Practica 3

Sergio Alejandro Diaz Pinilla
seradiazpin@unal.edu.co

Osmar Alejandro Castillo Lancheros
oacastillol@unal.edu.co

March 2015

1. Introducción

Se crearon dos programas cliente-servidor, que permiten realizar las siguientes funciones insertar, leer, borrar y buscar registros. Para esto se usaron los conceptos de comunicación de procesos mediante mensajes. El Servidor tiene una capacidad de 32 clientes, para manejar los clientes simultáneamente en la practica anterior se creo un proceso hijo del servidor con un fork(). En este proceso se atiende al cliente.

En esta practica se cambio esa funcionalidad por una en la cual los clientes se manejan por medio de hilos, al hacer esto se debe tener en cuenta la sección critica que tiene este programa la cual es cuando se abre el archivo donde se guardan los daros, para la sincronización de esta sección se implementaron tres métodos semáforos, mutex y tuberías. El cliente solo se encarga de imprimir datos e interfaz y de enviar y recibir datos, todos los procesos importantes se realizan en el servidor.

2. Practica 2

Codigo mas importante es:

- int conectar() : Esta función crea el socket y lo rellena, el parámetro que recibe es la dirección IP.
- void ingresar()
  - En el cliente se cargan los datos en un dato tipi dogType y es enviado con sendPerro() a el servidor.
  - En el servidor recibe el dato con recvPerro() y lo guarda en el archivo dogData.dat.Registra la acción con writeLog()
- void leer()

- En el cliente se recibe el número de registros que hay, se envía el registro que se desea ver luego espera la confirmación de la existencia del registro, si aun existe por ultimo se recibe el dato con recvPerro(), pero si ya no existe se vuelve a empezar el proceso, si no existen registros se muestra un mensaje advirtiendo que no existen registros y no se hace nada mas.
- En el servidor se calcula el número de registros que tiene el archivo dataDogs.dat y se envía, posteriormente se recibe el numero de registro y se lee del archivo, si aun existe se envía un numero confirmando la existencia al cliente por ultimo se envía el dato con sendPerro(), pero si el registro no se encuentra se envía el dato de confirmación de no existencia y se empieza a hacer el proceso de nuevo, pero si no existen registros solo envía el dato de confirmación y termina.

- void buscar()

  - En el cliente se recibe el número de registros que hay, se envía el nombre del perro que se desea buscar y se recibe una variable mientras el servidor recorre todo el archivo,si la variable indica que se encontró uno recibe el registro con recvPerro() luego lo imprime con imprimirPerro(), continua recibiendo la variable de continuación hasta que reciba el dato que indica que el archivo ya termino, por ultimo recibe la cantidad de registros encontrados imprime este número .
  - En el servidor se calcula el numero de registros que tiene el archivo dataDogs.dat y se envía, posteriormente se recibe el nombre del perro y se busca en el archivo enviando una variable que toma el valor de 0 si aun no termina de recorrer el archivo, 1 si encuentra un registro que coincide envía este registro con sendPerro() y -1 si ya termino de recorrer el archivo por ultimo se envía el numero de registros que encontró.

- void borrar();

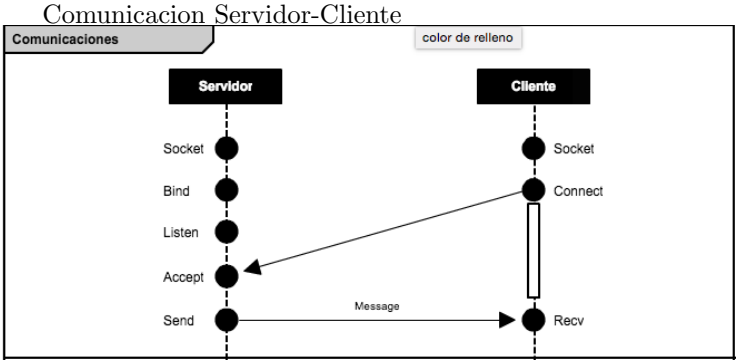
  - En el cliente se recibe el numero de registros que hay, se envía el registro que se desea borrar, luego recibe un numero de confirmación de la existencia del registro,por ultimo se recibe el recibe el dato que se borro con recvPerro(), si el registro no existe vuelve a empezar la función con los datos actuales, si no hay registros muestra un mensaje de advertencia.
  - En el servidor se calcula el numero de registros que tiene el archivo dataDogs.dat y se envía, posteriormente se recibe el número de registro, se confirma su existencia se envía esta confirmación al cliente se envía el dato que se borro con sendPerro(), por ultimo se borra del archivo, y se registra la operación con writeLog().

- void recvPerro() :Este método recibe un dato de tipo perro ya sea al cliente o al servidor, dependiendo de donde se llama y el numero identificador del socket a donde se recibirá.
- void sendPerro() : Este método envía un dato de tipo perro ya sea al cliente o al servidor, dependiendo de donde se llama y el número identificador del socket a donde se enviara.

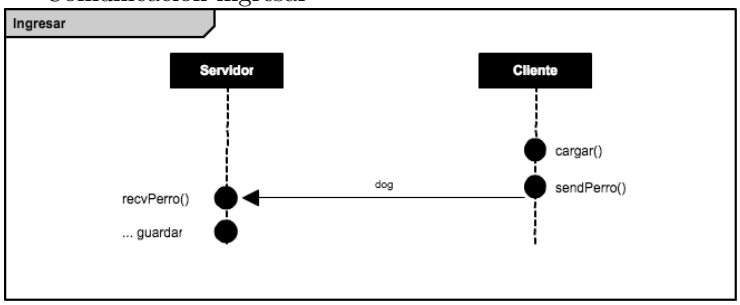
Además de estas funciones se implementaron algunas otras funciones complementarias las cuales son para impresión del menú y datos.

3. Diagramas Practica 2

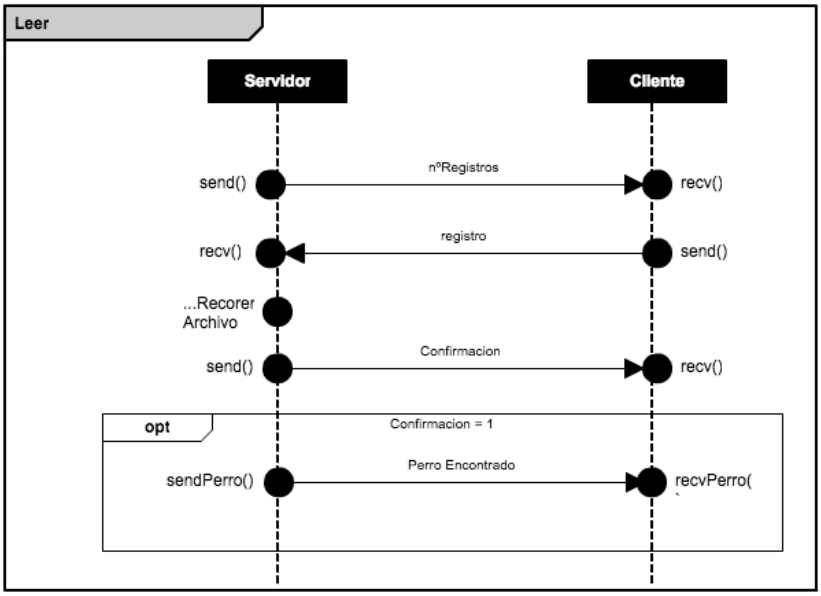
Comunicacion Servidor-Cliente



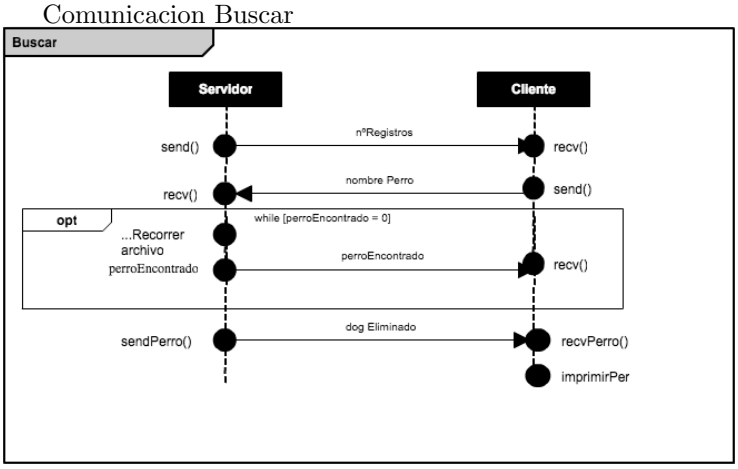
Comunicacion ingresar



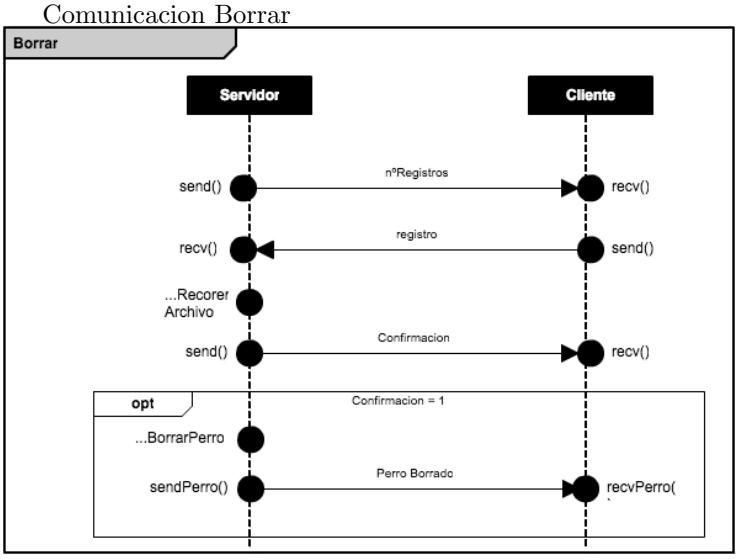
Comunicacion leer


Leer
Servidor
Cliente
send()
nºRegistros
recv()
recv()
registro
send()
...Recorer
Archivo
send()
Confirmacion
recv()
opt
Confirmacion = 1
sendPerro()
Perro Encontrado
recvPerro(

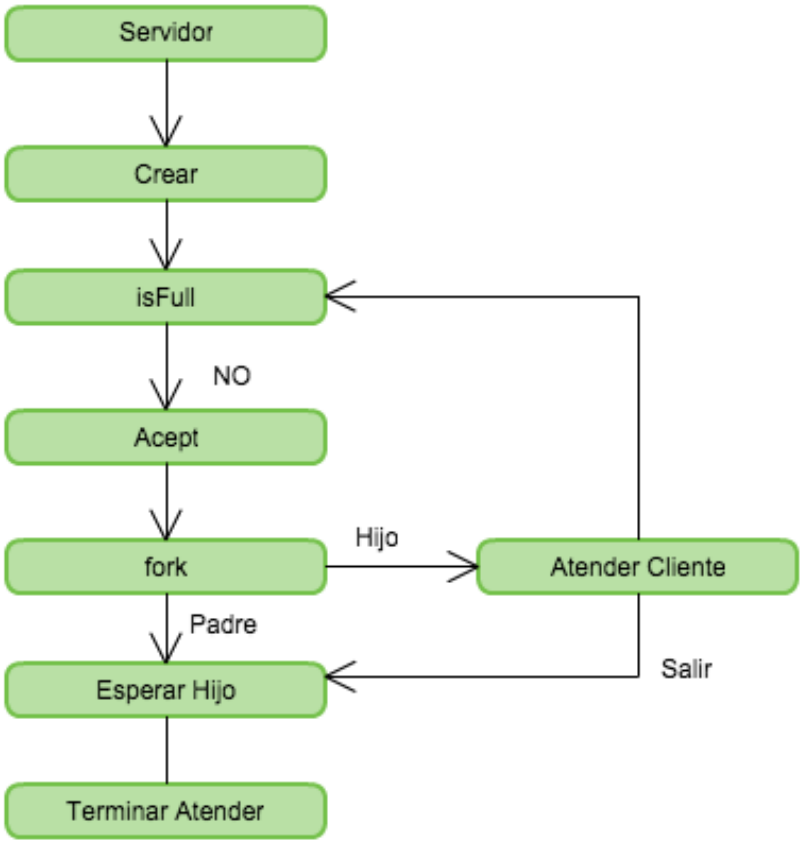
Comunicacion Buscar



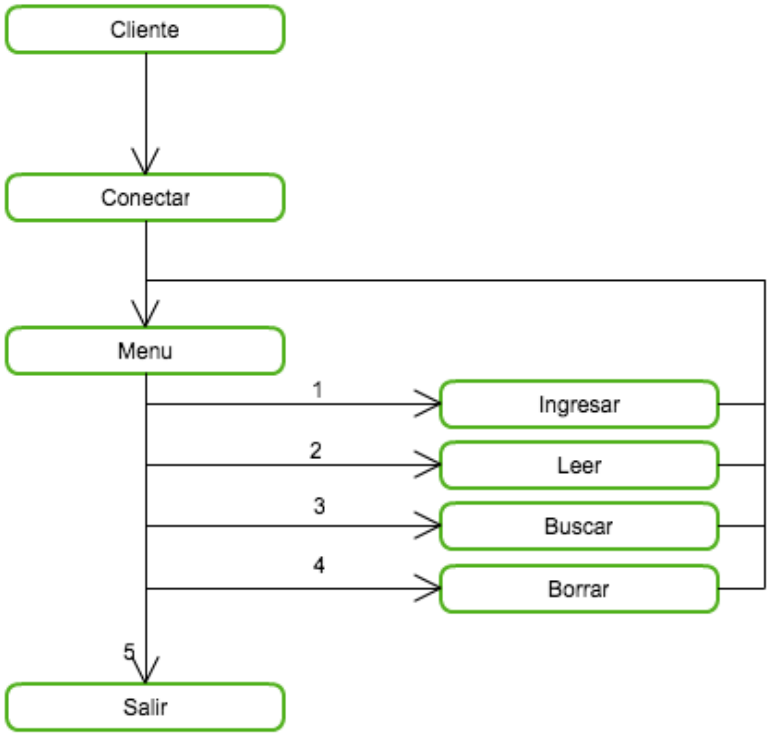
Comunicacion Borrar



Bloques Servidor



Bloques Cliente



4. Practica 3

El funcionamiento consiste en dividir la creación y destrucción de los hilos en dos, para esto se usaron dos hilos. Codigo importante:

- void *crearClientes(int *serverId) Función encargada de esperar la conexión de clientes y crear el hilo que se encarga de atender las solicitudes de este.
- void *eliminarClientes() Función que va a a esperar a los hilos de los clientes que ya se desconectaron, esta va a ser ejecutada por un hilo
- struct hijos: Estructura que fue necesaria crear para el paso de parámetros a los hilos, además de esto se creo un arreglo de hilos para poder asignar un hilo por cada cliente que se conecte al servidor.
- sincronización: El manejo de los métodos se hace por medio de un argumento que se le da al servidor ya sea pipe, mutex o semaphore para

elegir el método, estos métodos se encuentran en la funciones que están involucradas con la sección critica.

5. Diagramas Practica 3

Los diagramas del comunicación entre el cliente y servidor son iguales pero cambia el diagrama de bloques.