CSE344 System Programming

2023 Spring

Final Project

Bahadir Etka Kilinc

1901042701

Introduction

The BiBakDropBox is a client-server file synchronization program implemented in C. It enables clients to synchronize their files with a remote server, ensuring that both the client and server have consistent and up-to-date copies of the files. The program establishes a socket connection between the client and server and facilitates the exchange of file information and contents.

Server Side

Header Files and Definitions:

The code starts by including various header files required for socket programming, file operations, threading, and synchronization.
It defines constants such as BUFFER_SIZE (the size of the buffer used for communication), PERM (file permission), logfile (the name of the log file), and server_active (the name of the file indicating an active server).

Global Variables:

Several global variables are declared, including semaphores (empty, full, m, sem_print), buffer size (buf_size), indices for buffer manipulation (push, pop), an array to hold the buffer elements (thread_buf), thread-related variables (tids, thread_count, active_thread_count), and socket-related variables (s, socket_addr).
The active_directory variable stores the path to the file indicating an active server.

Main Function:

The main function is the entry point of the program.
It checks the command-line arguments to ensure the correct number of arguments is provided.
It creates a file (server_active) in the specified directory to indicate an active server. If the file already exists, it terminates the program.
It initializes the socket, binds it to a port, and starts listening for incoming connections.
Semaphore initialization and buffer allocation are performed.
It creates a number of threads specified by threadPoolSize and assigns them the server function as their entry point.
The program enters an infinite loop to accept client connections.

If the number of active threads reaches the maximum allowed (threadPoolSize), the client is informed and the loop continues.
When a client connects, the client's IP address is displayed.
The program waits for an empty slot in the buffer (empty semaphore) and then acquires the mutex (m semaphore) to update the buffer.
The client's connection socket file descriptor and the active directory path are stored in the buffer, and the necessary semaphores are released.
After the loop, the program cleans up and releases resources.

Server Function:

The server function is the entry point for each thread.
Inside an infinite loop, the thread waits for a connection from a client (full semaphore).
The thread acquires the mutex (m semaphore) to read the client's connection details from the buffer and updates the active thread count.
The log file for the client is opened in append mode.
The server performs file transfer operations by calling server_to_client and client_to_server functions.
After file transfer is completed or an error occurs, the client's connection is closed, the empty slot in the buffer is released (empty semaphore), and the active thread count is updated.
The log file is closed.
The thread repeats this process to handle the next client connection.

Helper Functions:

The code includes several helper functions such as server_to_client, client_to_server, sendfile, recvfile, intlenght, and handler.
server_to_client function is responsible for sending files from the server to the client. It recursively traverses the server directory, checks for file existence on the client, and transfers files accordingly. If a file is not present on the client, it is either deleted from the server or transferred to the client.
client_to_server function is responsible for receiving files from the client to the server. It receives file information from the client, checks if the file needs to be created or updated on the server, and performs the corresponding actions.
sendfile and recvfile functions are wrappers around send and recv functions respectively, with additional error handling and logging.
intlenght function calculates the number of digits in an integer, which is useful for formatting log messages.

handler function is the signal handler for the SIGINT signal. It gracefully terminates the server by cleaning up resources and deleting the active server file.

This code creates a multi-threaded server that allows clients to transfer files to and from the server. The server handles multiple connections concurrently using a thread pool. It ensures synchronization and mutual exclusion using semaphores and mutexes. The server logs the details of each file transfer in a log file for analysis and monitoring.

Client Side

callSocket Function:

This function establishes a socket connection with the server using the provided IP address and port number.
It creates a socket and connects it to the server using the specified IP address and port.
If the connection is successful, it returns the socket file descriptor; otherwise, it returns -1.

recvfile Function:

This function is responsible for receiving file contents from the server and writing them to a local file.
It reads data from the socket into a buffer with a specified buffer size.
The function continues reading from the socket until it receives a special delimiter ("-1"), indicating the end of the file transfer.
During the file transfer, the received data is written to the specified file descriptor.
Once the file transfer is complete, the function closes the file descriptor.

sendfile Function:

This function is responsible for reading file contents from a local file and sending them to the server.
It reads data from the local file into a buffer with a specified buffer size.
The function sends the data over the socket to the server.
If the size of the data read is less than the buffer size, it indicates the end of the file transfer.
The function sends a special delimiter ("-1") to indicate the completion of the file transfer.
Once the file transfer is complete, the function closes the local file descriptor.

intlenght Function:

This function calculates the length of an integer value.
It takes an integer as input and recursively determines the number of digits in the integer.
The function returns the length of the integer.

handler Function:

This function serves as the signal handler for the program.
It is triggered when the program receives a SIGINT signal (e.g., pressing Ctrl+C).
The function closes the socket connection and gracefully terminates the program.

pathname Function:

This function parses the given file path to extract the file name.
It takes a file path as input and recursively finds the file name by searching for the last occurrence of the "/" character.
The function returns the extracted file name.

server_to_client Function:

This function handles the data flow from the server to the client during file synchronization.
It receives file information and handles file creation or deletion on the client-side based on server responses.
The function receives file information from the server, including the file name and modification timestamp.
If the received size is -2, it indicates that the file is a directory.
If the directory does not exist on the client, the function creates the directory and continues the synchronization process recursively.
If the directory already exists, the function continues the synchronization process for the existing directory.
If the received size is -1, it indicates the end of the file synchronization process.
If the file does not exist on the client or the modification timestamp is newer than the client's version, the function sends a request to the server to send the file.
If the file exists on the client but is not present on the server, it is deleted from the client's directory.

client_to_server Function:

This function handles the data flow from the client to the server during file synchronization.
It sends file information and contents from the client to the server for synchronization.
The function opens the specified file and retrieves its modification timestamp.

It sends the file information (file name, modification timestamp, and file size) to the server.
If the file size is not zero, it sends the file contents to the server using the sendfile function.

client Function:

This function coordinates the file synchronization process between the client and server.
It receives the server's socket file descriptor as input.
The function initiates the file synchronization by browsing the client's directory and sending file information and contents to the server using the client_to_server function.
It then receives file information and handles file creation or deletion on the client-side based on server responses using the server_to_client function.
main Function:

This is the entry point of the program.
It takes command-line arguments for the directory name, server IP address, and port number.
The function establishes the socket connection with the server using the callSocket function.
It registers the signal handler using signal(SIGINT, handler) to handle program termination.
The function calls the client function to initiate the file synchronization process.

Conclusion

The BibakDropBox acomprehensive client-server file synchronization program implemented in C. It ensures the synchronization of files between a client and a remote server. The program establishes a socket connection, facilitates the exchange of file information and contents, and handles file creation and deletion based on server responses. It provides robust mechanisms for keeping files consistent and up-to-date on both the client and server sides.

Test Cases

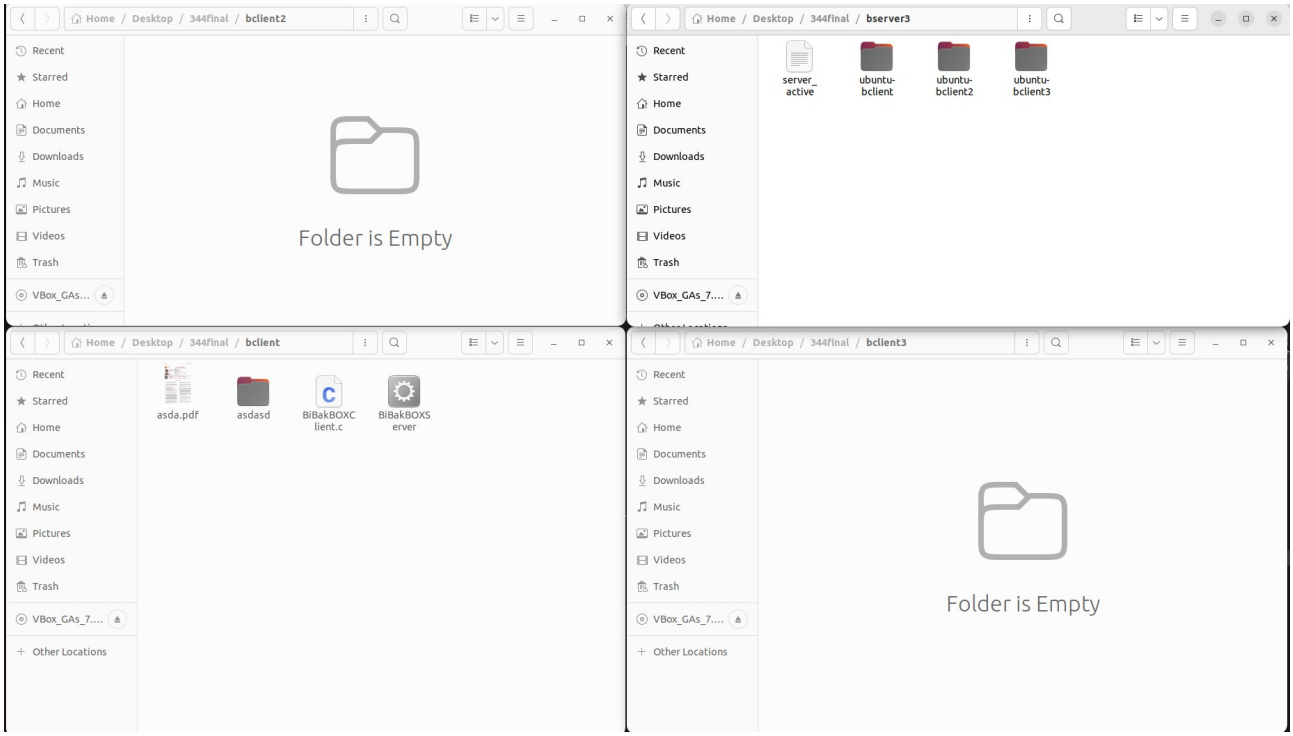
```
Failed to open directory
vboxuser@ubuntu:~/Desktop/344final$ ./BiBakBOXServer bserver3 10 1035
IP address: 127.0.0.1
IP address: 127.0.0.1
Read Failed : Success
Client disconnect
IP address: 127.0.0.1
IP address: 127.0.0.1
```

```
Server closed
vboxuser@ubuntu:~/Desktop/344final$ ./BiBakBOXClient bclient 127.0.0.1 1035
```

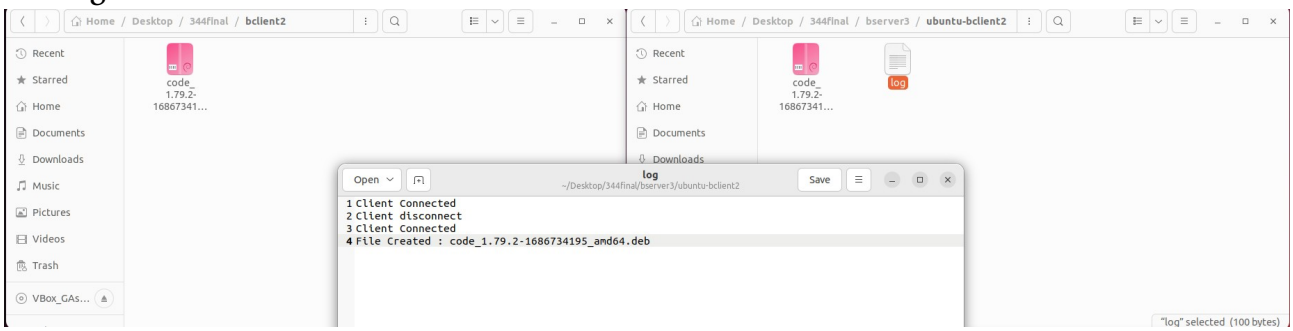
```
vboxuser@ubuntu:~/Desktop/344final$ ./BiBakBOXClient bclient2 127.0.0.1 1035
```

```
vboxuser@ubuntu:~/Desktop/344final$ ./BiBakBOXClient bclient3 127.0.0.1 1035
```

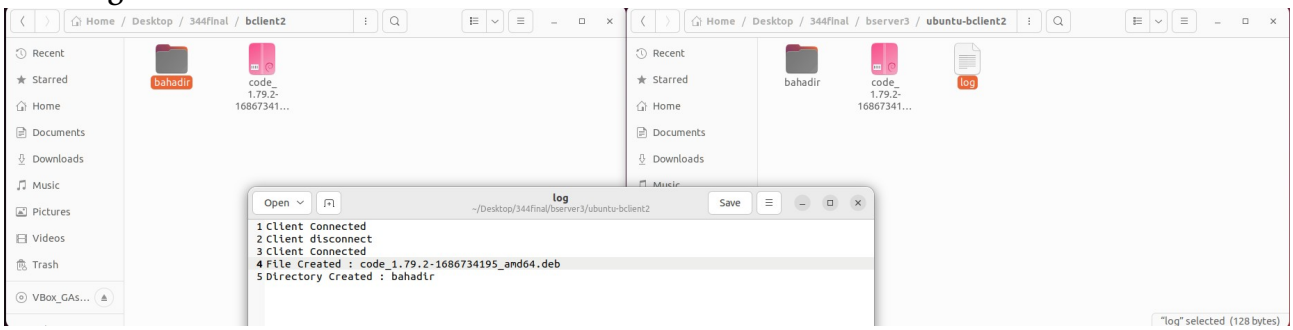
Multiple Clients connected the server.



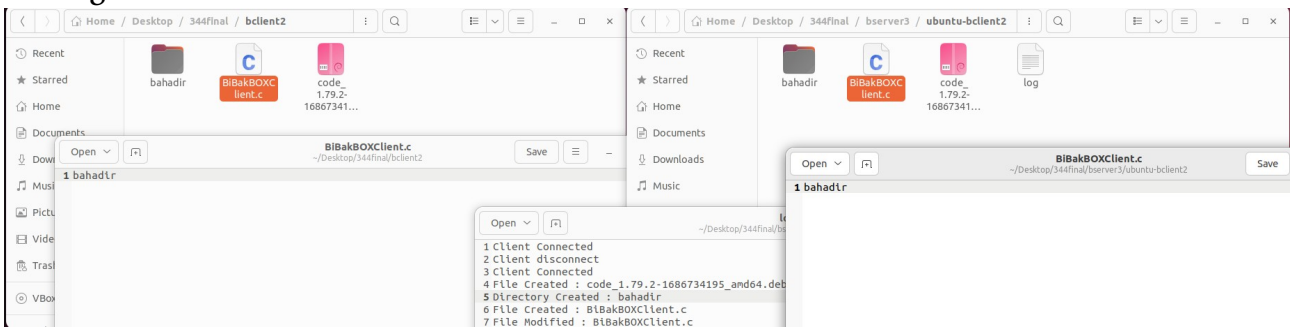
Adding a file to client



Creating file in client



Editing file in client



Removing file from client