Architecture Plan

Introduction

UFmyMusic will be a networked application that synchronizes files from multiple machines. It will follow the client-server architecture, where the server is a centralized machine that houses the current state of files.

Client-Server Architecture

The server should be capable of handling multiple client connections simultaneously. It should also be able to store files and respond to requests while giving the appropriate response.

The client should communicate with the server via a command line interface. They solely have the ability of initiating and breaking down a session with the server.

Message Structs

The messages the server will respond to are LIST, DIFF, PULL, and LEAVE.

There are two types of messages that can be exchanged, a request and a response. These are structs that contain fields each use respectively.

A request contains the service and a map, with filenames as the keys and hashes as the values.

A response contains the service, a similar, map, an output message, and an error message.

If the client sends a LIST request, the server will send a LIST response that will contain the filenames of all the files it currently has in the output field

If the client sends a DIFF request, the server will send a DIFF response that will contain the filenames of the files that aren't in sync in the output field

If the client sends PULL request, the server will send a PULL response that will contain whether or not the client should prepare for file transfer in the output field. When files have to be transferred, it will send it in chunks, and denote the end of file transfer by sending EOF_ALL

If the client sends a LEAVE request, it will close the client's connection to the server

Anything else will respond with an error message detailing the only messages the server can respond to.

Hashes will be computed using the SHA-256 algorithm.

We will use phthreads in order to make the application multithreaded. Each client performs independent tasks that take varying amount of time and performance, so having separate threads will ensure complete independence. Not only that, but file I/O will be able to be done concurrently, making our code less complex.