Práctica 1: PROCESO DE DESARROLLO EN XDR CON Sun RPC
(Será realizado en la Sala de Computo)

Diseñar un programa que permita registrar y consultar los pacientes de una clínica. Lo datos a registrar corresponden a: nombre del paciente, apellido del paciente, número de la habitación del paciente y edad del paciente. El sistema debe gestionar la información utilizando el modelo de RPC e implementándolo por medio de Sun RPC.

El administrador del programa debe contar con 2 operaciones: **registro de pacientes y consulta de pacientes**. Estas operaciones serán controladas mediante un Menú, tal como muestra la Figura:

========== Menú ==========

1. *Registrar paciente*
2. *Consultar paciente*
3. *Salir*

========== ==========

Digite opción:

Figura 1. Menú del administrador

La primera opción permite registrar todos los datos de un paciente. En el lado del servidor los pacientes serán almacenados en un vector. La máxima cantidad de pacientes a registrar será 5.

Desarrollo de la aplicación

La aplicación debe implementarse usando Sun RPC, mediante el lenguaje de programación C en un sistema operativo Linux. En la figura 2 se observa un diagrama de contexto que muestra las operaciones que puede realizar el cliente. Las operaciones debe ser implementadas en un servidor, cada vez que se realice una petición en el cliente y se reciba una petición en el servidor**, se debe crear un eco por medio de un mensaje en pantalla** en el cual se describe cual función se está pidiendo o ejecutando

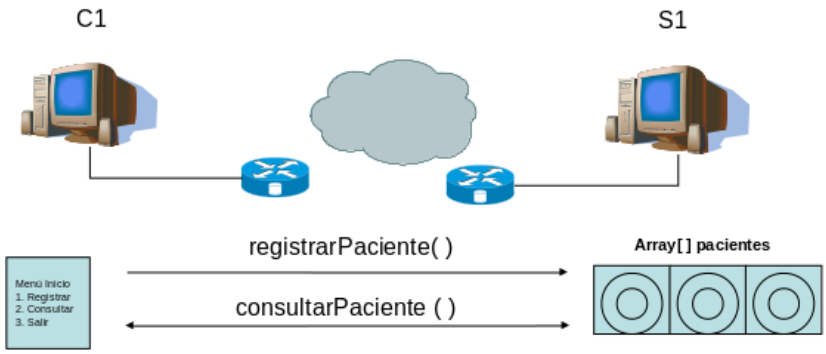

Figura 2. Sistema de registro.

Antes de iniciar se debe crear la siguiente estructura de directorios:

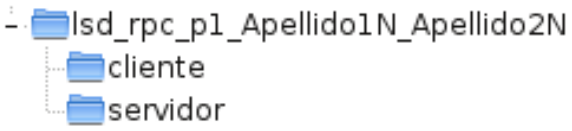

Figura 3. Estructura de directorios para desarrollar la práctica.

La solución de este ejercicio debe ser comprimida utilizando el comando:
tar cfvz directorioEmpaquetado.tar.gz directorioParaComprimir , este archivo debe ser subido al sitio del curso o ser enviado al correo del docente. El nombre del archivo comprimido debe seguir el siguiente formato lsd_rpc_p1_apellido1N_apellido2N.tar.gz. Donde apellido1 corresponde al primer apellido de uno de los integrantes, más la inicial del Primer Nombre y apellido2 corresponde al primer apellido del segundo integrante del grupo más la inicial del Primer Nombre. Dentro de la carpeta se debe establecer en un archivo los roles de cada uno de los integrantes.

1. Crear el IDL

En Sun RPC se definen las interfaces usando la notación XDR (External Data Representation) en un archivo con extensión **.x**. De acuerdo con el enunciado el cliente debe recurrir a un servicio que le permita realizar 2 operaciones especiales. Por lo tanto, el proceso Servidor deberá publicar mediante una definición de interface el procedimiento que atenderá la solicitud del proceso cliente. Para cumplir con los requerimientos del problema se recurre a la definición de 2 procedimientos, en un archivo denominado gestionPacientes.x, como se muestra en la figura 4.

```
/*Declaracion de datos a transferir entre el cliente y el servidor*/

/*Declaracion de constantes*/
const MAXNOM = 30;
const MAXTITULO = 60;


/*Declaracion de la estructura que permite almacenar los datos de un paciente*/
struct datos paciente{
    char nombre paciente[MAXNOM];
    char apellido paciente[MAXNOM];
    int numeroHabitacion;
    int edad;
};

/*Definicion de las operaciones que se pueden realizar*/
program gestion pacientes{
    version gestion pacientes version{
        bool registrarPaciente(datos paciente objPaciente)=1;
        datos paciente consultarPaciente(int numeroHabitacion)=2;
    }=1;
}=0x20000001;
```

Figura 4. Interface gestionPacientes.*x*

Mediante un editor de texto cree la definición de interface indicada en la Figura 4.

2. Procesamiento de la Interface

Ubicados en el directorio de trabajo, utilice el compilador de interfaces 'rpcgen' para procesar la definición de interfaces creada en el punto anterior. Desde una consola debe introducir el siguiente comando:

```
$ rpcgen gestionPacientes.x
```

Mediante el compilador de interfaces se generan siguientes archivos:

gestionPacientes_clnt.c: Contiene la funcionalidad del client stub.
gestionPacientes.h: En este archivo de cabecera se definen estructuras declaradas en la interface remota.
gestionPacientes_svc.c: Contiene la funcionalidad del server stub.
gestionPacientes_xdr.c: Contiene la rutinas de aplanamiento o codificación de los datos.

Los archivos generados se organizan en las carpetas de cliente y servidor de la siguiente forma:

Cliente:
gestionPacientes.x
gestionPacientes.h
gestionPacientes_clnt.c
gestionPacientes_xdr.c
Servidor:
gestionPacientes.x
gestionPacientes.h
gestionPacientes_svc.c
gestionPacientes_xdr.c

3. Crear el código del Cliente.

Sun RPC no posee un servicio binding de red global. En lugar de esto proporciona un servicio binding local llamado *portmapper*. El procesador de interfaces rpcgen permite generar templates (plantillas) del cliente y el servidor, las cuales crean las funcionalidades que logran que un cliente pueda comunicarse con un servidor utilizando el servicio de binding local *portmapper*. Para generar el código del cliente vamos a utilizar esta facilidad.

Pasos a seguir:

a. Ubicar el directorio cliente, mediante el comando:
cd cliente

b. Generar la plantilla del cliente, mediante el comando:
rpcgen –Sc gestionPacientes.x > Cliente.c

c. Ubicados en el subdirectorio 'cliente', cambiar los permisos para escribir en cliente.c, luego modificar el código generado (archivo cliente.c), introduciendo la lógica de control del programa, y los mensajes guía para orientar al usuario. En la figura 5 se observa un ejemplo de las variables de entrada y salida de los procedimientos, y en la figura 6, el llamado a los procedimientos que se encuentran dentro del archivo cliente.c.

```
CLIENT *clnt;
bool t  *result 1;
datos paciente  registrarpaciente 1 arg;
datos paciente  *result 2;
int  consultarpaciente 1 arg;
```

Figura 5. Variables de entrada y salida a los procedimientos en un archivo cliente.c

```
printf("Digite el nombre:");
scanf("%s",registrarpaciente 1 arg.nombre);
printf("Digite el apellido:");
scanf("%s",&registrarpaciente 1 arg.apellido);
.
.
printf("Digite el numero de habitación:");
scanf("%d",registrarpaciente 1 arg.numeroHabitacion);
result 1 = registrarpaciente 1(&registrarpaciente 1 arg, clnt);
if (result 1 == (bool t *) NULL) {
    clnt perror (clnt, "call failed");
}
else if(*result 1==TRUE){
    printf("\n Paciente registrado exitosamente");
}
else{
    printf("\n Paciente no registrado");
}
```

Figura 6. Ejemplo del llamado a un procedimiento en un archivo cliente.c

4. Crear el código del Servidor.

De acuerdo al requerimiento inicial es responsabilidad del programador del servicio implementar el código del servidor, para lo cual se utilizaran las plantillas de las Rutinas de Servicio.

Pasos a seguir:

a. Ubicar el directorio servidor, mediante el comando:
cd servidor

b. Generar la plantilla del servidor, mediante el comando:
rpcgen –Ss gestionPacientes.x > Servidor.c

c. Ubicados en el subdirectorio servidor, cambiar los permisos para escribir en servidor.c, luego modificar el código generado (archivo servidor.c), introduciendo la lógica de cada una de los procedimientos, un ejemplo de uno de los procedimientos se observan en la figura 7.

```
datos paciente vectorPacientes[5];
int posPaciente=0;

bool t *
registrarpaciente 1 svc(datos paciente *argp, struct svc req *rqstp)
{
    static bool t  result;

    printf("\n Invocando a registrarpaciente()\n");
    if (posPaciente<5)
    {
        vectorPacientes[posPaciente]=*argp;
        result =TRUE;
        posPaciente++;
        printf("\n Registrando paciente...\n");
    }
    else {
        printf("\n Cantidad maxima de registros alcanzado\n");
        result=FALSE;
    }
    return &result;
}
```

Figura 7. Procedimiento para registrar un paciente.

5. Compilar códigos

Pasos a seguir:

a. Estando en la carpeta **Servidor,** desde la consola generar el archivo **makefile** a través de la opción que posee la aplicación rpcgen para facilitar la compilación de los archivos fuente. Esto se logra mediante el siguiente comando:

```
rpcgen -Sm gestionPacientes.x > makeS
```

Cambiar los permisos para escribir en makeS.

Estando en la carpeta **Cliente,** desde la consola generar el archivo **makefile** a través de la opción que posee la aplicación rpcgen para facilitar la compilación de los archivos fuente. Esto se logra mediante el siguiente comando:

```
rpcgen -Sm gestionPacientes.x > makeC
```

Cambiar los permisos para escribir en makeC.

b. Luego de haber generado los anteriores archivos, se edita el **makefile** generado para el servidor y el cliente:

Editar y modificar los siguientes parámetros en el archivo **makeC**

CLIENT= cliente
SERVER= Borrar valor (Debe quedar en blanco)
SOURCES_CLNT.c = Cliente.c
SOURCES_CLNT.h = gestionPacientes.h

Editar y modificar los siguientes parámetros en el archivo **makeS**

CLIENT= Borrar valor (Debe quedar en blanco)
SERVER= Servidor
SOURCES_SVC.c = Servidor.c
SOURCES_SVC.h = gestionPacientes.h

En los archivos makefile asegúrese de que aparecen los archivos en los recuadros:

```
TARGETS_SVC.c =     gestionPacientes_xdr.c
TARGETS_CLNT.c = gestionPacientes_clnt.c   gestionPacientes_xdr.c
TARGETS = gestionPacientes.h gestionPacientes_xdr.c gestionPacientes_clnt.c
```

Si no aparecen debe colocarlos manualmente, lo anterior debido a que el makefile no se generó para el archivo gestionPacientes_xdr.c encargado de codificar y decodificar los datos.

c. Compilar los fuentes:

Para compilar los códigos fuentes del cliente desde la carpeta **Cliente:**

make -f makeC

Para compilar los códigos fuentes del servidor desde la carpeta **Servidor:**

make -f makeS

6. Ejecutar el cliente y el servidor

Cuando se compilan los archivos fuente se generan dos ejecutables, cuyo nombre depende de los parámetros fijados en la plantilla del archivo makefile.

a. Ubicados en la carpeta **Servidor**. El programa Servidor, se puede ejecutar localmente o en otra máquina.

   ./servidor

b. Ubicados desde la carpeta **Cliente**. El programa Cliente requiere como parámetro de entrada el nombre de la máquina (o dirección IP) donde está el Servidor.

   ./cliente localhost (o dirección IP)