Slovak university of technology Bratislava
Faculty of Informatics and Information Technologies

MiniDrive - Client/Server File Synchronization System (C++)

Semester project

Marek Dieška, 127133

Cvičiaci: Mgr. Jozef Melicher

Predmet: Aplikačné programovanie v C++

2025/2026

Obsah

Task

Brief description

The task was to create a client–server file synchronization system written in C++.
The application enables file transfer and synchronization between a local machine and a remote server. The system consists of two components: a server application, which listens for client connections and manages the file repository, and a client application, which connects to the server and provides an interactive command-line interface.

Key Features

- File Operations: Upload, download, delete, list files
- Directory Operations: Create, remove, move, copy directories
- Synchronization: One-way sync from local to remote with hash-based change detection
- Authentication: Public mode (shared) and private mode (per-user isolated storage)
- Concurrent Access: Multiple clients can connect simultaneously
- Resume Transfers: Interrupted uploads can be resumed after reconnection
- Security: Password hashing with Argon2id, path traversal prevention

Technologies

- C++23
- BSD TCP Sockets
- nlohmann/json (serialization)
- libsodium (cryptography)
- spdlog (logging)
- CMake (build system)

Design decisions

Communication Protocol

JSON messages separated by newline characters. Each client request contains a command name and arguments, server responds with status and data. File chunks are Base64-encoded to keep everything as text within JSON structure.

Networking

BSD sockets with blocking I/O and timeouts. Server creates a new thread for each client connection, providing simple isolation between clients without complex async logic.

File Transfer

Files are split into 64KB chunks for transfer. During upload, data is written to a .part file and renamed only after SHA-256 hash verification succeeds. This prevents incomplete or corrupted files from appearing in the file system.

Authentication

Passwords are hashed using Argon2id with a random salt per user. Credentials stored in JSON files within each user's directory. Authenticated users see a virtual filesystem with /private (isolated) and /public (shared) directories.

Resume Transfers

Server tracks incomplete uploads in a JSON file. On disconnect or shutdown, current progress is saved. Client can resume from the last received byte on reconnection. Incomplete uploads are cleaned up after one hour. Only available in authenticated mode.

Synchronization

Access to shared resources is synchronized to prevent race conditions. Updates to /public files, directories, and metadata are atomic. The server ensures that all clients see a consistent view of shared files, and concurrent modifications are detected and resolved according to a last-write-wins or conflict-resolution policy.

Path Security

All paths validated against user's root directory. Sequences like ../ are rejected to prevent directory traversal. Resolved paths must stay within permitted boundaries.

Error Handling

Errors use numeric codes organized by category (auth, files, directories, etc.). Server catches all exceptions and returns appropriate error responses without crashing. Client disconnections don't affect other clients.

Protocol description

The project uses a simple client–server protocol over a persistent TCP connection. Each connection represents a user session, which can operate in either public or authenticated (private) mode. The client sends commands to the server, and the server responds with either a success or an error message.

Commands and responses are exchanged as JSON-formatted text messages. Each client message includes a command field and optional parameters, while each server response includes a status field and optional data or error information. The client sends commands sequentially, waiting for the server’s response before issuing the next command. The server supports multiple concurrent client sessions independently.

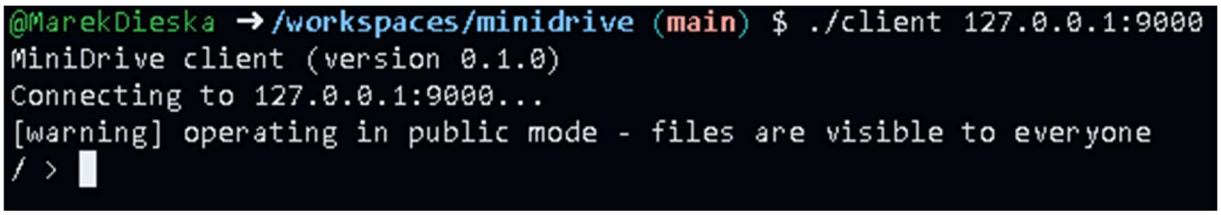
File uploads and downloads are performed in binary mode using fixed-size chunks of 64 kB. Control information is exchanged before the transfer begins, and the protocol allows resuming interrupted transfers in authenticated mode by specifying chunk ranges.

The server enforces strict access restrictions to the user's root directory, sanitizes all file paths to prevent traversal attacks, and handles errors gracefully. This protocol directly supports all required project functionality, including file management, synchronization, authentication, and concurrent client handling.

Examples

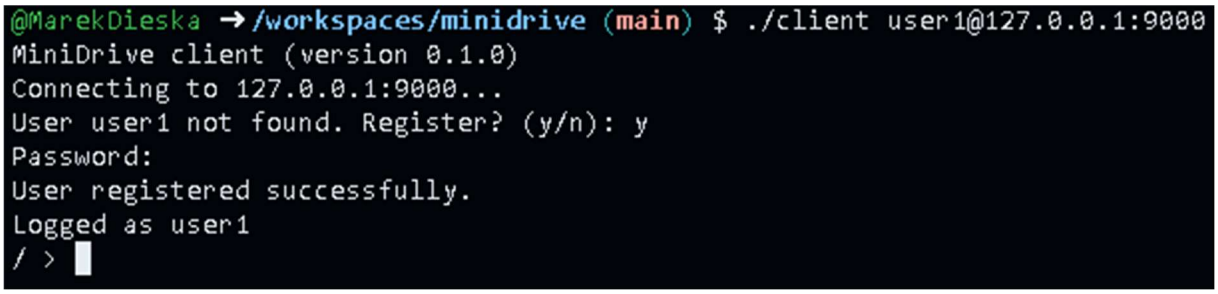
Running the server

Running client in public mode

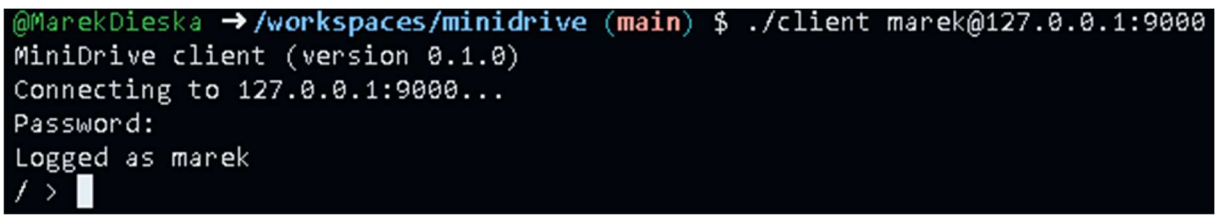


Running client in private mode

Register new user



Log-in user - successful



Log-in user – unsuccessful

```
@MarekDieska ➜ /workspaces/minidrive (main) $ ./client marek@127.0.0.1:9000
MiniDrive client (version 0.1.0)
Connecting to 127.0.0.1:9000...
Password:
Authentication failed: Invalid password
@MarekDieska ➜ /workspaces/minidrive (main) $
```

Listing and changing directories

```
Logged as marek
/ > LIST
OK
  [DIR]  private/
  [DIR]  public/
Total: 2 items
/ > CD private
OK
Current directory: /private
/private >
```

Uploading and deleting a file

```
/public > UPLOAD /tmp/nothing.txt nothing.txt
Uploading /tmp/nothing.txt...
ERROR: 100
Local file not found
/public > UPLOAD /tmp/test.txt test1.txt
Uploading /tmp/test.txt...
[========================================] 100% 13 B/13 B
OK
/public > LIST
OK
  [FILE] test1.txt (13 B)
Total: 1 items
/public > DELETE test1.txt
OK
/public > LIST
OK
Total: 0 items
/public >
```

```
/public > UPLOAD /tmp/test.txt test1.txt
Uploading /tmp/test.txt...
ERROR: 101
File already exists
/public > DELETE test1.txt
OK
/public > DELETE test1.txt
ERROR: 100
File not found
/public >
```

Creating and removing a directory

```
/public > MKDIR directory
OK
/public > CD directory
OK
Current directory: /public/directory
/public/directory > RMDIR /directory
ERROR: 5
Path traversal not allowed
/public/directory > CD /public
OK
Current directory: /public
/public > RMDIR directory
OK
/public > LIST
OK
Total: 0 items
/public >
```

```
/public > RMDIR directory
ERROR: 200
Directory not found
/public >
```

Downloading a file from the drive

```
/private > UPLOAD /tmp/test.txt test1.txt
Uploading /tmp/test.txt...
[========================================] 100% 13 B/13 B
OK
/private > DOWNLOAD test1.txt /workspaces/minidrive/files/test2.txt
Downloading test1.txt...
[========================================] 100% 13 B/13 B
OK
/private > DOWNLOAD nonexistent /workspaces/minidrive/files/test3.txt
Downloading nonexistent...
ERROR: 105
File hash mismatch
/private >
```

Copy and Move file

```
/private > LIST
OK
  [FILE] test1.txt (13 B)
Total: 1 items
/private > COPY test1.txt test2.txt
OK
/private > LIST
OK
  [FILE] test1.txt (13 B)
  [FILE] test2.txt (13 B)
Total: 2 items
/private > MOVE test2.txt /public/test2.txt
OK
/private > LIST
OK
  [FILE] test1.txt (13 B)
Total: 1 items
/private > CD /public
OK
Current directory: /public
/public > LIST
OK
  [FILE] test2.txt (13 B)
Total: 1 items
/public >
```

Path traversal safety

```
/ > cd ..
ERROR: 5
Path traversal not allowed
/ >
```

SYNC

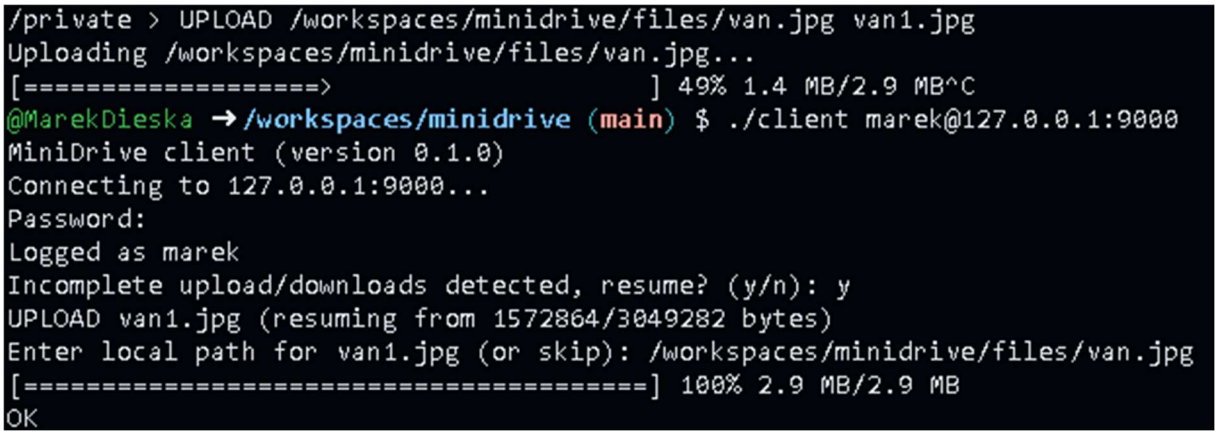
New SYNC

```
Current directory: /private
/private > MKDIR backup
OK
/private > SYNC /tmp/project backup
Synchronizing /tmp/project -> backup...
[======================================] 100% 10 B/10 B
[======================================] 100% 17 B/17 B
[======================================] 100% 25 B/25 B
Uploading: README.md
Uploading: src/utils.cpp
Uploading: src/main.cpp

Sync complete:
  Uploaded: 3
  Deleted:  0
  Skipped:  0
/private > LIST backup
OK
  [FILE] README.md (10 B)
  [DIR]  src/
Total: 2 items
/private > LIST backup/src
OK
  [FILE] utils.cpp (17 B)
  [FILE] main.cpp (25 B)
Total: 2 items
/private >
```

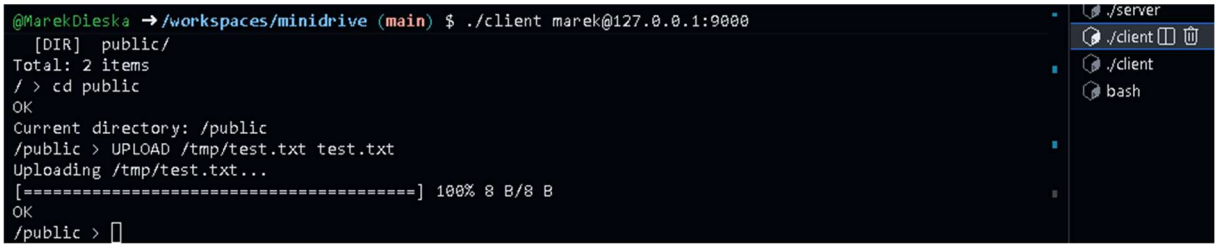
SYNC after change

```
/private > SYNC /tmp/project backup
Synchronizing /tmp/project -> backup...
[======================================] 100% 25 B/25 B
Updating: src/main.cpp
Deleting: src/utils.cpp

Sync complete:
  Uploaded: 1
  Deleted:  1
  Skipped:  1
/private >
```

Resuming interrupted upload



Concurrent users



HELP command

```
/public > HELP

Available commands:

Local Commands:
  HELP                          Show this help message
  EXIT                          Close connection and exit

File Commands:
  LIST [path]                   List files in directory
  UPLOAD <local> [remote]       Upload file to server
  DOWNLOAD <remote> [local]     Download file from server
  DELETE <path>                 Delete file on server

Directory Commands:
  CD <path>                     Change current directory
  MKDIR <path>                  Create directory
  RMDIR <path>                  Remove directory (recursive)
  MOVE <src> <dst>              Move/rename file or directory
  COPY <src> <dst>              Copy file or directory

Synchronization:
  SYNC <local> <remote>         Sync local directory to server

Notes:
  - Paths starting with / are absolute (relative to user root)
  - Other paths are relative to current directory
  - Use quotes for paths with spaces: "my file.txt"

/public >
```

EXIT

```
/public > EXIT
Goodbye!
@MarekDieska ➜ /workspaces/minidrive (main) $
```

Server logging

```
@MarekDieska ➜ /workspaces/minidrive (main) $ ./server --port 9000 --root ./data
[2025-12-23 16:03:31.994] [info] [127.0.0.1:51162#1] User authenticated: marek
[2025-12-23 16:04:43.659] [info] [127.0.0.1:51162#1] Upload complete: van1.jpg (3049282 bytes)
[2025-12-23 16:05:23.314] [error] [127.0.0.1:51162#1] Send error: Broken pipe
[2025-12-23 16:05:23.314] [info] [127.0.0.1:51162#1] Client handler stopped
[2025-12-23 16:05:31.798] [info] New connection from 127.0.0.1 (id: 127.0.0.1:47410#2)
[2025-12-23 16:05:31.798] [info] [127.0.0.1:47410#2] Client handler started
[2025-12-23 16:05:38.906] [info] User authenticated: marek
[2025-12-23 16:05:38.907] [info] [127.0.0.1:47410#2] User authenticated: marek
[2025-12-23 16:06:02.877] [info] [127.0.0.1:47410#2] Upload complete: van1.jpg (3049282 bytes)
[2025-12-23 16:11:24.476] [info] New connection from 127.0.0.1 (id: 127.0.0.1:52640#3)
[2025-12-23 16:11:24.476] [info] [127.0.0.1:52640#3] Client handler started
[2025-12-23 16:11:31.119] [info] User authenticated: user1
[2025-12-23 16:11:31.119] [info] [127.0.0.1:52640#3] User authenticated: user1
[2025-12-23 16:11:46.757] [warning] [127.0.0.1:47410#2] Client timeout
[2025-12-23 16:11:46.757] [info] [127.0.0.1:47410#2] Client handler stopped
[2025-12-23 16:12:26.815] [info] New connection from 127.0.0.1 (id: 127.0.0.1:42370#4)
[2025-12-23 16:12:26.815] [info] [127.0.0.1:42370#4] Client handler started
[2025-12-23 16:12:34.851] [info] User authenticated: marek
[2025-12-23 16:12:34.851] [info] [127.0.0.1:42370#4] User authenticated: marek
[2025-12-23 16:13:56.959] [info] [127.0.0.1:42370#4] Client handler stopped
[2025-12-23 16:14:07.502] [info] New connection from 127.0.0.1 (id: 127.0.0.1:56880#5)
[2025-12-23 16:14:07.502] [info] [127.0.0.1:56880#5] Client handler started
[2025-12-23 16:14:13.436] [info] User authenticated: marek
[2025-12-23 16:14:13.437] [info] [127.0.0.1:56880#5] User authenticated: marek
[2025-12-23 16:14:37.121] [info] [127.0.0.1:56880#5] Upload complete: test.txt (8 bytes)
[2025-12-23 16:19:58.277] [warning] [127.0.0.1:56880#5] Client timeout
[2025-12-23 16:19:58.277] [info] [127.0.0.1:56880#5] Client handler stopped
[2025-12-23 16:19:58.277] [warning] [127.0.0.1:52640#3] Client timeout
[2025-12-23 16:19:58.277] [info] [127.0.0.1:52640#3] Client handler stopped
^C[2025-12-23 16:24:48.318] [info] Shutdown signal received
```

Commands after server stopped

Error codes

Code	Name	Description
1	UNKNOWN_ERROR	Unspecified error
100	INVALID_COMMAND	Unknown or malformed command
101	INVALID_ARGUMENT	Missing or invalid argument
102	INVALID_JSON	Malformed JSON message
200	AUTH_REQUIRED	Authentication required
201	AUTH_FAILED	Invalid credentials
202	USER_NOT_FOUND	User does not exist
203	USER_EXISTS	Username already taken
300	PATH_TRAVERSAL_DENIED	Attempted directory traversal outside root
301	PATH_NOT_FOUND	Path does not exist
400	FILE_NOT_FOUND	File does not exist
401	FILE_READ_ERROR	Cannot read file
402	FILE_WRITE_ERROR	Cannot write file
403	FILE_ALREADY_EXISTS	File already exists
404	FILE_TOO_LARGE	File exceeds size limit (4GB)
405	FILE_HASH_MISMATCH	Hash verification failed
500	DIR_NOT_FOUND	Directory does not exist
501	DIR_NOT_EMPTY	Directory is not empty
502	DIR_ALREADY_EXISTS	Directory already exists
600	SERVER_ERROR	Internal server error
601	NOT_IMPLEMENTED	Feature not implemented

Conclusion

The solution fulfills all required functionality specified in the assignment. The system implements file operations (upload, download, delete, list), directory management (create, remove, move, copy), and one-way synchronization with hash-based change detection.

Users can operate in public mode with shared storage or authenticate for private directories. Passwords are securely hashed and never stored in plain text. The server handles multiple concurrent clients using a thread-per-client model.

Resume functionality allows interrupted uploads to be continued after reconnection, with automatic cleanup of incomplete transfers after timeout.

The project uses only permitted libraries: BSD sockets for networking, nlohmann/json for serialization, libsodium for cryptography, and spdlog for logging. Bonus features include structured logging and multiple simultaneous sessions per user.