Memoria de P3

Filtros de conversión XDR (Parte I)

Shunya Zhan

22 de abril del 2025

ÍNDICE

Introducción

El objetivo de este proyecto es implementar un sistema cliente-servidor utilizando Remote Procedure Call (RPC) para listar los archivos de un directorio en el servidor desde un cliente remoto. Este sistema permite explorar directorios de manera remota, utilizando una arquitectura distribuida basada en RPC.

En el contexto de la informática distribuida, RPC es una tecnología que permite a un programa ejecutar procedimientos en otro programa que se encuentra en una máquina diferente, como si fueran locales. Esto facilita la creación de aplicaciones que pueden aprovechar los recursos de múltiples sistemas, mejorando la eficiencia y la capacidad de procesamiento.

El sistema desarrollado en este proyecto consta de dos componentes principales: el servidor RPC y el cliente RPC. El servidor RPC es responsable de recibir las solicitudes del cliente, abrir el directorio especificado, y devolver una lista enlazada con los nombres de los archivos. Por otro lado, el cliente RPC envía la solicitud al servidor y muestra los nombres de los archivos recibidos.

Este proyecto no solo demuestra la implementación de un sistema cliente-servidor utilizando RPC, sino que también destaca la modularidad y escalabilidad del código. La utilización de listas enlazadas para manejar los nombres de los archivos permite una gestión dinámica y eficiente de los datos. Además, este enfoque puede ser extendido para implementar funcionalidades más complejas, como la manipulación remota de archivos o la gestión de permisos.

Desarrollo

El archivo credenciales.x define el programa RPC, los tipos de datos y las funciones disponibles.

Se utiliza rpcgen para generar los archivos necesarios (credenciales.h, credenciales_clnt.c, credenciales_svc.c, credenciales_xdr.c).

El sistema está compuesto por dos componentes principales:

1. Servidor RPC: Este componente recibe las solicitudes del cliente, abre el directorio especificado y devuelve una lista enlazada con los nombres de los archivos. La lógica del servidor se implementa en el archivo directorios_server.c.
2. Cliente RPC: Este componente envía la solicitud al servidor y muestra los nombres de los archivos recibidos. La lógica del cliente se implementa en el archivo directorios_client.c.

Archivos principales:

1. directorios.x: Define las estructuras de datos y las funciones RPC. Este archivo es fundamental para la generación de los archivos de código necesarios para la comunicación RPC.
2. directorios_server.c: Implementa la lógica del servidor, incluyendo la apertura del directorio y la construcción de la lista enlazada con los nombres de los archivos.
3. directorios_client.c: Implementa la lógica del cliente, incluyendo el envío de la solicitud al servidor y la impresión de los nombres de los archivos recibidos.
4. directorios.h: Archivo de cabecera generado por rpcgen, que contiene las definiciones necesarias para la comunicación RPC.
5. directorios_clnt.c, directorios_svc.c, directorios_xdr.c: Archivos generados automáticamente por rpcgen, que contienen el código necesario para la serialización y deserialización de datos, así como las funciones de cliente y servidor.

Flujo del programa

1. Solicitud del cliente: El cliente envía una solicitud al servidor con la ruta del directorio que desea explorar.
2. Procesamiento del servidor: El servidor recibe la solicitud, abre el directorio especificado y construye una lista enlazada con los nombres de los archivos.
3. Respuesta del servidor: El servidor envía la lista de nombres de archivos al cliente.
4. Visualización del cliente: El cliente recibe la lista y muestra los nombres de los archivos en la consola.

Memoria

Estructuras de datos

- namenode: Nodo de la lista enlazada que contiene el nombre de un archivo y un puntero al siguiente nodo.
- namelist: Puntero al primer nodo de la lista enlazada.

Funciones principales

Servidor:

- get_directorios_1_svc: Implementa la lógica para abrir el directorio, recorrerlo, y construir la lista enlazada.

Cliente:

- get_directorios_1: Realiza la llamada RPC al servidor.
- imprimir_nombres: Recorre la lista enlazada y muestra los nombres de los archivos.

Compilación y ejecución

Compilación: Abre una terminal, levantar el docker cliente y servidor y después navega al directorio donde se encuentran los archivos fuente. Luego, ejecuta el siguiente comando para compilar:

- gcc -o servidor -I /usr/include/tirpc directorios_server.c directorios_svc.c directorios_xdr.c -l tirpc
- gcc -o cliente -I /usr/include/tirpc directorios_client.c directorios_clnt.c directorios_xdr.c -l tirpc

Ejecución: En la terminal, ejecuta el servidor con el siguiente comando:

*NOTA: antes de poner el nombre del directorio, hay que asegurar que había creado el directorio al mismo nivel, y que contenga fichero dentro del directorio

- ./cliente <IP_del_servidor> <nombre_del_directorio>
- ./servidor

Prueba y resultado

Servidor:

```
rpcuserserver@15e5bddb0ad8:~/p5$ gcc -o servidor -I /usr/include/tirpc credenciales_server.c credenciales_svc.
c credenciales_xdr.c -l tirpc
```

```
rpcuserserver@15e5bddb0ad8:~/p3$ ls
Makefile.directorios  directorios.h  directorios_client.c  directorios_server.c  directorios_xdr.c
directorios           directorios.x  directorios_clnt.c    directorios_svc.c     servidor
rpcuserserver@15e5bddb0ad8:~/p3$ ./servidor
OK: Directorio ** directorios ** abierto
Directorio actual: prueba
Directorio actual: .
Directorio actual: prueba2
Directorio actual: ..
```

Cliente:

```
rpcuserclient@937a38ebb92c:~/p3$ gcc -o cliente -I /usr/include/tirpc directorios_client.c directorios_clnt.c directorio
s_xdr.c -l tirpc
```

```
rpcuserclient@937a38ebb92c:~/p3$ ./cliente 172.17.0.2 directorios
prueba
prueba2
```

Conclusión

Este proyecto demuestra cómo implementar un sistema cliente-servidor utilizando Remote Procedure Call (RPC) para realizar operaciones remotas, en este caso, listar los archivos de un directorio en el servidor desde un cliente remoto. La arquitectura distribuida basada en RPC permite explorar directorios de manera eficiente y segura, aprovechando las ventajas de la informática distribuida.

La modularidad del código facilita su mantenimiento y escalabilidad. Cada componente del sistema, tanto el servidor como el cliente, está claramente definido y separado, lo que permite realizar modificaciones y mejoras de manera sencilla. Además, el uso de listas enlazadas para manejar los nombres de los archivos proporciona una gestión dinámica y eficiente de los datos, adaptándose a diferentes tamaños y cantidades de archivos sin problemas.

Este enfoque puede extenderse para implementar funcionalidades más complejas, como la manipulación remota de archivos, la gestión de permisos, o incluso la integración con otros sistemas distribuidos. La flexibilidad y la robustez del sistema desarrollado en este proyecto ofrecen una base sólida para futuras expansiones y aplicaciones en el ámbito de la informática distribuida.

En resumen, este proyecto no solo cumple con el objetivo de listar archivos de manera remota, sino que también proporciona una comprensión profunda de los conceptos de RPC y su aplicación en sistemas distribuidos. La implementación exitosa de este sistema cliente-servidor demuestra la viabilidad y las ventajas de utilizar RPC para operaciones remotas, abriendo la puerta a nuevas posibilidades y desarrollos en el campo de la informática.

Información extra

Ejecuta rpcgen ¿Qué ficheros se generan?

Al ejecutar rpcgen con el archivo directorios.x, se generan varios ficheros:

- directorios.h: Archivo de cabecera con las definiciones necesarias para la comunicación RPC.
- directorios_clnt.c: Código fuente del cliente RPC.
- directorios_svc.c: Código fuente del servidor RPC.
- directorios_xdr.c: Código fuente para la serialización y deserialización de datos (XDR).

¿Qué hay en el *.h y en el fichero *_xdr.c?

- directorios.h: Contiene las definiciones de las estructuras de datos y las funciones RPC. Define los tipos de datos utilizados en la comunicación entre el cliente y el servidor.
- directorios_xdr.c: Contiene las funciones para la serialización (codificación) y deserialización (decodificación) de los datos. Estas funciones permiten convertir las estructuras de datos a un formato que puede ser transmitido a través de la red.

Explica con tus palabras los tipos que se han creado.

En el archivo directorios.x se define los datos necesarios para la comunicación RPC

- nametype: Representa un nombre de archivo como una cadena de caracteres con una longitud máxima definida.
- namenode: Representa un nodo en una lista enlazada que contiene el nombre de un archivo y un puntero al siguiente nodo.
- namelist: Es un puntero al primer nodo de la lista enlazada de nombres de archivos.

La constante definida ¿dónde se la declara con un #define?

Las constantes se declaran en el archivo de cabecera (directorios.h) utilizando la directiva #define.

¿Qué estructura de datos se utiliza para almacenar los nombres de archivos en el servidor?

En el servidor, se utiliza una lista enlazada para almacenar los nombres de los archivos. Cada nodo de la lista es una estructura namenode que contiene el nombre del archivo y un puntero al siguiente nodo.

¿Cómo se maneja la memoria para evitar fugas en la lista enlazada?

Para evitar fugas de memoria, es importante liberar la memoria asignada a cada nodo de la lista enlazada una vez que ya no se necesita. Esto se puede hacer recorriendo la lista y liberando cada nodo individualmente:

```
void liberar_lista(namelist *lista) {
        struct namenode *actual = *lista;
        struct namenode *siguiente;
        while (actual != NULL) {
        siguiente = actual->next;
        free(actual->name);
        free(actual);
        actual = siguiente;
        }
}
```

Explica el propósito de xdr_free(). ¿Sirve para toda la estructura o solo para un nodo?

El propósito de xdr_free() es liberar la memoria asignada a las estructuras de datos serializadas. Esta función puede liberar toda la estructura, no solo un nodo. Se utiliza para evitar fugas de memoria al deserializar datos recibidos a través de RPC.

¿Cómo se registra el servicio en el servidor y cómo se comunica el cliente con él?

Registro del servicio en el servidor: El servicio se registra en el servidor utilizando la función svc_register(). Esta función asocia el programa RPC con un puerto específico y lo hace disponible para recibir solicitudes.

Comunicación del cliente con el servidor: El cliente se comunica con el servidor utilizando las funciones generadas por rpcgen. Estas funciones realizan la llamada RPC y manejan la comunicación de manera transparente.

Modifica el código del servidor para que ignore los archivos ocultos (aquellos cuyo nombre empieza con un punto .).

Para modificar el código del servidor y que ignore los archivos ocultos, puedes agregar una condición en la función get_directorios_1_svc para omitir los archivos cuyo nombre empieza con un punto:

Anexo

- Código fuente
- Campus Virtual
- Copilot