Diseño de Sistemas Transaccionales

Ayudantía 2

Gabriel Hourton B | ghourtonb@gmail.com | 09 - 299 1137

Consola PosgreSQL

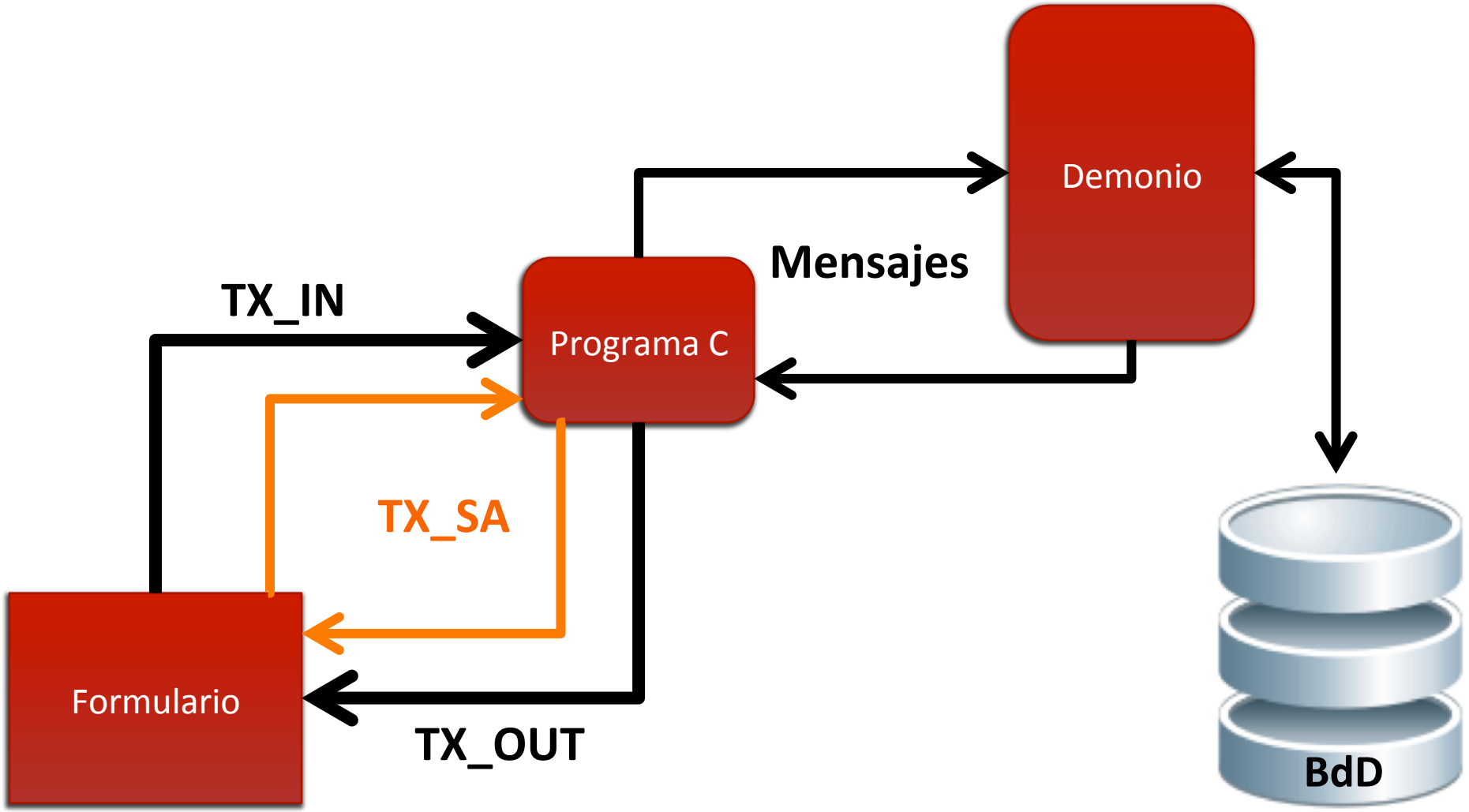
- **Ver Bases de Datos:**
  - psql -l
- **Abrir base de Datos:**
  - Psql <nombre_BdD> -U nombreusuario
- **Ver Tablas:**
  - \d
- **Salir:**
  - \q

http://www.postgresql.org/docs/9.0/interactive/index.html

Consola PosgreSQL

- Crear Base de Datos:
  - createdb <nombre_BdD> -U nombreusuario
- Eliminar Base de Datos:
  - dropdb <nombre_BdD> -U nombreusuario
- Eliminar tabla:
  - drop table <nombre_tabla> -U nombreusuario

Comunicación Formulario <-> Demonio



Demonio

¿Que es un Demonio?

- Proceso que se encuentra en ejecución por un periodo de tiempo prolongado ocupando muy poco recurso.
- En este caso es aquel que recibe el mensaje enviado por el Proceso Transaccional y envía la consulta a la base de datos.

Demonio

- **¿Por qué Archivo *.pgc?**
  - Código SQL “enfrascado” en C.
- **¿Como compilarlo?**
  - 1.- export LD_LIBRARY_PATH=/usr/pgsql-9.0/lib
  - 2.- ecpg demonio.pgc
  - 3.- gcc -I /usr/pgsql-9.0/include demonio.c -o demonio -L /usr/pgsql-9.0/lib –lecpg
- **¿Como ejecutarlo?**
  - ./demonio

Estructura Demonio

- **1.- Incluir Librerías necesarias.**
- **2.- Incluir mecanismo de manejo de errores SQL**
  - *EXEC SQL INCLUDE sqlca;*
- **3.- Declarar variables SQL**

```
EXEC SQL BEGIN DECLARE SECTION;

    VARCHAR SQL_dbname[9];
    VARCHAR SQL_user[7];
    VARCHAR SQL_password[20];
    int SQL_rut;
    VARCHAR SQL_nombre[31];
    int SQL_count;

    EXEC SQL END DECLARE SECTION;
```

Estructura Demonio

- **4.- Conectar con Base de Datos**

```
int SQLConectar() {

        strcpy(SQL_dbname.arr, "ayudantia");
        SQL_dbname.len = strlen(SQL_dbname.arr);
        strcpy(SQL_user.arr, "ghourton");
        SQL_user.len = strlen(SQL_user.arr);
        strcpy(SQL_password.arr, "");
        SQL_password.len = strlen(SQL_password.arr);

        EXEC SQL CONNECT TO :SQL_dbname USER :SQL_user IDENTIFIED BY :SQL_password

        if(sqlca.sqlcode != 0) {
                printf("Error en la conexion con la base de datos\n\n");
                return(0);
        } else {
                printf("Conexion con base de datos realizada\n\n");
                return(1);
        }
}// Fin SQLConectar()
```

Estructura Demonio

- **5.- Definir variables y estructuras para la comunicación con los .c**

```
int qid, pid , len;

struct msgbuf
{
        long mtype;
        struct
        {
                int pid;
                char datos[2000];
        } texto;
} mensaje, respuesta;

qid = msgget (5942016, IPC_CREAT|0666);
pid = getpid ();
```

Estructura Demonio

- **6.- Recibir mensaje**

```
if((len = msgrcv (qid, &mensaje, 2500, 1, 0)) > 0) {

        memset (&respuesta, 0, sizeof respuesta);
        printf("Recibido: (%d) <%d/%s>\n\n", mensaje.mtype, mensaje.texto.pid, mensaje.texto.datos);
        int pid_destino = mensaje.texto.pid;

        // Fomulario del que viene

        char formulario[7];
        memset (formulario, 0, sizeof formulario);
        sscanf(mensaje.texto.datos, "%6c", formulario);
        printf("El formulario es: [%s]\n\n", formulario);
```

- **7.- Identificar Proceso**

```
if (strcmp(formulario, "formej")==0){
```

Estructura Demonio

- **8.- Procesar información y Realizar consultas SQL correspondientes**

```
// Verificamos si el usuario ya esta registrado
EXEC SQL SELECT COUNT(*) INTO :SQL_count FROM EJEMPLO WHERE RUT=:SQL_rut;


EXEC SQL SELECT nombre INTO :SQL_nombre FROM EJEMPLO WHERE rut=:SQL_rut;
EXEC SQL COMMIT;

// Insertamos el alumno en la BdD
EXEC SQL INSERT INTO EJEMPLO (nombre, rut) VALUES (:SQL_nombre, :SQL_rut);
EXEC SQL COMMIT;
```

Estructura Demonio

- **9.- Enviar Respuesta**

```
memset( &respuesta, 0, sizeof respuesta);
sprintf(respuesta.texto.datos, "%s", "02");
respuesta.mtype = pid_destino;
respuesta.texto.pid = pid;
msgsnd(qid, &respuesta, strlen(respuesta.texto.datos)+4,0);
```