Práctica 2: PROCESO DE DESARROLLO EN XDR CON Sun RPC
(Será realizado en la Sala de Computo)

Diseñar un programa que permita enviar los datos básicos de un paciente y el indicador de su frecuencia cardiaca a un servidor de alertas, el cual analizara si la frecuencia cardiaca se encuentra fuera de un rango normal, si es el caso, el servidor enviará una notificación de alerta médica a un servidor de notificaciones el cual mostrará el siguiente mensaje por pantalla:

> El paciente [Nombres] que se encuentra en la habitación [Número] presenta una frecuencia cardiaca de [Indicador de la frecuencia] la cual está fuera del rango normal

Los datos del paciente corresponden a: nombre del paciente, edad, número de la habitación del paciente y frecuencia cardiaca. El sistema debe gestionar la información utilizando el modelo de RPC e implementándolo por medio de Sun RPC. El rango normal de la frecuencia cardiaca por edad se muestra a continuación:

FRECUENCIA CARDIACA		
Grupo	**Edad**	**Latidos por minuto**
Adolescente	**13 – 16 años**	**70-80**
Adulto	**16 años y más**	**60-80**

El medico administrador del programa debe contar con 1 operación: **enviar registro del paciente**. Esta operación será controlada mediante un Menú, tal como muestra la Figura:

========== Menú ==========

1. *Enviar registro del paciente*
2. *Salir*

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

Digite opción:

Figura 1. Menú del medico

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:

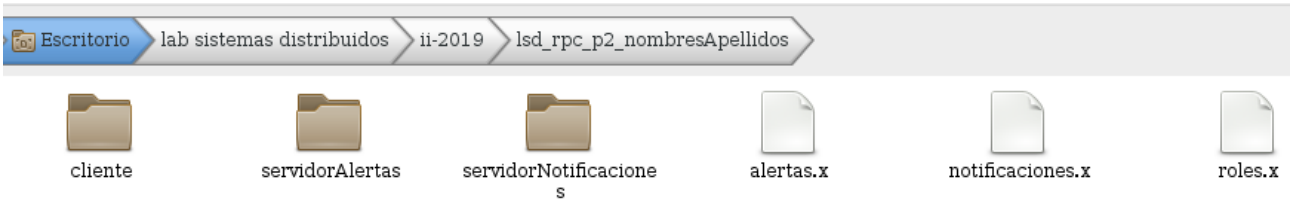

Figura 3. Estructura de directorios para desarrollar la práctica.

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_p2_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 es necesario crear 2 interfaces. La interface alertas.x permite definir el procedimiento para enviar los datos del paciente y la interface notificaciones.x permite enviar la notificación cuando la frecuencia cardiaca este fuera de rango normal.

Las inteface alertas.x se muestra en la figura 4, y la interface notificaciones.x se muestra en la figura 5.

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

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


/*Declaracion de la estructura que permite almacenar los datos de un anteproyecto*/
struct nodo_paciente{
        char nombres[MAXNOM];
        int edad;
        int numHabitacion;
        nodo_indicadores indicadores;
};

struct nodo_indicadores{
        float frecuenciaCardiaca;
};

/*Definicion de las operaciones que se pueden realizar*/
program gestion_alertas{
        version gestion_alertas_version{
                bool enviarIndicador(nodo_paciente objPaciente)=1;
        }=1;
}=0x20000001;
```

Figura 4. Interface alertas.*x*

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

/*Definicion de las operaciones que se pueden realizar*/
program gestion_notificaciones{
        version gestion_notificaciones_version{
                bool enviarNotificacion(string mensaje)=1;
        }=2;
}=0x20000002;
```

Figura 5. Interface notificaciones.*x*

Mediante un editor de texto cree la definición de interface indicada en la Figura 4 y la figura 5

2. Procesamiento de la Interface alertas

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 alertas.x
```

Mediante el compilador de interfaces se generan los siguientes archivos:

alertas _clnt.c: Contiene la funcionalidad del client stub.
alertas.h: En este archivo de cabecera se definen estructuras declaradas en la interface remota.
alertas _svc.c: Contiene la funcionalidad del server stub.
alertas _xdr.c: Contiene la rutinas de aplanamiento o codificación de los datos.

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

Cliente:
alertas.x
alertas.h
alertas _clnt.c
alertas _xdr.c

ServidorAlertas:
alertas.x
alertas.h
alertas _svc.c
alertas _xdr.c

3. Procesamiento de la Interface notificaciones

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 notificaciones.x
```

Mediante el compilador de interfaces se generan los siguientes archivos:

notificaciones _clnt.c: Contiene la funcionalidad del client stub.
notificaciones.h: En este archivo de cabecera se definen estructuras declaradas en la interface remota.
notificaciones _svc.c: Contiene la funcionalidad del server stub.

Los archivos generados se organizan en las carpetas de servidorAlertas y servidorNotificaciones de la siguiente forma:

servidorAlertas

alertas.x
alertas.h
alertas _svc.c
alertas _xdr.c
notificaciones.x
notificaciones.h
notificaciones _clnt.c

Los archivos deben agregarse a la carpeta servidorAlertas

servidorNotificaciones

notificaciones.x
notificaciones.h
notificaciones _svc.c

4. 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 alertas.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;
nodo_paciente  objPaciente;
```

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

```
printf("Digite el nombre del paciente: ");
scanf("%s",objPaciente.nombres);
printf("Digite la edad del paciente: ");
scanf("%d",&objPaciente.edad);
printf("Digite la frecuencia cardiaca del paciente: ");
scanf("%f",&objPaciente.indicadores.frecuenciaCardiaca);
result_1 = enviarindicador_1(&objPaciente, clnt);
if (result_1 == (bool_t *) NULL) {
        clnt_perror (clnt, "call failed");
}
```

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

5. Crear el código del Servidor de alertas

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 de alertas, mediante el comando:
cd servidorAlertas

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

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

d. Abrir el archivo Cliente.c y copiar el siguiente fragmento de código:

```
        CLIENT *clnt;
        bool_t  *result_1;
        char * enviarnotificacion_2_arg;

#ifndef DEBUG
        clnt = clnt_create (host, gestion_notificaciones, gestion_notificaciones_version, "udp");
        if (clnt == NULL) {
                clnt_pcreateerror (host);
                exit (1);
        }
#endif  /* DEBUG */

        result_1 = enviarnotificacion_2(&enviarnotificacion_2_arg, clnt);
        if (result_1 == (bool_t *) NULL) {
                clnt_perror (clnt, "call failed");
        }
#ifndef DEBUG
        clnt_destroy (clnt);
#endif   /* DEBUG */
```

e. Abrir el archivo Servidor.c y pegar el fragmento de código de la siguiente manera:

```
#include "alertas.h"

bool_t *
enviarindicador_1_svc(nodo_paciente *argp, struct svc_req *rqstp)
{
        static bool_t  result;

        CLIENT *clnt;
        bool_t  *result_1;
        char * enviarnotificacion_2_arg;

        #ifndef DEBUG
        clnt = clnt_create (host, gestion_notificaciones, gestion_notificaciones_version, "udp");
        if (clnt == NULL) {
                clnt_pcreateerror (host);
                exit (1);
        }
        #endif  /* DEBUG */

        result_1 = enviarnotificacion_2(&enviarnotificacion_2_arg, clnt);
        if (result_1 == (bool_t *) NULL) {
                clnt_perror (clnt, "call failed");
        }
        #ifndef DEBUG
        clnt_destroy (clnt);
        #endif   /* DEBUG */

        return &result;
}
```

El fragmento permite que el servidor de alertas consulte al portmapper por el puerto del servidor de notificaciones e invoque un procedimiento remoto del servidor de notificaciones.

Modificar el código pegado, incluyendo la librería notificaciones.h y agregando una variable donde se almacena la dirección ip del servidor de notificaciones, de la siguiente manera:

```
#include "alertas.h"
#include "notificaciones.h"
bool_t *
enviarindicador_1_svc(nodo_paciente *argp, struct svc_req *rqstp)
{
        static bool_t  result;
        CLIENT *clnt;
        bool_t  *result_1;
        char * mensajeAEnviar="notificacion: la frecuencia cardiaca esta fuera del rango normal";
        printf("\n Datos del paciente nombres: %s,frecuencia cardiaca %f \n", (*argp).nombres, (*argp).indicadores.frecuenciaCardiaca);
        char *dirIpServidorNotificaciones="localhost";

        #ifndef DEBUG
        clnt = clnt_create (dirIpServidorNotificaciones, gestion_notificaciones, gestion_notificaciones_version, "udp");
        if (clnt == NULL) {
                clnt_pcreateerror (dirIpServidorNotificaciones);
                exit (1);
        }
        #endif  /* DEBUG */

        result_1 = enviarnotificacion_2(&mensajeAEnviar, clnt);
        if (result_1 == (bool_t *) NULL) {
                clnt_perror (clnt, "call failed");
        }
        #ifndef DEBUG
        clnt_destroy (clnt);
        #endif   /* DEBUG */
        return &result;
}
```


f. Eliminar el archivo Cliente.c mediante el comando:
rm –r Cliente.c

6. Crear el código del Servidor de notificaciones

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 de notificaciones, mediante el comando:
cd servidorNotificaciones

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

Un ejemplo de la implementación del procedimiento remoto del servidor de notificaciones es el siguiente:

```
#include "notificaciones.h"

bool_t *
enviarnotificacion_2_svc(char **argp, struct svc_req *rqstp)
{
        static bool_t  result;

        printf("\n Mensaje desde el servidor de alertas %s \n",*argp);

        return &result;
}
```

5. Compilar códigos

Pasos a seguir:

a. Estando en la carpeta **ServidorAlertas,** desde la consola generar el archivo **makefile** mediante el siguiente comando:

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

Cambiar los permisos para escribir en makeS.

b. Estando en la carpeta **ServidorNotificaciones,** desde la consola generar el archivo **makefile** mediante el siguiente comando:

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

Cambiar los permisos para escribir en makeS.

c. Estando en la carpeta **Cliente,** desde la consola generar el archivo **makefile** mediante el siguiente comando:

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

Cambiar los permisos para escribir en makeC.

d. Editar el **makefile** generado para 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 = alertas.h

e. Editar el makefile generado para el servidor de alertas

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 = alertas.h notificaciones.h

TARGETS_SVC.c = alertas_svc.c alertas_xdr.c notificaciones_clnt.c

f. Editar el makefile generado para el servidor de notificaciones

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 = notificaciones.h

g. Compilar los fuentes:

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

make -f makeC

Para compilar los códigos fuentes del servidor de alertas desde la carpeta **ServidorAlertas:**

make -f makeS

Para compilar los códigos fuentes del servidor de notificaciones desde la carpeta **ServidorNotificaciones:**

make -f makeS

6. Ejecutar el cliente y el servidor

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

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

   ./servidor

b. Ubicados en la carpeta **ServidorAlertas**. El programa Servidor, se puede ejecutar localmente o en otra máquina. El programa servidor requiere como parámetro la dirección ip donde está el

Servidor de notificaciones, para este ejercicio la dirección ip estará fijada en el código del servidor de alertas.

./servidor

c. Ubicados desde la carpeta **Cliente**. El programa Cliente requiere como parámetro la dirección ip donde está el Servidor de alertas.

./cliente localhost (o dirección IP)