Documentación de Diseño del Cliente y Servidor TFTP

Introducción

Este documento detalla el diseño e implementación de un cliente y servidor TFTP (Trivial File Transfer Protocol). Tanto el servidor como el cliente están implementados en lenguaje C. El protocolo TFTP es un protocolo simple utilizado para la transferencia de archivos, basado en el protocolo UDP (User Datagram Protocol).

Descripción General

El servidor TFTP escucha en un puerto específico, que está implementado como constante y como variable para ingresar en la consola, y procesa solicitudes de lectura (RRQ) y escritura (WRQ) de archivos. El cliente envía solicitudes al servidor para leer o escribir archivos. En ambos casos se guardan en carpetas específicas, los archivos que lee el cliente se guardan en la carpeta "client_files", mientras que que los archivos que lee el servidor se guardan en la carpeta "server_files".

Diseño del Servidor

Función principal (`main`)

1. **Creación del Socket:**
    - El servidor crea un socket UDP utilizando `socket(AF_INET, SOCK_DGRAM, 0)`.
2. **Asignación del Puerto:**
    - Utiliza `setsockopt` para permitir la reutilización de direcciones.
    - El servidor se enlaza (`bind`) a la dirección IP y al puerto especificado.
3. **Bucle Principal:**
    - El servidor entra en un bucle infinito donde espera recibir datos con `recvfrom`.
    - Según el opcode recibido, el servidor llama a `handle_rrq` para solicitudes de lectura y `handle_wrq` para solicitudes de escritura.

Funciones de Manejo de Solicitudes

1. **handle_rrq:**
    - Abre el archivo solicitado en modo lectura.
    - Envía el archivo en bloques de 512 bytes hasta completar la transferencia o encontrar un error.

    - Espera y verifica la recepción de ACKs por cada bloque enviado.
2. **handle_wrq:**
    - Crea un archivo para escritura.
    - Envía un ACK inicial.
    - Recibe bloques de datos del cliente, escribiendo en el archivo y enviando un ACK por cada bloque.

Funciones de Envío y Recepción

1. **send_error:**
    - Envía un paquete de error al cliente, indicando el código y mensaje de error.
2. **send_ack:**
    - Envía un ACK por un bloque recibido correctamente.
3. **receive_ack:**
    - Recibe un ACK y verifica que corresponde al bloque esperado.

Diseño del Cliente

Función principal (`main`)

1. **Creación del Socket:**
    - El cliente crea un socket UDP con `socket(AF_INET, SOCK_DGRAM, 0)`.
2. **Operaciones de Transferencia:**
    - El cliente envía una solicitud RRQ o WRQ al servidor según los argumentos de la línea de comandos.
    - Maneja la recepción de datos para RRQ o el envío de datos para WRQ.

Funciones de Envío y Recepción

1. **send_rrq:**
    - Envía una solicitud de lectura de archivo al servidor.
2. **send_wrq:**
    - Envía una solicitud de escritura de archivo al servidor.
    - Envía el archivo en bloques de 512 bytes.
    - Espera y verifica la recepción de ACKs por cada bloque enviado.
3. **handle_data:**
    - Recibe bloques de datos del servidor y escribe en el archivo local.
    - Envía ACKs por cada bloque recibido.
4. **send_error:**
    - Envía un paquete de error al servidor.
5. **send_ack:**
    - Envía un ACK por un bloque recibido correctamente.
6. **receive_ack:**
    - Recibe un ACK y verifica que corresponde al bloque esperado.

Consideraciones de Diseño

1. **Protocolos de Red:**
    - El uso de UDP permite la simplicidad y rapidez en la transferencia de datos, pero requiere manejo explícito de errores y confirmaciones (ACKs).
2. **Estructura de Paquetes:**
    - Los paquetes de datos y control siguen la estructura del protocolo TFTP, con opcodes para diferenciar los tipos de mensajes (RRQ, WRQ, DATA, ACK, ERROR).
3. **Manejo de Errores:**
    - El servidor y cliente están diseñados para enviar y recibir mensajes de error para manejar condiciones excepcionales como archivos no encontrados o errores de protocolo.
4. **Modularidad:**
    - La implementación está dividida en funciones modulares para manejar distintas partes del protocolo, facilitando la comprensión y mantenimiento del código.

Consideraciones en el Diseño del Ingreso de Comandos

Para ingresar la acción deseada, cada cliente debe seguir un orden específico al introducir comandos en la consola. Al ejecutar el programa, el usuario debe proporcionar la IP del servidor, el puerto en el que el servidor escucha, el comando que desea ejecutar (ya sea RRQ o WRQ), y el nombre del archivo que debe estar en una de las carpetas previamente mencionadas (“client_files” y “server_files”).
El formato completo del comando es:

`bin/client-tftp <IP> <PUERTO> <RRQ/WRQ> <ARCHIVO>`

Si el comando no está correctamente estructurado, el cliente mostrará un mensaje de error al usuario, especificando el orden correcto de los parámetros.

Resumen

En este proyecto se implementó un cliente y servidor TFTP (Trivial File Transfer Protocol), utilizando el lenguaje C para el servidor y otro lenguaje a elección para el cliente. El servidor TFTP escucha en un puerto específico y procesa solicitudes de lectura (RRQ) y escritura (WRQ) de archivos, almacenando los archivos en la carpeta “server_files”. El cliente envía solicitudes para leer o escribir archivos, y guarda los archivos leídos en la carpeta “client_files”.