**Ayuda TP Servidor Switch de acceso a Postgres y FireBird**

**Para Linux Debian en Lenguaje C**

**Conexión de acceso al motor Postgres**

**Hay que instalar libpq**

$ sudo apt-get update

$ sudo apt-get install libpq-dev

**Código ejemplo de uso, para recuperar datos de una tabla.**

/\*

Línea de compilación

$ gcc Servidor.c -o Ser -lpq -I/usr/include/postgresql/

\*/

#include <libpq-fe.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <sys/types.h>

#include <string.h>

#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

#include <netdb.h>

#define IPSerBD "127.0.0.1"

#define PuertoSerBD 6666

#define IPPostgrsql "127.0.0.1"

#define PuertoPostgresql "5432"

void funcionPostgresql(int);

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

{

struct sockaddr\_in s\_sock,c\_sock;

int idsocks,idsockc;

int 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)

{

printf("conexion aceptada desde el cliente\n");

char buf[32];

int nb;

nb = read(idsockc,buf,32);

buf[nb]='\0';

printf(".......recibido del cliente %d : %s\n",idsockc,buf);

funcionPostgresql(idsockc);

close(idsockc);

}

else

{

printf("conexion rechazada %d \n",idsockc);

}

}

return 0 ;

}

void funcionPostgresql(int idsockc)

{

PGconn \*conn;

PGresult \*res;

int i,j;

char respuesta[1024];

memset(respuesta,0,1024);

conn = PQsetdbLogin(IPPostgrsql,PuertoPostgresql,NULL,NULL,"prueba","postgres","clavedepostgres");

if (PQstatus(conn) != CONNECTION\_BAD)

{

res = PQexec(conn, "select codigo, nombre from tabla1 order by codigo");

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");

// printf("%s\t",PQgetvalue(res,i,j));

}

strcat(respuesta,"\n");

// printf("\n");

}

strcat(respuesta,"\0");

PQclear(res);

}

}

PQfinish(conn);

write(idsockc,respuesta,1024);

}

**Conexión de acceso al motor FireBird**

**Hay que instalar libfb se descarga de**

[**http://sourceforge.net/projects/libfb/**](http://sourceforge.net/projects/libfb/)

**donde hay una guía de instalación**

**Código ejemplo de uso, para recuperar datos de una tabla.**

/\*

Línea de compilación

$ gcc -Wall -o ser -I /home/jromer/Escritorio/ConexionPostgres -L /home/jromer/Escritorio/ConexionPostgres -lfb Servidor.c

\*/

#include <libfb.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <sys/types.h>

#include <string.h>

#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

#include <netdb.h>

#define IPSerBD "127.0.0.1"

#define PuertoSerBD 6666

#define IPFirebird "127.0.0.1"

#define PuertoFirebird "3050"

void funcionFirebird(int);

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

{

struct sockaddr\_in s\_sock,c\_sock;

int idsocks,idsockc;

//int lensock = sizeof(struct sockaddr\_in);

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)

{

printf("conexion aceptada desde el cliente\n");

char buf[32];

int nb;

nb = read(idsockc,buf,32);

buf[nb]='\0';

printf(".......recibido del cliente %d : %s\n",idsockc,buf);

funcionFirebird(idsockc);

close(idsockc);

}

else

{

printf("conexion rechazada %d \n",idsockc);

}

}

return 0 ;

}

void funcionFirebird(int idsockc)

{

/\*\*

\* Programa de ejemplo de uso de libreria libfb

\* Se conecta a la base de datos /var/lib/firebird/2.5/data/isft.fdb

\* del servidor local de firebird, realiza un query, muestra su resultado

\* y se desconecta

\*

\* \*/

extern int FB\_SHOW\_MESSAGES; // 1 activa/ 0 desactiva los mensajes de libreria FB

FB\_SHOW\_MESSAGES = 1; // desactivo mensajes de libreria libfb

fb\_db\_info dbinfo;

strcpy(dbinfo.user,"sysdba");

strcpy(dbinfo.passw,"masterkey");

strcpy(dbinfo.dbname,"localhost:/var/lib/firebird/2.5/data/prueba.fdb");

strcpy(dbinfo.role,"sysdb");

if (fb\_do\_connect(&dbinfo)) { // me conecte!

printf("Me conecte!\n");

query myquery;

fb\_init(&myquery); // atencion! myquery debe estar inicializado

printf("Ejecuto Query!\n");

if ( fb\_do\_query(&dbinfo,1,"SELECT CODIGO, DESCR FROM TABLA1 ORDER BY DESCR ASC",

onDoGenericQuery,&myquery) ) {

printf("Recupero datos de Query!\n");

rquery \*q = myquery.top; // primer tupla del query

char \*\*cols;

while (q) {

cols = (char \*\*) q->col; // q->col arreglo de apuntadores a char \*

printf("[%s][%s]\n",\*cols,\*(cols+1));

q = q->next; // siguiente tupla del query

}

printf("Libero memoria de Query!\n");

fb\_free(&myquery);

} else { // error en ejecucion de query

printf("Error en ejecucion de Store Procedure!\n\

Error FB [%d] mensaje [%s] sql code [%ld]\n",myquery.fb\_error,myquery.errmsg,myquery.SQLCODE);

}

fb\_do\_disconnect(&dbinfo);

printf("Me desconecte!\n");

}

}