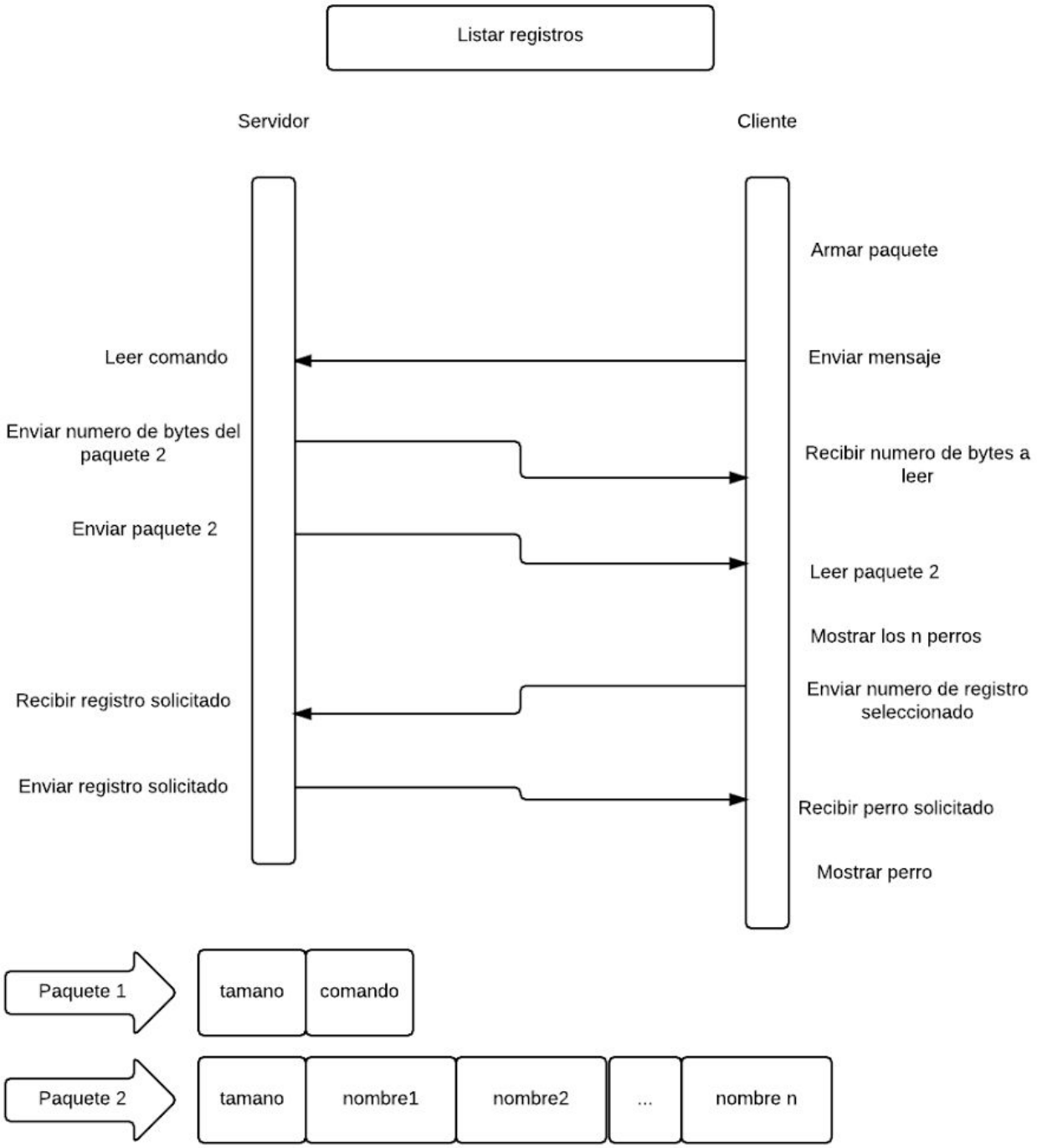


Diagrama de comunicación para erase_regestry

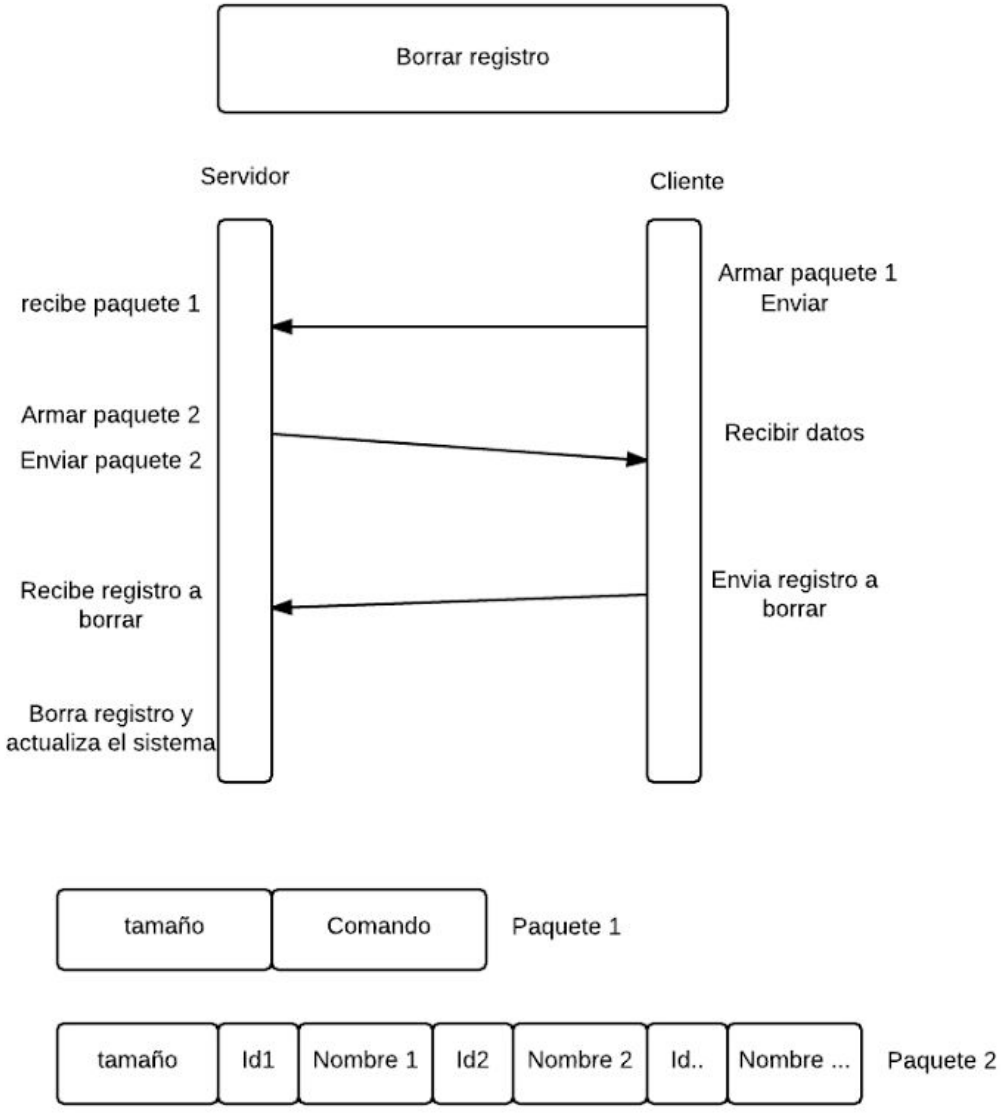
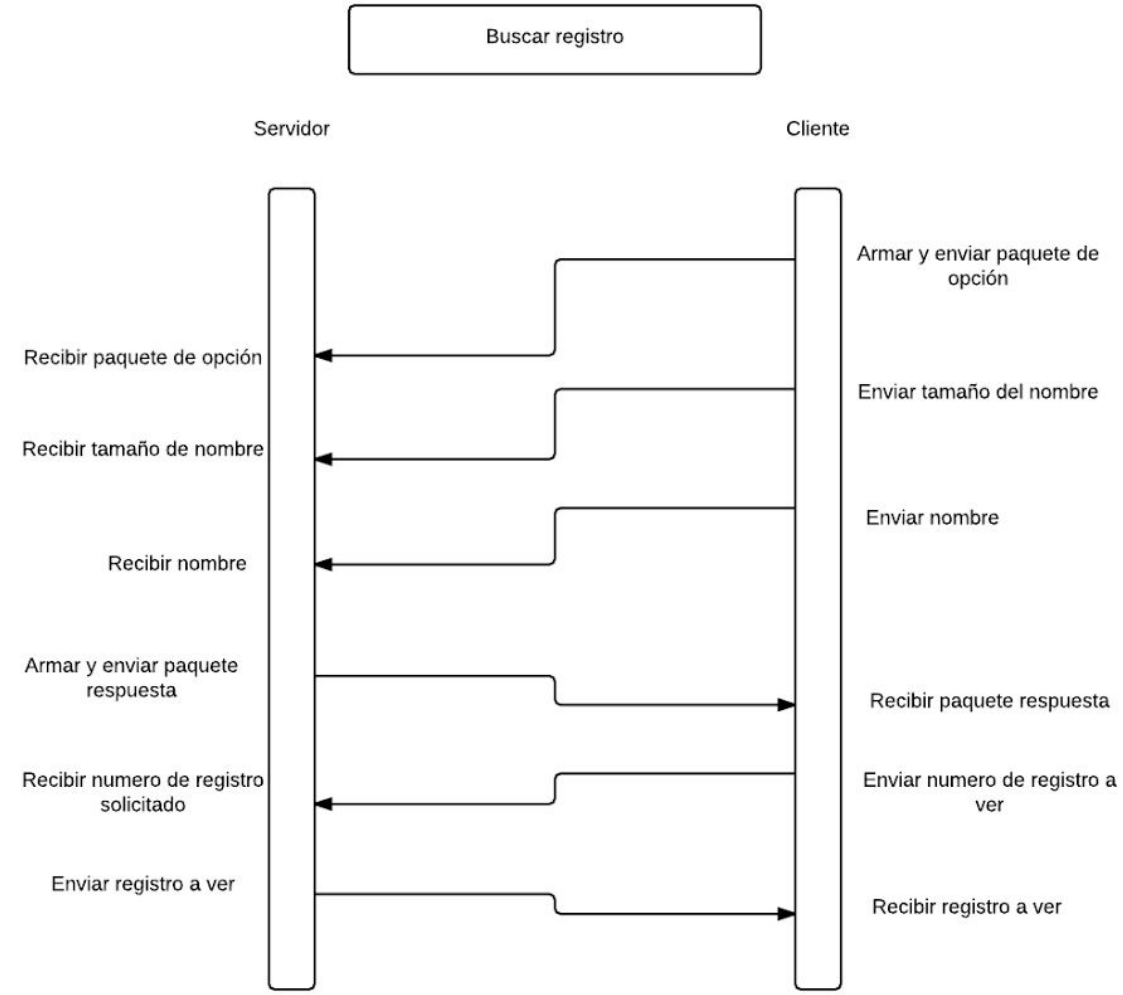
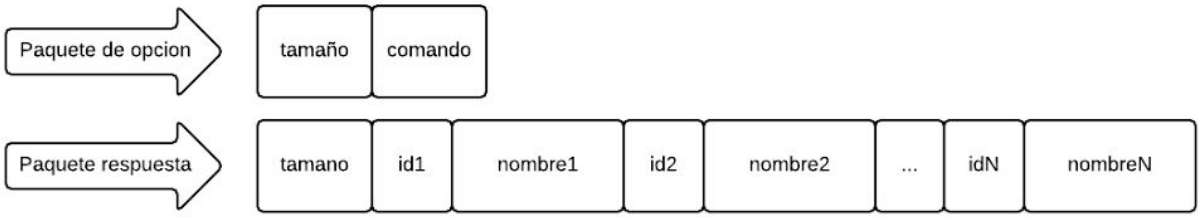


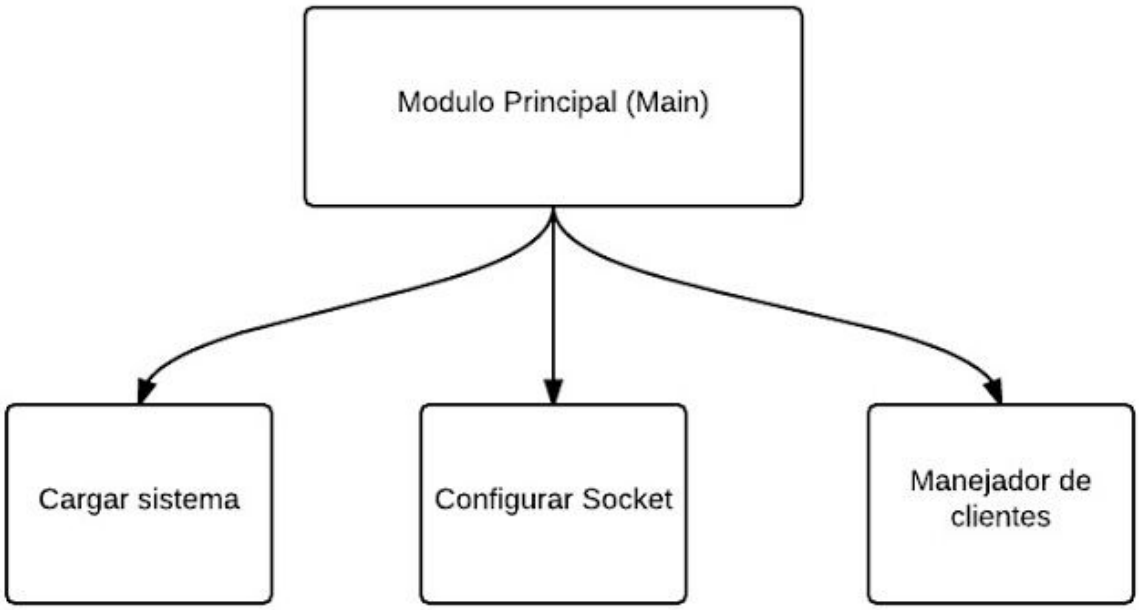
Diagrama de comunicación para buscar registro

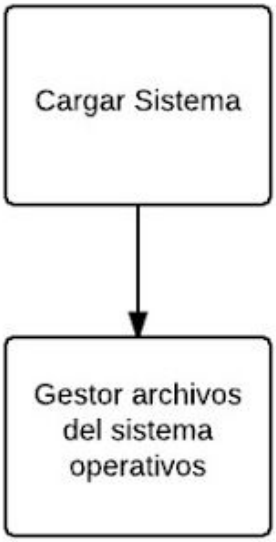


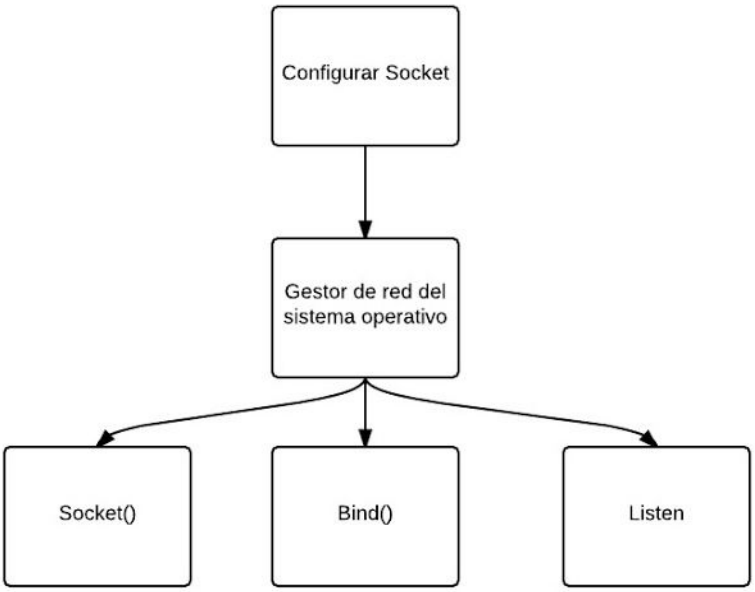


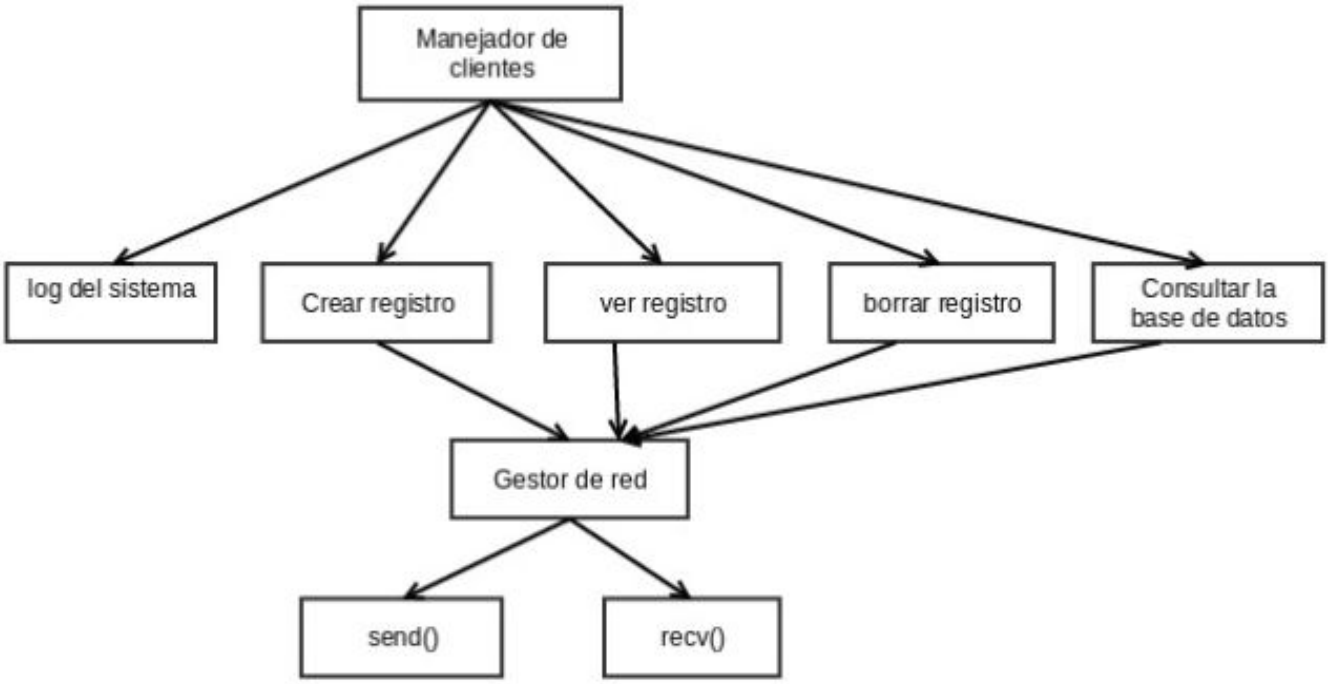
Diagramas de bloques

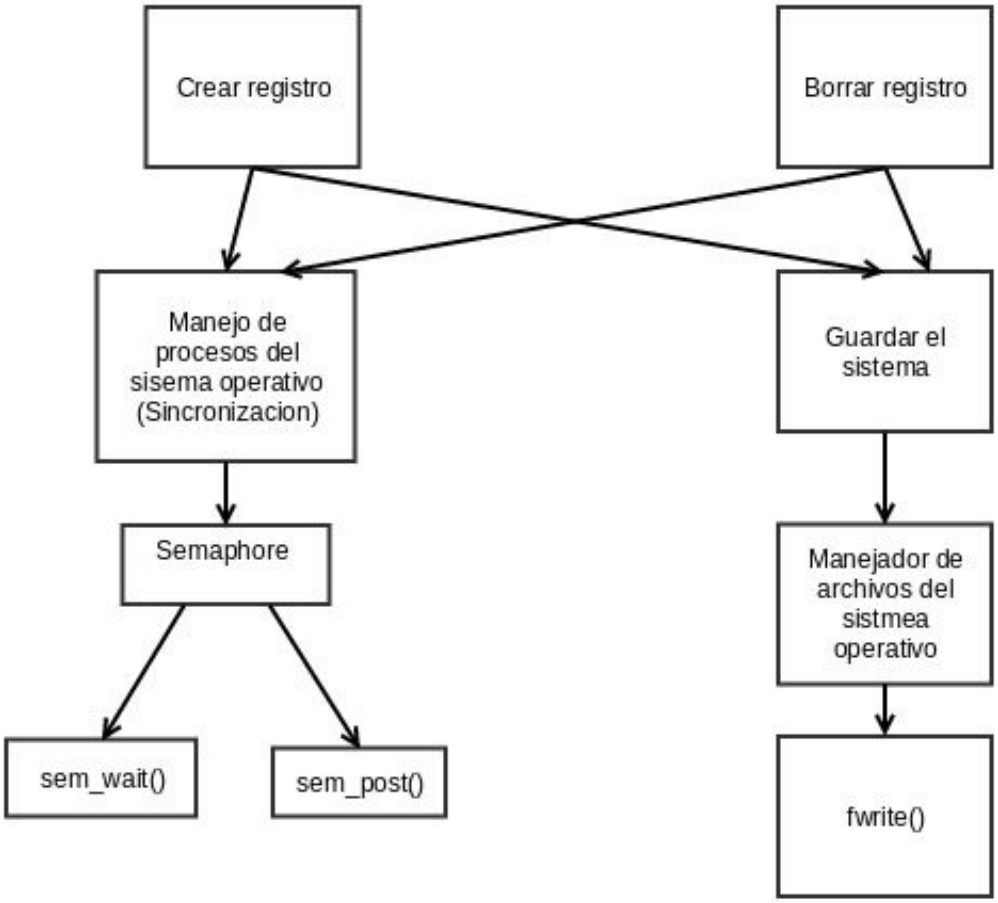
Servidor.

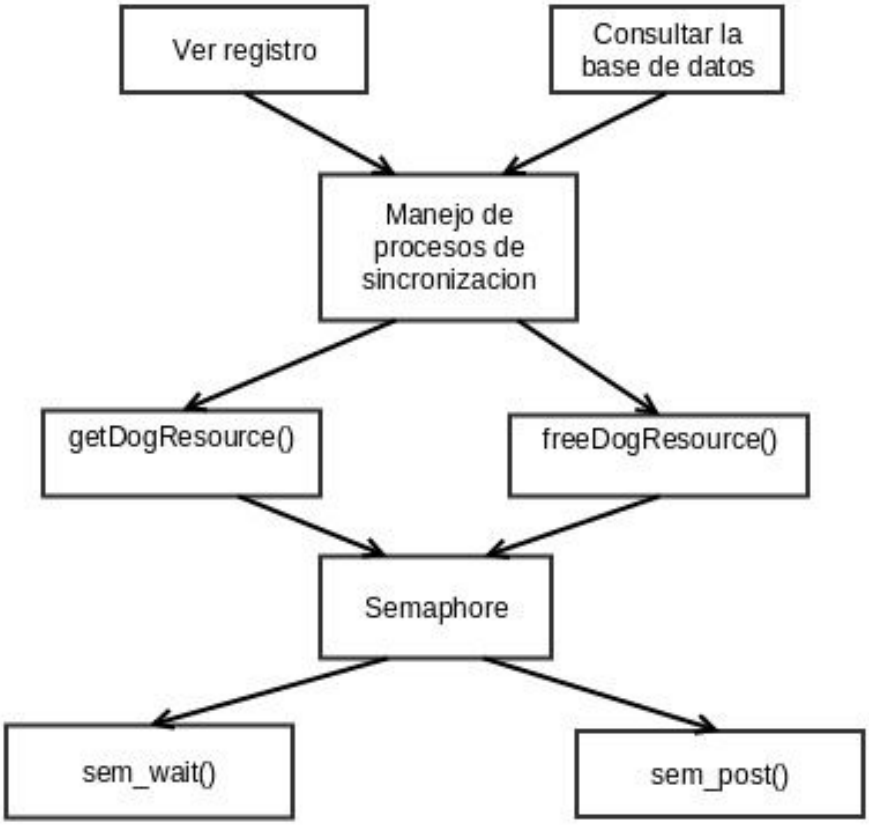


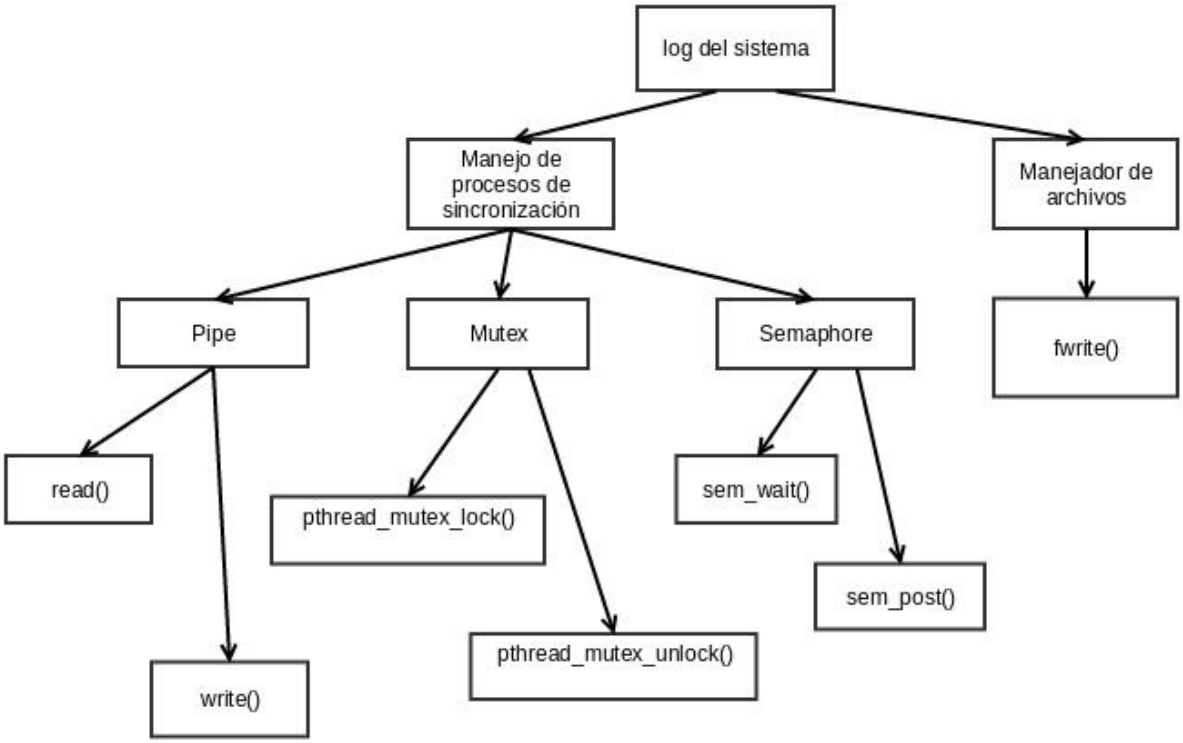



Configurar Socket
Gestor de red del sistema operativo
Socket()
Bind()
Listen

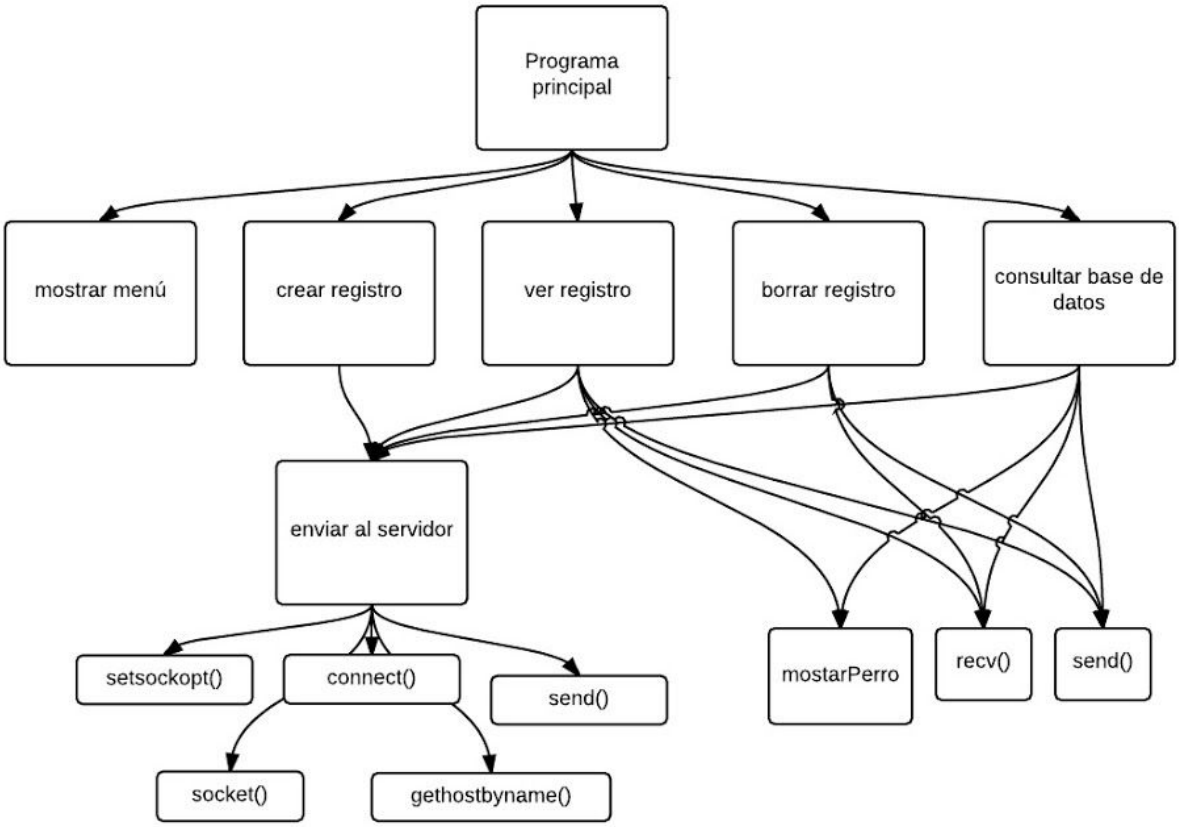

Manejador de clientes
log del sistema
Crear registro
ver registro
borrar registro
Consultar la base de datos
Gestor de red
send()
recv()


Crear registro
Borrar registro
Manejo de procesos del sisema operativo (Sincronizacion)
Guardar el sistema
Semaphore
Manejador de archivos del sistmea operativo
sem_wait()
sem_post()
fwrite()


Ver registro
Consultar la base de datos
Manejo de procesos de sincronizacion
getDogResource()
freeDogResource()
Semaphore
sem_wait()
sem_post()


log del sistema
Manejo de procesos de sincronización
Manejador de archivos
Pipe
Mutex
Semaphore
fwrite()
read()
pthread_mutex_lock()
sem_wait()
sem_post()
pthread_mutex_unlock()
write()

Cliente


Programa principal
mostrar menú
crear registro
ver registro
borrar registro
consultar base de datos
enviar al servidor
setsockopt()
connect()
send()
socket()
gethostbyname()
mostarPerro
recv()
send()