**Server Documentation**

**This documentation provides an overview and explanation of the server code implementation.**

Table of Contents:

Introduction

Server Initialization

Handling Client Connections

Handling Client Requests

Shutting Down the Server

1. Introduction

The server code is responsible for setting up a server socket, accepting client connections, and handling client requests. It supports three main operations: list files, upload files, and download files. The code utilizes multi-threading to handle multiple clients concurrently.

The server code consists of the following functions:

handle\_interrupt\_signal: Handles the interrupt signal to shut down the server safely.

start\_server: Initializes the server socket, binds it to a specific port, and starts listening for incoming client connections.

handle\_client: Handles the communication with a client, authenticates the client, and processes the client's request.

Additional helper functions: authenticate\_client, end\_dialog, send\_list, receive\_file, send\_file, save\_users\_log, and save\_errors.

2. Server Initialization

handle\_interrupt\_signal(int signum)

This function is called when the server receives an interrupt signal (e.g., Ctrl+C). It is responsible for shutting down the server safely by closing the server socket and exiting the program.

start\_server()

This function is the entry point for starting the server. It performs the following steps:

Creates a server socket using the socket system call.

Sets up the server address using a sockaddr\_in structure.

Binds the server socket to the specified port.

Starts listening for incoming connections on the server socket.

3. Handling Client Connections

handle\_client(void\* ptr\_client\_socket)

This function is executed in a separate thread for each client connection. It handles the communication with the client and processes the client's request. The steps involved are as follows:

Extracts the client socket from the input parameter.

Receives the client's username, password, and user option (LIST, UPLOAD, or DOWNLOAD) from the client.

Authenticates the client by verifying the username and password using the authenticate\_client function.

Based on the user option, performs the following actions:

For LIST option: Sends the list of files to the client using the send\_list function.

For UPLOAD option: Requests the client to send a file name, checks if the file already exists, and if not, receives the file from the client using the receive\_file function.

For DOWNLOAD option: Sends the requested file to the client using the send\_file function.

Closes the client socket and exits the thread.

4. Handling Client Requests

authenticate\_client(char\* username, char\* password)

This function authenticates the client by checking if the provided username and password match the credentials stored on the server. The authentication mechanism may vary and is not implemented in the given code snippet.

send\_list(int client\_socket)

This function retrieves the list of files available on the server and sends it to the client over the client socket.

receive\_file(char\* file\_name, int client\_socket)

This function receives a file from the client and saves it on the server under the specified file name.

send\_file(char\* file\_name, int client\_socket)

This function sends a requested file to the client over the client socket.

5. Shutting Down the Server

When the server receives an interrupt signal (e.g., Ctrl+C), the handle\_interrupt\_signal function is triggered. It performs the following steps to shut down the server safely:

Prints a message indicating that the server is shutting down.

Closes the server socket.

Exits the program using the provided signal number.

It is essential to handle the interrupt signal to ensure proper cleanup and graceful termination of the server.

Note: The code snippet provided is missing the implementations of some helper functions (end\_dialog, save\_users\_log, save\_errors) and the necessary includes. These functions are assumed to be implemented elsewhere, and their functionalities are not described in this documentation.