Ayuda TP ServidorSwitch

ServidorSwitch.c en Linux Debian en Lenguaje C
ClienteServidorSwitch en Windows 7 en Lenguaje C#

El ServidorSwitch es un pequeño programa en lenguaje C en Linux.
Es Servidor en el sentido que espera conexiones de Clientes que necesitan acceder a través de él a servidores de Bases de Datos, también se puede decir que el ServidorSwitch es un cliente de los servidores de Bases de Datos Postgresql, MySql y FireBird (no implementado en este ejemplo), porque este ServidorSwitch tiene que pedirle a los servidores de Bases de Datos, lo que su Cliente le pidió a él.

Alguien puede preguntar ¿Por qué no comunicamos directamente al Cliente del ServidorSwitch con el servidor de Bases de Datos? Así evitamos programar el ServidorSwitch, la respuesta es simple "porque estamos aprendiendo".

En el ejemplo, tenemos dos máquinas virtuales, una máquina tiene un Linux Debian, y ahí vamos a ejecutar el ServidorSwitch y un motor de Bases de Datos Postgrsql, la otra máquina tiene un Windows 7, y ahí vamos a ejecutar un Cliente del ServidorSwitch y un motor de Bases de Datos MySql.
Tenemos esta estructura de máquinas virtuales para evitar levantar cuatro máquinas virtuales, que demandaría una tecnología de hardware superior a la disponible en esta cuarentena.

El ServidorSwitch se escribió en lenguaje C y el Cliente del ServidorSwitch se escribió en lenguaje C#.

El ejemplo se puede implementar de distintas formas o estructuras de software, en este caso se crea un programa objeto para el acceso a Postgresql, un programa objeto para el acceso a MySql y un programa objeto para el ServidorSwitch y con los programas objeto se obtendrá un programa ejecutable llamado Servidor.

La tecnología de comunicación utilizada es la de sockets que se hace transparente para el programador en las funciones de acceso a las Bases de Datos implementadas en AccesoPostgresql.c y AccesoMysql.c pero no se hace transparente en el ServidorSwitch.c donde se realiza la comunicación con sus Clientes.

Códigos en Lenguaje C

AccesoPostgresql.c (acceso al motor Postgresql)

```
#include <string.h>
#include <libpq-fe.h>  //Hay que descargarla
#include <sys/types.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>

//Ubicacion en la red del servidor Postgresql
#define IPPostgresql "192.168.1.30" // IP de la máquina virtual Linux Debian donde esta
// en ejecucion el motor postgresql
#define PuertoPostgresql "5432"

void funcionPostgresql(char bufnomb[],char bufquery[],char respuesta[])
{
  PGconn *conn;
  PGresult *res;
  int i,j;
  conn =
PQsetdbLogin(IPPostgresql,PuertoPostgresql,NULL,NULL,bufnomb,"postgres","clave");
  if (PQstatus(conn) != CONNECTION_BAD)
```

```
   {
    res = PQexec(conn, bufquery);
    if (res != NULL && PGRES_TUPLES_OK == PQresultStatus(res))
      {
       for (i = 0 ; i <= PQntuples(res)-1;  i++)
        {
          for (j = 0 ; j < PQnfields(res); j++)
             {
              strcat(respuesta,PQgetvalue(res,i,j));
              strcat(respuesta,"\t");
          }
          strcat(respuesta,"\n");
        }
        strcat(respuesta,"\0");
        PQclear(res);
      }
   }
   else
   {
    strcat(respuesta,"Fallo conexion postgresql\n");
   }
   PQfinish(conn);
}
```

AccesoMysql.c (acceso al motor Mysql)

```
#include "/home/jromer/Escritorio/mysql/mysql-connector-c-6.1.11-linux-glibc2.12-
i686/include/mysql.h"

#include <sys/types.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>

#define IPMysql "192.168.1.40" //Ubicación del motor MySql
#define PuertoMysql "3306"

void funcionMysql(char dabasename[],char query[],char respuesta[])
{
      MYSQL *conn; // variable de conexión para MySQL
          MYSQL_RES *res; // variable que contendra el resultado de la consuta
          MYSQL_ROW row; // variable que contendra los campos por cada registro
consultado
          char *server = IPMysql ; //direccion del motor MySql
          char *user = "jromer"; //usuario para consultar la base de datos
          char *password = "xxxxxxxxx"; // contraseña para el usuario
      char *database = dabasename; //nombre de la base de datos a consultar
      conn = mysql_init(NULL); //inicializacion

      memset(respuesta,0,1024);

          /* conectar a la base de datos */
          if (!mysql_real_connect(conn, server, user, password, database, 0, NULL, 0))
          { /* definir los parámetros de la conexión antes establecidos */
                    sprintf(respuesta, "%s\n", mysql_error(conn)); /* si hay un error definir cual
fue dicho error */
  //            strcat(respuesta,"\0");
          }
      else
       {
             /* enviar consulta SQL */
         if (mysql_query(conn, query))
          { /* definicion de la consulta y el origen de la conexion */
                    sprintf(respuesta, "%s\n", mysql_error(conn));
```

```
            strcat(respuesta,"\0");
              }
          else
           {
             res = mysql_use_result(conn);
             if (res == NULL)
              {strcpy(respuesta,"ok query\0");}
             else
             {
              int num_attrib = mysql_num_fields(res);
                  while ((row = mysql_fetch_row(res)) != NULL)
                 {
                  /* recorrer la variable res con todos los registros obtenidos para su uso */
                      // printf("%s\t%s\t%s \n", row[0],row[1]); /* la variable row se convierte
en un arreglo por el numero de campos que hay en la tabla */
                  int i ;
                  for (i = 0 ; i < num_attrib ; i++)
                   {
                     strcat(respuesta,row[i]);
                     strcat(respuesta,"  ");
                   }
                  strcat(respuesta,"\n");
                 }
              strcat(respuesta,"\0");
                      mysql_free_result(res);
             }
           }
         }
      /* se libera la variable res y se cierra la conexión */
          mysql_close(conn);
}
```

ServidorSwitch

El ServidorSwitch es un servidor concurrente implementado con procesos hijos (fork()) para la atención de los Clientes, no está resuelto en este ejemplo la eliminación de los procesos zombies.

El ServidorSwitch también implementa un archivo de Log en el File System con el historial de las consultas que hicieron los Clientes.

ServidorSwitch.h

```
#define IPSerBD "192.168.1.30"
#define PuertoSerBD 6666

#define IPPostgrsql "192.168.1.30"
#define PuertoPostgresql "5432"

#define IPMySql "192.168.1.40"
#define PuertoMySql "3306"

char * DBnombres[1024];
char  DBservidor[1024];

void funcionPostgresql(char [],char [],char []);
void funcionMysql(char [],char [],char []);
char buscarServidor(char * ,int);

void DameFechaMaquina( char *);
void DameHoraMaquina( char *);
void Log(char * ,char *);
```

ServidorSwitch.c

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

#include "ServidorSwitch.h"

void ObtenerNombreBaseDatos(char query[],char databasename[]) ;

int main(int argc, char *argv[])
```

```
{
  struct sockaddr_in s_sock,c_sock;
  int idsocks,idsockc;
  socklen_t lensock = sizeof(struct sockaddr_in);
  idsocks = socket(AF_INET, SOCK_STREAM, 0);
  printf("idsocks %d\n",idsocks);

  s_sock.sin_family      = AF_INET;
  s_sock.sin_port        = htons(PuertoSerBD);
  s_sock.sin_addr.s_addr = inet_addr(IPSerBD);
  memset(s_sock.sin_zero,0,8);

  printf("bind %d\n", bind(idsocks,(struct sockaddr *) &s_sock,lensock));
  printf("listen %d\n",listen(idsocks,5));

  while(1)
   {
     printf("esperando conexion\n");
     idsockc = accept(idsocks,(struct sockaddr *)&c_sock,&lensock);
     if(idsockc != -1)
      {
        if (!fork())
        {
          char query[1024];
          char databasename[1024];
          int nb1;
          printf("conexion aceptada desde el cliente\n");
          nb1 = read(idsockc,query,1024);
          query[nb1]  = '\0' ;
          if(query[0] == 'm')
             {
                 ObtenerNombreBaseDatos(query,databasename) ;
                 printf(".......recibido del cliente %d : %s  %s\n",idsockc,databasename,query);
                 char respuesta [1024];
                 memset(respuesta,0,1024);
                 funcionMysql(databasename, query ,respuesta);
                 write(idsockc,respuesta,1024);
                 Log(databasename,query);
              }
          if(query[0] == 'p')
             {
                 ObtenerNombreBaseDatos(query,databasename) ;
                 printf(".......recibido del cliente %d : %s  %s\n",idsockc,databasename,query);
                 char respuesta [1024];
```

```
                memset(respuesta,0,1024);
                funcionPostgresql(databasename, query ,respuesta);
                write(idsockc,respuesta,1024);
                Log(databasename,query);
              }
            printf("conexion finalizada con el cliente\n");
            close(idsockc);
            exit(0);
          }
        }
      else
        {
          printf("conexion rechazada %d \n",idsockc);
        }
    }
  return 0 ;
}

void ObtenerNombreBaseDatos(char query[],char databasename[])
{
  char queryaux[1024] ;
  int i ;
  for (i = 0 ; query[i] != ':' ; i++);
  int j , x = 0 ;
  for (j = i+1 ; i < strlen(query) && query[j] != ':' ; j++)
  {
    databasename[x] = query[j] ;
    x++ ;
  }
  databasename[x] = '\0';
  j++ ;
  int p ;
  x = 0 ;
  for (p = j ; p < strlen(query) && query[p] != '\0' ; p++)
  {
    queryaux[x] = query[p] ;
    x++ ;
  }
  queryaux[x] =  '\0';
  strcpy(query,queryaux);
}
```

```
void Log(char * db,char * sql)
{
    char Fecha[128] ;
    char Hora[128] ;
    DameFechaMaquina(Fecha);
    DameHoraMaquina(Hora);
    int fd = open("log",O_CREAT|O_WRONLY,0666) ;
    lseek(fd,0,2);
    write(fd,Fecha,strlen(Fecha));
    write(fd," ",1);
    write(fd,Hora,strlen(Hora));
    write(fd," ",1);
    write(fd,db,strlen(db));
    write(fd," ",1);
    write(fd,sql,strlen(sql));
    write(fd,"\n",1);
    printf("%s %s %s %s\n",Fecha,Hora,db,sql);
    close(fd);
}

void DameFechaMaquina(char * Fecha)
{
      time_t tiempo = time(0);
      struct tm *tlocal = localtime(&tiempo);
      strftime(Fecha,128,"%d/%m/%y",tlocal);
}

void DameHoraMaquina(char * Hora)
{
      time_t tiempo = time(0);
      struct tm *tlocal = localtime(&tiempo);
      strftime(Hora,128,"%H:%M:%S",tlocal);
}
```

Script compilar

```
gcc AccesoPostgresql.c -Wall -g -c -lpq -I/usr/include/postgresql/
gcc AccesoMysql.c -Wall -g -c -L/usr/lib/mysql -lmysqlclient
gcc ServidorSwitch.c -Wall -g -c
gcc -Wall -g AccesoPostgresql.o AccesoMysql.o ServidorSwitch.o -o Servidor -
L/usr/lib/mysql -lmysqlclient -lpq -I/usr/include/postgresql/
```

Ejecutar

$ ./Servidor

ClienteServidorSwitch (aquí pueden encontrar los query relacionados con la metadata de las bases de datos)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;

namespace ClienteServidorSwitch
{
   public partial class Form1 : Form
   {
      string IPServidorSwitch = "192.168.1.30";
      int PuertoServidorSwitch = 6666;
      public Form1()
      {
         InitializeComponent();
         textBox2.Visible = false;
      }

      //probar conexion con el servidor 192.168.1.30
      private void button1_Click(object sender, EventArgs e)
      {
         PruebaConexion();
      }

      //Query a MySql
      private void button3_Click(object sender, EventArgs e)
      {
         RealizarQuery("m:");
      }
```

```
        //metadata bases de datos MySql
        private void button5_Click(object sender, EventArgs e)
        {
            ConsultaBasesDatos("m:INFORMATION_SCHEMA:SELECT SCHEMA_NAME FROM
SCHEMATA");
        }

        //metadata tablas
        private void button6_Click(object sender, EventArgs e)
        {
            ConsultaTablas("m:INFORMATION_SCHEMA:SELECT table_name  FROM tables
WHERE table_schema = '" + txBaseDatos.Text.Trim() + "'");
        }

        //Atributos MySql
        private void button7_Click(object sender, EventArgs e)
        {
            ConsultaAtributos("m:INFORMATION_SCHEMA:SELECT COLUMN_NAME FROM
COLUMNS WHERE table_name = '" + txTabla.Text.Trim() + "'");
        }

        private void comboBox3_SelectedIndexChanged(object sender, EventArgs e)
        {
      //      textBox1.Text = comboBox3.SelectedItem.ToString().Trim();
        }

        private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            txBaseDatos.Text = listBox1.SelectedItem.ToString().Trim();
        }

        private void listBox2_SelectedIndexChanged(object sender, EventArgs e)
        {
            txTabla.Text = listBox2.SelectedItem.ToString().Trim();
        }

        private void listBox3_SelectedIndexChanged(object sender, EventArgs e)
        {
            textBox1.Text = listBox3.SelectedItem.ToString().Trim();
        }

        //Bases de datos Postgresql
        private void button8_Click(object sender, EventArgs e)
        {
```

```
        ConsultaBasesDatos("p:postgres:SELECT datname FROM pg_database WHERE
datistemplate = false;");
    }


    //Tablas de la base de datos postgresql
    private void button9_Click(object sender, EventArgs e)
    {
        ConsultaTablas("p:" + txBaseDatos.Text.Trim() + ":select table_name FROM
information_schema.tables where table_schema = 'public' order by table_name");
    }

    //Atributos Postgresql
    private void button10_Click(object sender, EventArgs e)
    {
        ConsultaAtributos("p:" + txBaseDatos.Text.Trim() + ":SELECT attname FROM
pg_catalog.pg_attribute " +
            " inner join pg_catalog.pg_class on  pg_catalog.pg_class.relfilenode =
pg_catalog.pg_attribute.attrelid " +
      " where pg_catalog.pg_class.relname = '" + txTabla.Text.Trim() + "'");
    }

    private void PG_BaseDeDatos_Click(object sender, EventArgs e)
    {
        ConsultaBasesDatos("p:postgres:SELECT datname FROM pg_database WHERE
datistemplate = false;");
    }

    //Query Postgresql
    private void button2_Click(object sender, EventArgs e)
    {
        RealizarQuery("p:");
    }

    //Query Postgresql
    private void PG_Query_Click(object sender, EventArgs e)
    {
        RealizarQuery("p:");
    }

    private void PG_Tablas_Click(object sender, EventArgs e)
    {
        ConsultaTablas("p:" + txBaseDatos.Text.Trim() + ":select table_name FROM
information_schema.tables where table_schema = 'public' order by table_name");
```

```
    }

    private void PG_Atributos_Click(object sender, EventArgs e)
    {
       ConsultaAtributos("p:" + txBaseDatos.Text.Trim() + ":SELECT attname FROM
pg_catalog.pg_attribute " +
          " inner join pg_catalog.pg_class on  pg_catalog.pg_class.relfilenode =
pg_catalog.pg_attribute.attrelid " +
    " where pg_catalog.pg_class.relname = '" + txTabla.Text.Trim() + "'");
    }

    public void RealizarQuery(string SGBD)
    {
       if ( SGBD[0] == 'm')
        label4.Text = "Query a MySql";
       else
          label4.Text = "Query a Postgresql";

       listBox4.Items.Clear();
       textBox2.Text = " ";
       textBox2.Visible = false;
       byte[] bytes = new byte[1024];
       Socket miSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
       IPEndPoint Direccion = new IPEndPoint(IPAddress.Parse(IPServidorSwitch),
PuertoServidorSwitch);
       try
       {
          miSocket.Connect(Direccion);
          if (textBox1.Text.Length == 0)
          {
             textBox1.Text = "?????";
          }

          byte[] query = Encoding.ASCII.GetBytes(SGBD + txBaseDatos.Text.Trim() + ":" +
textBox1.Text.Trim());
          int bytesSentq = miSocket.Send(query);
          int bytesRec = miSocket.Receive(bytes);
          textBox2.Text = Encoding.ASCII.GetString(bytes, 0, bytesRec);
          miSocket.Close();

          int i; listBox4.Text = "Seleccionar";
          for (i = 0; i < textBox2.Lines.Length; i++)
             listBox4.Items.Add(textBox2.Lines[i]);
```

```
        }
        catch (Exception error)
        {
            textBox2.Visible = true;
            textBox2.Text = "Error: {0}" + error.ToString();
        }
    }

    private void MY_Query_Click(object sender, EventArgs e)
    {
        RealizarQuery("m:");
    }

    public void ConsultaBasesDatos(string ConsultaBasesDatosPostgresql)
    {
        if (ConsultaBasesDatosPostgresql[0] == 'p')
                label1.Text = "Bases de Datos Postgresql";
        else
            label1.Text = "Bases de Datos MySql";

        listBox1.Items.Clear();
        listBox2.Items.Clear();
        listBox3.Items.Clear();
        listBox4.Items.Clear();

        txBaseDatos.Text = " ";
        txTabla.Text = " ";
        textBox2.Text = " ";
        textBox2.Visible = false;
        textBox1.Text = ConsultaBasesDatosPostgresql;

        byte[] bytes = new byte[1024];
        Socket miSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
        IPEndPoint Direccion = new IPEndPoint(IPAddress.Parse(IPServidorSwitch),
PuertoServidorSwitch);
        try
        {
            miSocket.Connect(Direccion);
            byte[] query = Encoding.ASCII.GetBytes(textBox1.Text.Trim());
            int bytesSentq = miSocket.Send(query);
            int bytesRec = miSocket.Receive(bytes);
            textBox2.Text = Encoding.ASCII.GetString(bytes, 0, bytesRec);
            miSocket.Close();
```

```
            int i;  listBox1.Text = "Seleccionar";
            for (i = 0; i < textBox2.Lines.Length; i++)
               listBox1.Items.Add(textBox2.Lines[i]);
         }
         catch (Exception error)
         {
            textBox2.Visible = true;
            textBox2.Text = "Error: {0}" + error.ToString();
         }
      }

      public void ConsultaTablas(string QueryTablas)
      {
         label2.Text = "Tablas";
         listBox2.Items.Clear();
         listBox3.Items.Clear();
         listBox4.Items.Clear();

         textBox2.Text = " ";
         textBox2.Visible = false;
         textBox1.Text = QueryTablas;

         byte[] bytes = new byte[1024];
         Socket miSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
         IPEndPoint Direccion = new IPEndPoint(IPAddress.Parse(IPServidorSwitch),
PuertoServidorSwitch);
         try
         {
            miSocket.Connect(Direccion);
            byte[] query = Encoding.ASCII.GetBytes(textBox1.Text.Trim());
            int bytesSentq = miSocket.Send(query);
            int bytesRec = miSocket.Receive(bytes);
            textBox2.Text = Encoding.ASCII.GetString(bytes, 0, bytesRec);
            miSocket.Close();

            int i; listBox2.Text = "Seleccionar";
            for (i = 0; i < textBox2.Lines.Length; i++)
               listBox2.Items.Add(textBox2.Lines[i]);
         }
         catch (Exception error)
         {
            textBox2.Visible = true;
```

```
            textBox2.Text = "Error: {0}" + error.ToString();
        }

    }

    private void MY_BasesDeDatos_Click(object sender, EventArgs e)
    {
        ConsultaBasesDatos("m:INFORMATION_SCHEMA:SELECT SCHEMA_NAME FROM
SCHEMATA");
    }

    public void ConsultaAtributos(string Atributos)
    {

        label3.Text = "Atributos";
        listBox3.Items.Clear();
        listBox4.Items.Clear();
        textBox2.Text = " ";
        textBox2.Visible = false;
        textBox1.Text = Atributos;
        byte[] bytes = new byte[1024];
        Socket miSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
        IPEndPoint Direccion = new IPEndPoint(IPAddress.Parse(IPServidorSwitch),
PuertoServidorSwitch);
        try
        {
            miSocket.Connect(Direccion);
            byte[] query = Encoding.ASCII.GetBytes(textBox1.Text.Trim());
            int bytesSentq = miSocket.Send(query);
            int bytesRec = miSocket.Receive(bytes);
            textBox2.Text = Encoding.ASCII.GetString(bytes, 0, bytesRec);
            miSocket.Close();

            int i; listBox3.Text = "Seleccionar";
            for (i = 0; i < textBox2.Lines.Length; i++)
                listBox3.Items.Add(textBox2.Lines[i]);
        }
        catch (Exception error)
        {
            textBox2.Visible = true;
            textBox2.Text = "Error: {0}" + error.ToString();
        }
    }
```

```
        private void MY_Tablas_Click(object sender, EventArgs e)
        {
            ConsultaTablas("m:INFORMATION_SCHEMA:SELECT table_name  FROM tables
WHERE table_schema = '" + txBaseDatos.Text.Trim() + "'");
        }

        private void MY_Atributos_Click(object sender, EventArgs e)
        {
            ConsultaAtributos("m:INFORMATION_SCHEMA:SELECT COLUMN_NAME FROM
COLUMNS WHERE table_name = '" + txTabla.Text.Trim() + "'");
        }

        public void PruebaConexion()
        {
            textBox2.Text = " ";
            textBox2.Visible = false;
            Socket miSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
            IPEndPoint Direccion = new IPEndPoint(IPAddress.Parse(IPServidorSwitch),
PuertoServidorSwitch);
            textBox2.Visible = true;
            try
            {
                miSocket.Connect(Direccion); // Conectamos
                textBox2.Text = "Conectado con éxito";
                miSocket.Close();
            }
            catch (Exception error)
            {
                textBox2.Text = "Error: {0}" + error.ToString();
            }
        }

        private void ProbarConexion_Click(object sender, EventArgs e)
        {
            PruebaConexion();
        }
    }
}
```