

CS 550 - Spring 2022
Isaias Rivera
A20442116

P2P File Sharing - Manual

Note

This manual only covers how to use the program. Terms that will be not explained in this manual are `ip address`, `ports`, `port forwarding`, `directories`, and `file hashes`. If this program is only needed to run locally, the *Simple* examples should work just fine. However, this manual is also mainly for Windows, meaning, the exact syntax of some commands may depend on your system.

Procedure

The typical procedure for using this program is as follows

1. Create valid folders for each client
2. Move files to folders for each client that want to be shared
3. Start the server
4. Start at least two clients, using the folders previously created
5. On one client, search for file
6. Request file, given the file hash (The long string of characters) from searching
7. Wait for file download or repeat steps 2 or 5-6
8. Close all clients
9. Close server

Program arguments

A `<int>` just means it should be a number.

Server

Usage:

Server `[-h] [-version] [-p <int>]`

Where:

`-p <int>`, `-port <int>` The port this server should listen to

Example

Simple - Local Only `.\Server.exe`

Given Args `.\Server.exe -p 44455`

Client

Usage:

Client [-h] [-version] [-c <int>] [-e <int>] [-s <ip address>] -f <directory> -i <int>

Where:

- i <int>, -identity <int> (required) Unique ID identifying this client
- s <ip address>, -serverIP <ip address> The indexing server IP address this client should connect to
- e <int>, -serverPort <int> The indexing server Port this client should use
- c <int>, -clientPort <int> The client Port other peers should connect to
- f <directory>, -downloadFolder <directory> (required) The local folder files should be uploaded and downloaded to

Example

Simple - Local Only `.\Client.exe -i 45 -f "watchFolder"`

Make sure to replace **watchFolder** with a folder directory that exists, otherwise, create a folder next to the program named **watchFolder**

Given Args `.\Client -i 791 -c 44910 -s "192.168.1.200" -e 55555 -f "../..../testFolder"`

Using the Interactive Console

The main interface with this program involves using the terminal of your device when using the client.

The following options are allowed when using the client.

- **ping**
 - Ping the server to check it's response time, meaning, how long the server takes to respond
- **list**
 - List all the files on the indexing server, meaning, show all the files we can either download or already have
- **search [query]**
 - Search for the query as a substring the in the name of all the files on the indexing server, meaning, search for a file where **query** is what you want to look for
 - Example: **search am**
 - * Search for files with an **am** in it's name
- **request [hash]**
 - Request a specific file, given the hash, meaning, copy the **hash** (The long string of characters) from a single file when you did either a **search** or **list** and paste it where **hash** is.
 - Example: **request E2D0FE1585A63EC6009C8016FF8DDA8B17719A637405A4E23C0FF81339148249**
- **q or quit or exit**
 - Stop the interactive console, this will close the client

To stop the server simply close the terminal or press **Ctrl + C**

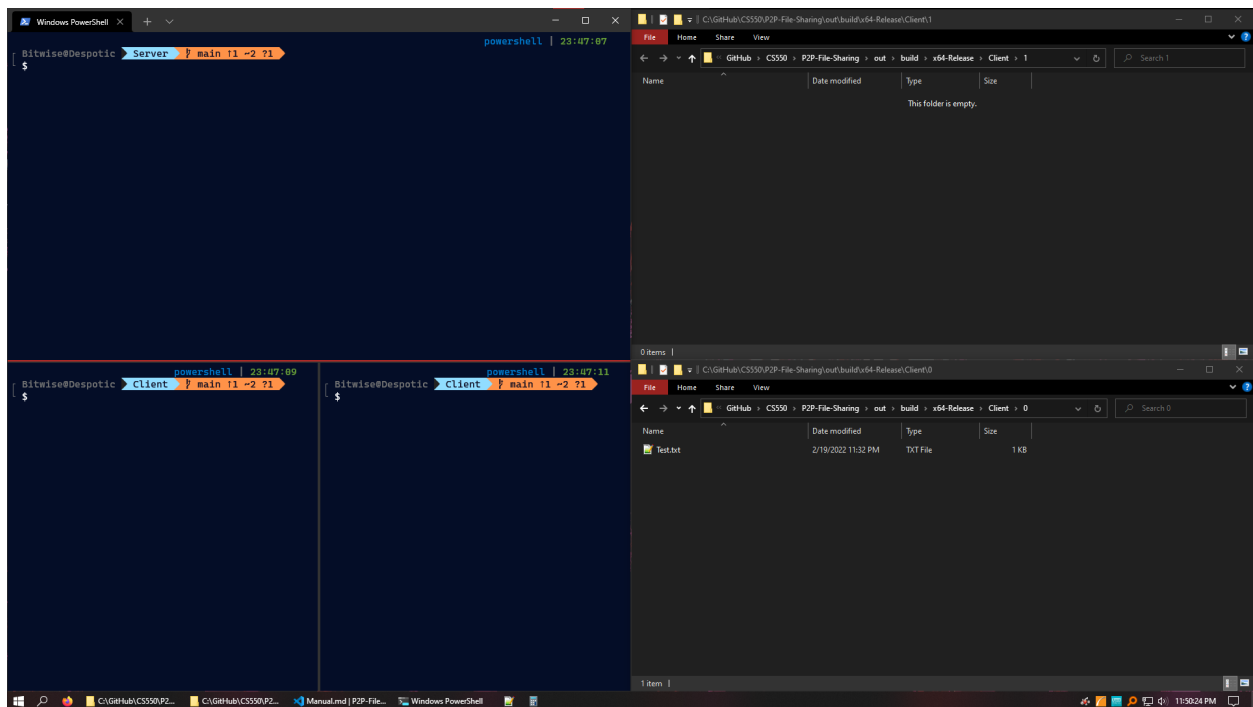
Remember, to add or remove files for sharing. Simply delete or add them to your folder that you initially created for the client.

Overall Example

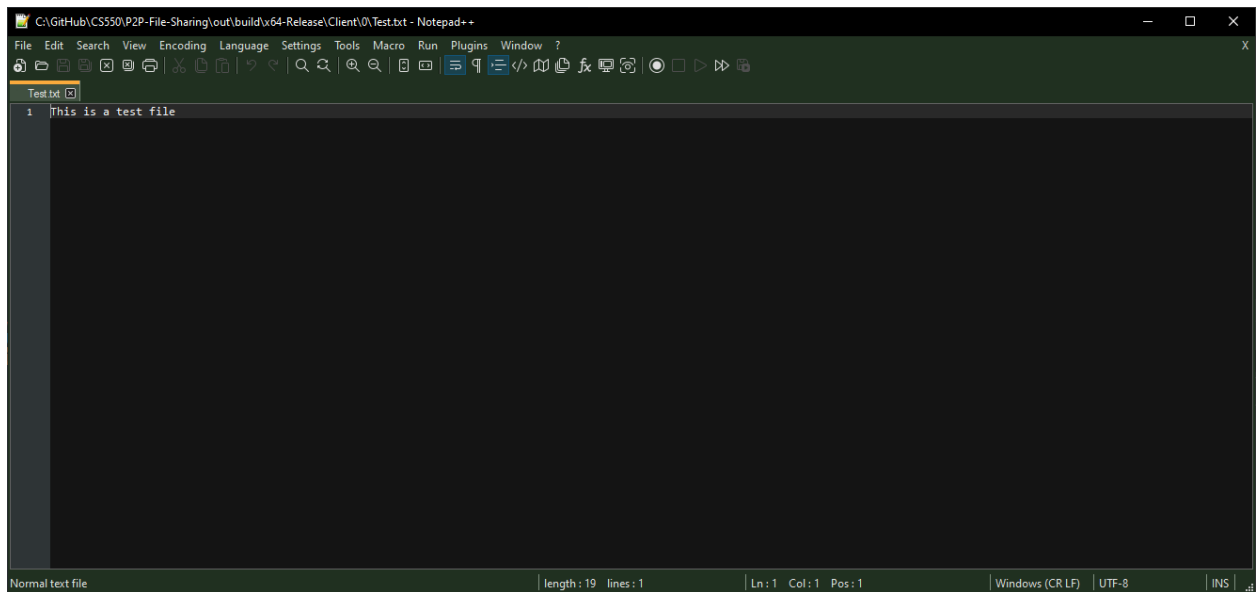
This example will demonstrate following the procedures for a *simple* local case.

Initial setup

Here I have layed everything out onto one screen, simply take note of what everything is. Note that your terminal may be separate windows instead of just one.

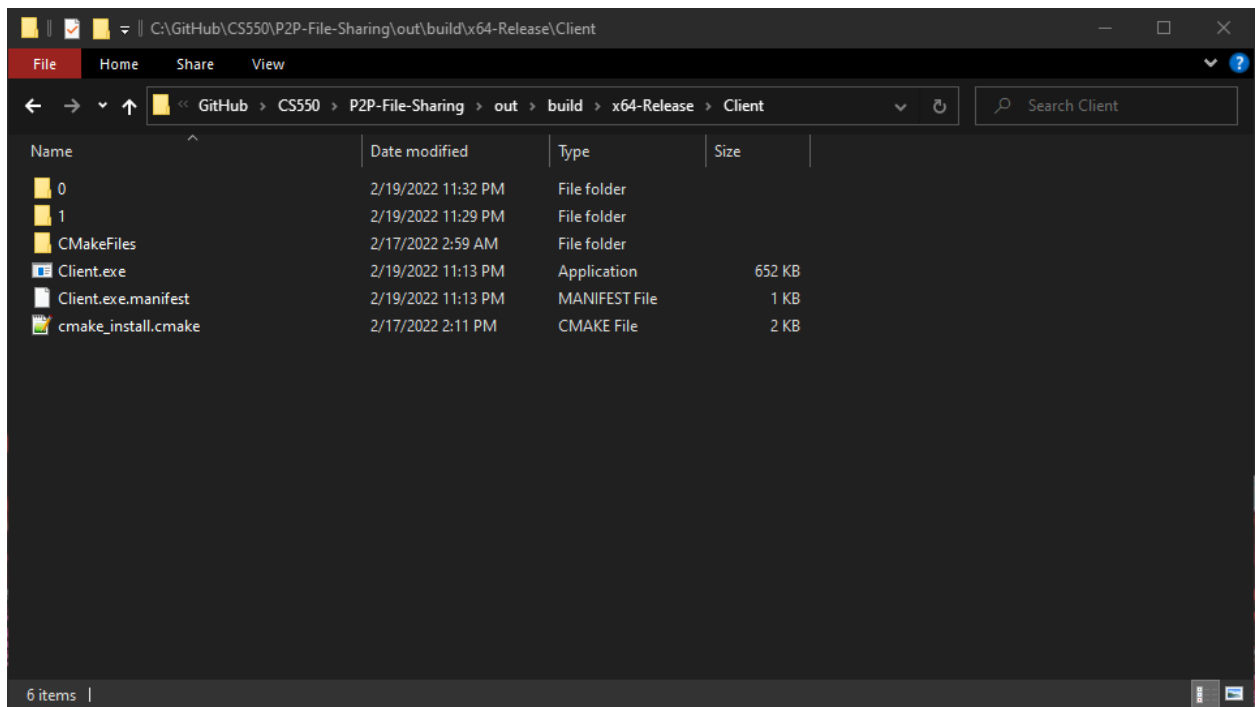


Here we can also see the file contents for my test file.



A screenshot of the Notepad++ application. The title bar shows the file path: C:\GitHub\CS550\P2P-File-Sharing\out\build\x64-Release\Client\0\Test.txt - Notepad++. The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, Window, and ?. The toolbar contains various icons for file operations and editing. The text area shows a single line of text: "This is a test file". The status bar at the bottom indicates "Normal text file", "length: 19 lines: 1", "Ln: 1 Col: 1 Pos: 1", "Windows (CR LF)", "UTF-8", and "INS".

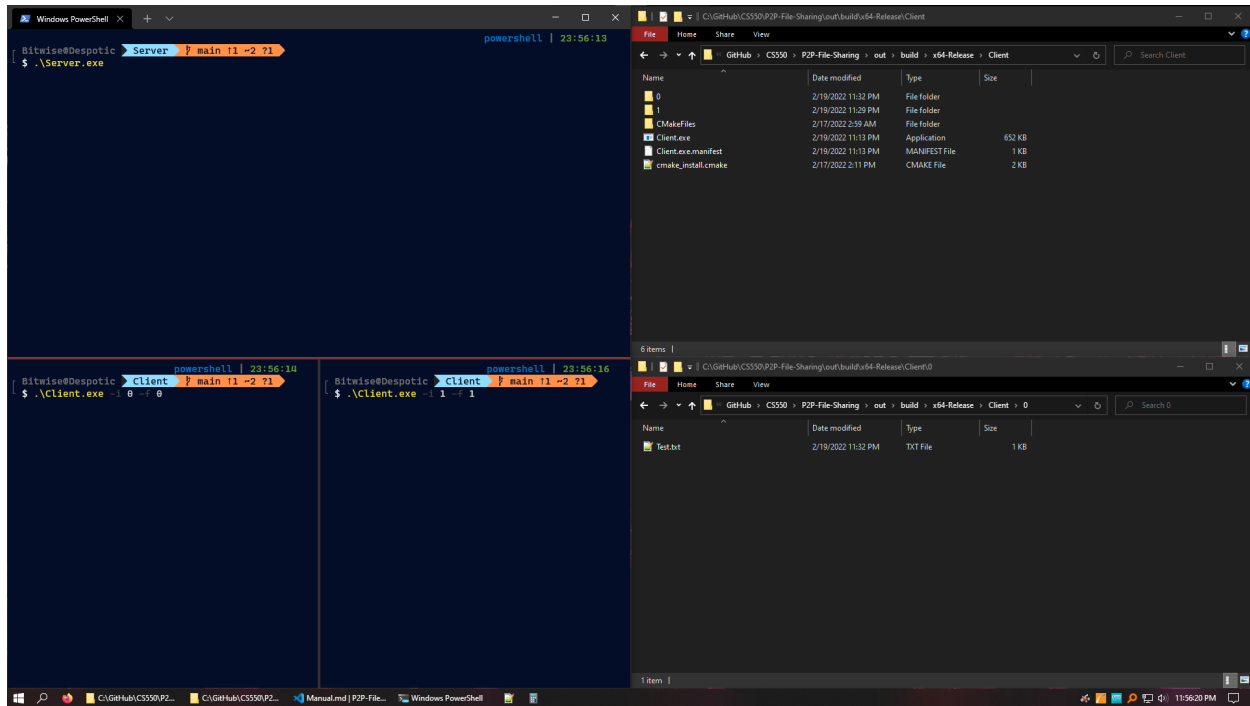
And before that we can see where the directories are relative to the program.



And just as a tip, you can shift right click on Windows in the same directory as the program to more easily open a terminal.

Start Programs

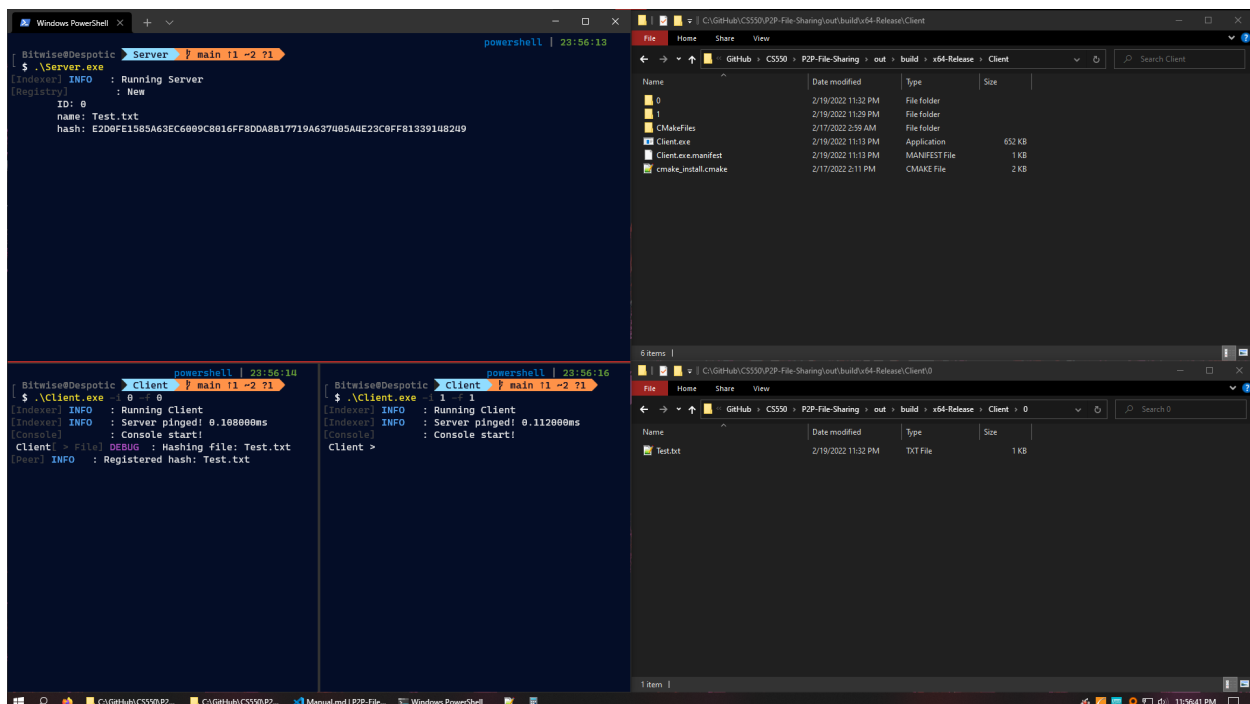
Here we can see the commands are setup to be run.



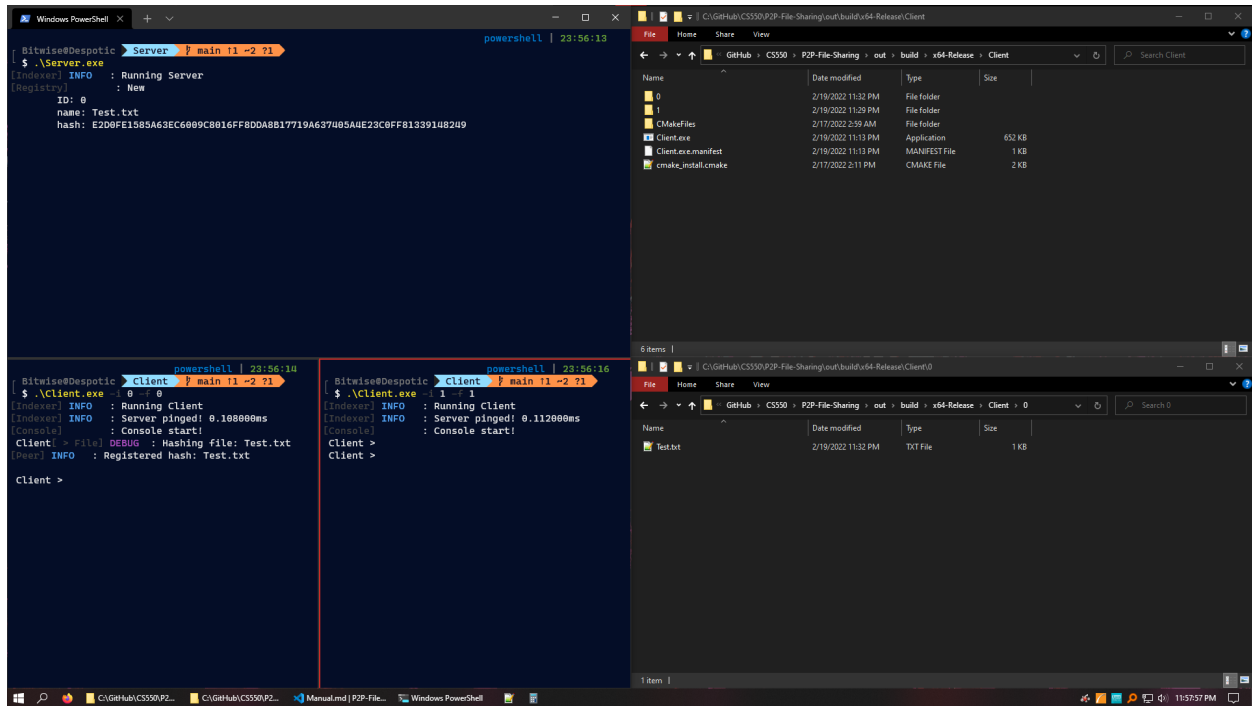
`.\Server.exe` to start the server

`.\Client.exe -i 0 -f 0` to start the client with id 0 and watching the folder named 0

`.\Client.exe -i 1 -f 1` to start the client with id 1 and watching the folder named 1



After running them all, you might notice the prompt `Client >` becomes a bit garbled, simply pressing `enter` once should fix it



```
Bitwise@Despotic: ~$ .\Server.exe
[Indexer] INFO : Running Server
[Register] INFO : New
ID: 0
name: Test.txt
hash: E2D8FE1585A63EC6009C8016FF8DDA8B17719A657405A0E23C0FFB1339140249

Bitwise@Despotic: ~$ .\Client.exe
[Indexer] INFO : Running Client
[Register] INFO : Server pinged! 0.108800ms
[Console] INFO : Console start!
Client > File DEBUI : Hashing file: Test.txt
[Power] INFO : Registered hash: Test.txt
Client >

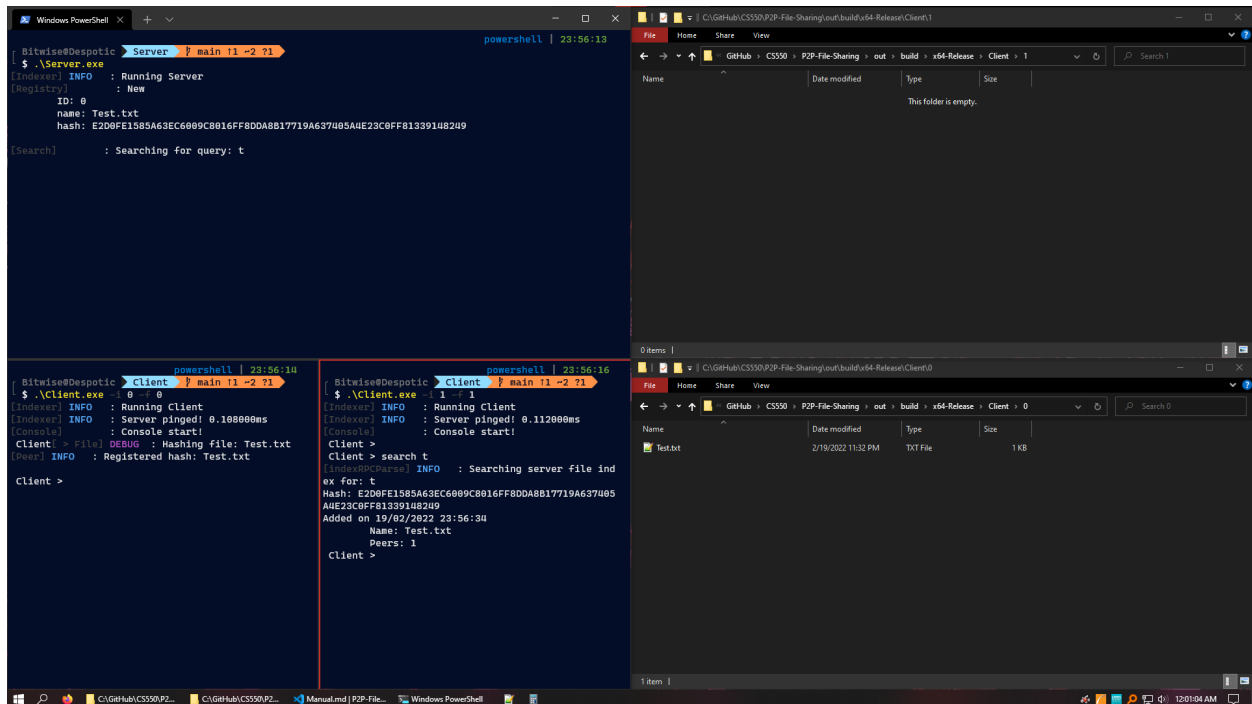
Bitwise@Despotic: ~$ .\Client.exe
[Indexer] INFO : Running Client
[Register] INFO : Server pinged! 0.112000ms
[Console] INFO : Console start!
Client >
```

File Explorer: C:\GitHub\CS550\P2P-File-Sharing\out\build\vs64-Release\Client

Name	Date modified	Type	Size
0	2/19/2022 11:32 PM	File folder	
1	2/19/2022 11:39 PM	File folder	
chakefiles	2/17/2022 2:39 AM	File folder	
Client.exe	2/19/2022 11:13 PM	Application	653 KB
Client.exe.manifest	2/19/2022 11:13 PM	MANIFEST File	1 KB
cmake_install.cmake	2/17/2022 2:11 PM	CMAKE File	2 KB

Running a search

Here we are searching for the file we want, just looking for the letter `t` is enough. Note that we are using the client that is not watching the folder with the test file already in it.



```
Bitwise@Despotic: ~$ .\Server.exe
[Indexer] INFO : Running Server
[Register] INFO : New
ID: 0
name: Test.txt
hash: E2D8FE1585A63EC6009C8016FF8DDA8B17719A657405A0E23C0FFB1339140249

[Search] : Searching for query: t

Bitwise@Despotic: ~$ .\Client.exe
[Indexer] INFO : Running Client
[Register] INFO : Server pinged! 0.112000ms
[Console] INFO : Console start!
Client > File DEBUI : Hashing file: Test.txt
[Power] INFO : Registered hash: Test.txt
Client >

Bitwise@Despotic: ~$ .\Client.exe
[Indexer] INFO : Running Client
[Register] INFO : Server pinged! 0.112000ms
[Console] INFO : Console start!
Client > search t
[IndexRPCParse] INFO : Searching server file index for: t
Hash: E2D8FE1585A63EC6009C8016FF8DDA8B17719A657405A0E23C0FFB1339140249
Added on 19/02/2022 23:56:34
Name: Test.txt
Peers: 1
Client >
```

File Explorer: C:\GitHub\CS550\P2P-File-Sharing\out\build\vs64-Release\Client\1

This folder is empty.

Make sure to copy the hash, in this case it would be E2D0FE1585A63EC6009C8016FF8DDA8B17719A637405A4E23C0FF81339148249

Requesting a file

Now we need to actually request this file by running the following command on the same client.

request E2D0FE1585A63EC6009C8016FF8DDA8B17719A637405A4E23C0FF81339148249

After this runs, we can now see that the file is in the second directory.

The screenshot shows a Windows PowerShell session and a file explorer window. The PowerShell session is running the 'Server.exe' and 'Client.exe' programs. The 'Server.exe' output shows the registration of a file named 'Test.txt' with a specific hash. The 'Client.exe' output shows the client requesting the file, receiving the hash, and then streaming the file. The file explorer window shows the file 'Test.txt' in the directory 'C:\GitHub\CS550\P2P-File-Sharing\out\build\x64-Release\Client\1'.

```
BitwiseDespotic > Server.exe
[Indexer] INFO : Running Server
[Registry] INFO : New
ID: 0
name: Test.txt
hash: E2D0FE1585A63EC6009C8016FF8DDA8B17719A637405A4E23C0FF81339148249

[Search] : Searching for query: t
[Request] : Searching for hash: E2D0FE1585A63EC6009C8016FF8DDA8B17719A637405A4E23C0FF81339148249
[Request] INFO : Entry found
name: Test.txt
hash: E2D0FE1585A63EC6009C8016FF8DDA8B17719A637405A4E23C0FF81339148249
[Registry] : New
ID: 1
name: Test.txt
hash: E2D0FE1585A63EC6009C8016FF8DDA8B17719A637405A4E23C0FF81339148249

BitwiseDespotic > Client.exe
[Indexer] INFO : Running Client
[Indexer] INFO : Server pinged! 0.138000ms
[Console] : Console start!
[ClientFile] DEBUG : Hashing file: Test.txt
[Peer] INFO : Registered hash: Test.txt

Client > [Exchanger] DEBUG : p2p conn: 127.0.0.1:60456
[Exchanger Server] DEBUG : Sending ID
[Exchanger Server] DEBUG : Reading hash size
[Exchanger Server] DEBUG : Hash get: E2D0FE1585A63EC6009C8016FF8DDA8B17719A637405A4E23C0FF81339148249
[Exchanger Server] DEBUG : File found, sending size: 19
[Exchanger Server] DEBUG : Waiting to stream
[Exchanger Server] DEBUG : Streaming
[Exchanger Server] DEBUG : Written 100%
[Exchanger Server] INFO : Finished streaming

ex for: t
Hash: E2D0FE1585A63EC6009C8016FF8DDA8B17719A637405A4E23C0FF81339148249
Added on 20/02/2022 00:05:41
Name: Test.txt
Deers: 1
Client > request E2D0FE1585A63EC6009C8016FF8DDA8B17719A637405A4E23C0FF81339148249
[Indexer] INFO : Searching server peer index for: E2D0FE1585A63EC6009C8016FF8DDA8B17719A637405A4E23C0FF81339148249
[0] 127.0.0.1:40000
Client > [Exchanger] DEBUG : Connecting to peer: 0
[Exchanger Client] DEBUG : Receiving server ID
[Exchanger Client] DEBUG : ID match, sending hash: E2D0FE1585A63EC6009C8016FF8DDA8B17719A637405A4E23C0FF81339148249
[Exchanger Client] DEBUG : Receiving file size
[Exchanger Client] DEBUG : Valid filesize, notifying server: 19
[Exchanger Client] DEBUG : Streaming
[Exchanger Client] INFO : Finished streaming
[File] DEBUG : Hashing file: Test.txt
[Peer] INFO : Registered hash: Test.txt
```

Just to make sure, we can see that the file has the same contents as it's copy.

The screenshot shows a Notepad++ window with the file 'Test.txt' open. The file contains the text 'This is a test file'.

```
1 |This is a test file
```

We now can close both clients with `q` and the server with `Ctrl+C`.

