Web 2

Introduction

This practical requires us to use the WebSockets API and in order to create server that could display the contents of a directory and traverse through them. Multiple users should be able to connect to the server and view the files without affecting others who are connected.

Usage Instructions

1. Navigate to the cs2003-web2 folder in terminal and enter the command:

   ```
   node server.js
   ```

2. The terminal should then display:

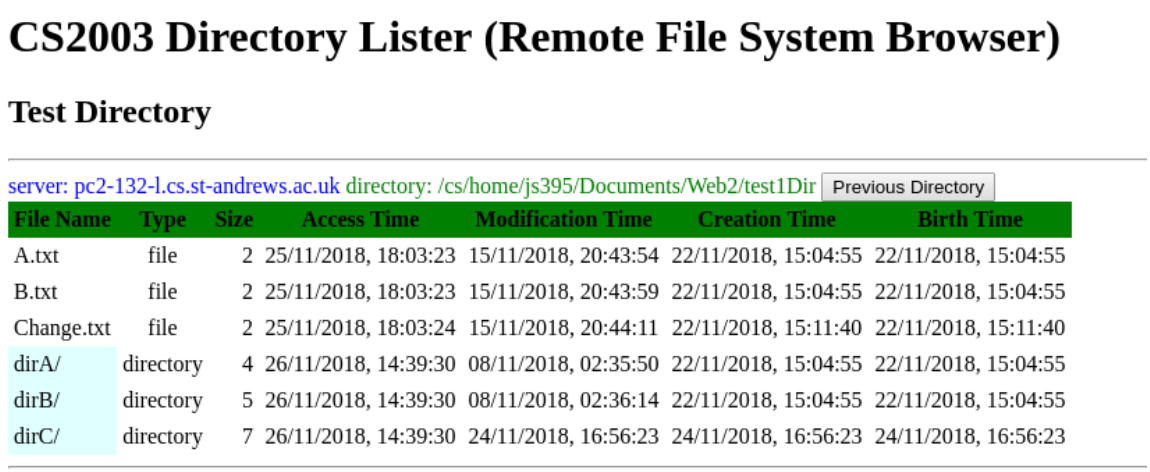
   ```
   -- server is running: <address:portnumber>
   ```

   Copy <address:portnumber> and paste it into the address bar of a internet browser of your choice (Mozilla Firefox or Google Chrome are ideal). The below directory should be displayed.

3. Should you wish to change the directory displayed, simply remove the contents of /Web2/test1Dir and put whatever files you wish to put in instead.

4. In order to traverse deeper, click on the directory.

5. In order to go to a parent directory press the ‘Previous Directory’ button in the top right.



CS2003 Directory Lister (Remote File System Browser)

Test Directory

server: pc2-132-l.cs.st-andrews.ac.uk directory: /cs/home/js395/Documents/Web2/test1Dir Previous Directory

File Name	Type	Size	Access Time	Modification Time	Creation Time	Birth Time
A.txt	file	2	25/11/2018, 18:03:23	15/11/2018, 20:43:54	22/11/2018, 15:04:55	22/11/2018, 15:04:55
B.txt	file	2	25/11/2018, 18:03:23	15/11/2018, 20:43:59	22/11/2018, 15:04:55	22/11/2018, 15:04:55
Change.txt	file	2	25/11/2018, 18:03:24	15/11/2018, 20:44:11	22/11/2018, 15:11:40	22/11/2018, 15:11:40
dirA/	directory	4	26/11/2018, 14:39:30	08/11/2018, 02:35:50	22/11/2018, 15:04:55	22/11/2018, 15:04:55
dirB/	directory	5	26/11/2018, 14:39:30	08/11/2018, 02:36:14	22/11/2018, 15:04:55	22/11/2018, 15:04:55
dirC/	directory	7	26/11/2018, 14:39:30	24/11/2018, 16:56:23	24/11/2018, 16:56:23	24/11/2018, 16:56:23

Summary

Summary of requirements completed:

- Remote directory (the ‘root’ can be viewed, parent directory of said directory cannot be viewed.
- The remote server and directory being accessed are shown in the user interface.
- Clients can navigate into the directory provided and back up to the root.
- Clients can select which details are visible.
- Multiple clients can connect to the server and view the directory and sub-directories without affecting each other.
- Client and server communication conforms to the specification provided.

Design

The important aspects of designing this practical were deciding what features needed to be on the front-end and what needed to be on the back-end as this is not immediately obvious. My implementation follows this basic structure:

1. In my implementation the server gets a directory request from the client and converts the information about files in that directory into a JSON object.
2. The server then wraps that JSON object into another in order to comply with the message format protocol and converts the JSON object to a string to transmit.
3. The client receives the string from the server and parses it back to JSON
4. Then the client converts the JSON to html markup to display on a webpage.
5. Steps 1-4 are repeated as new directories are requested by the client.

I chose to give headers to the table provided in dir_list_example.html (renamed to index.html) in order to make the columns easier to navigate. In addition, I changed the CSS to highlight the header when a user hovers over it to try and indicate its function to hide the column when it is clicked on.

Implementation

Server.js

Some of the Javascript and the CSS are kept separately from the main html page in order to make the html page easier to read, while also giving a greater degree of flexibility with implementation.

On a connection from a client the server first supplies them with the contents of the root directory. As each connection is handled separately by the server, there isn’t any extra coding required in order for the server to handle multiple clients at the same time.

On each connection the server should only receive one type of message from the front-end, a request for another directory to be displayed. In order to stop clients from accessing directories above the root provided, there is an if statement that checks the directory requested contains the entire root directory (i.e. ~/cs2003-web2/test1Dir). If it doesn't this means that the client is asking for a folder they shouldn't have access to.

Index.html

Each of the headers has an onclick event that hides the information in that column while maintaining the table structure. This was achieved by changing the visibility attribute using javascript.

Both going deeper or up to a parent directory use for very similar code. To go to a parent directory, the full directory is retrieved from the directoryname tag in the html and the last folder is removed using the slice method, this provides the parent directory. To go deeper the name of the folder being accessed is appended to the current directory. Then in both instances a request is made to the server.

There is some server-side checking to make sure that the address requested isn't above the root directory by checking that the directory requested contains the root folder name. This isn't ideal on its own as front-end validation can be bypassed, or if there is another folder on the host machine that shares the folder name in a different location, and that location can be requested successfully.

Dir_list_json.js

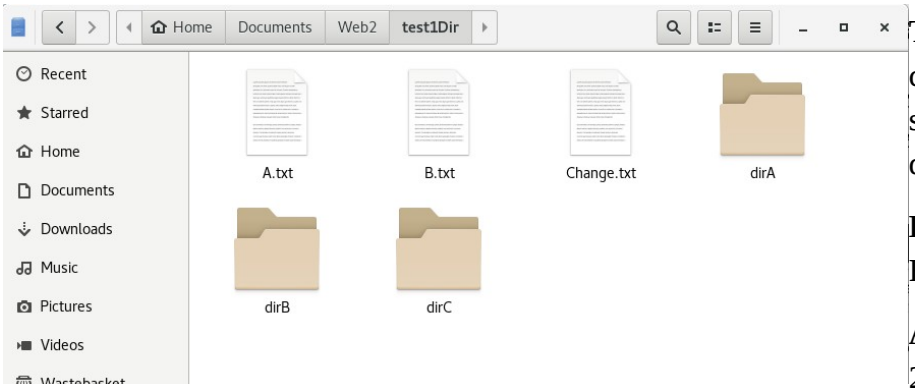
DirInfo2Markup was edited to include headers for the table(lines 174-185), with onclick events to allow the visibility of the column to be toggled. FileInfo2Markup was edited so that the markup for directories included an onclick event to allow traversal (lines 125 - 131).

Testing

Tests were conducted on a Mozilla Firefox and Google Chrome internet browser at the same time.

Test 1 – Access Root Directory

Figure 1. Test1Dir



Test Brief: Clients upon connecting to the address if the server, should be presented with the directory of /test1Dir.

Expected Results: The contents of Figure 1. should be displayed.

Actual Results: Successful Figures 2 & 3.

Test 2 – Traversal

Test Brief: Separate clients should be able to go deeper into the folder structure without affecting one another.

Input: The chrome client will enter dirA, Firefox will enter dirC.

Expected Results: Both clients display their directories successfully

Actual Results: Successful. Figures 4 & 5

Figure 2. Chrome

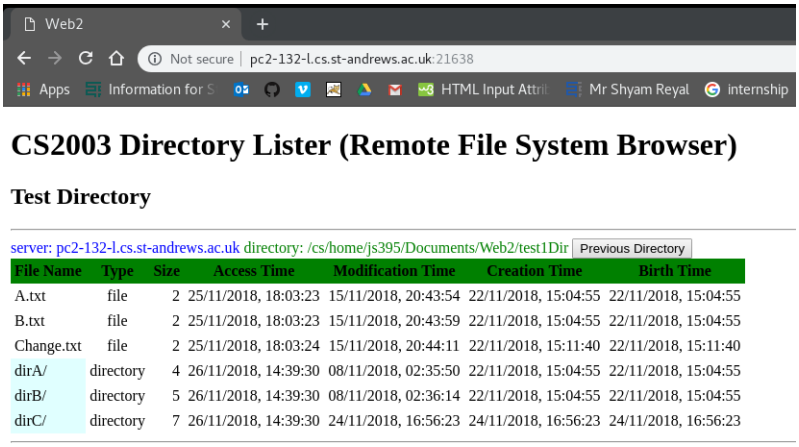


Figure 3. Firefox

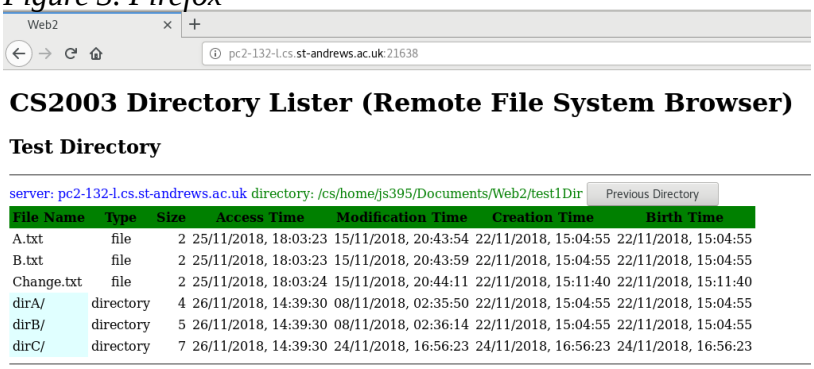
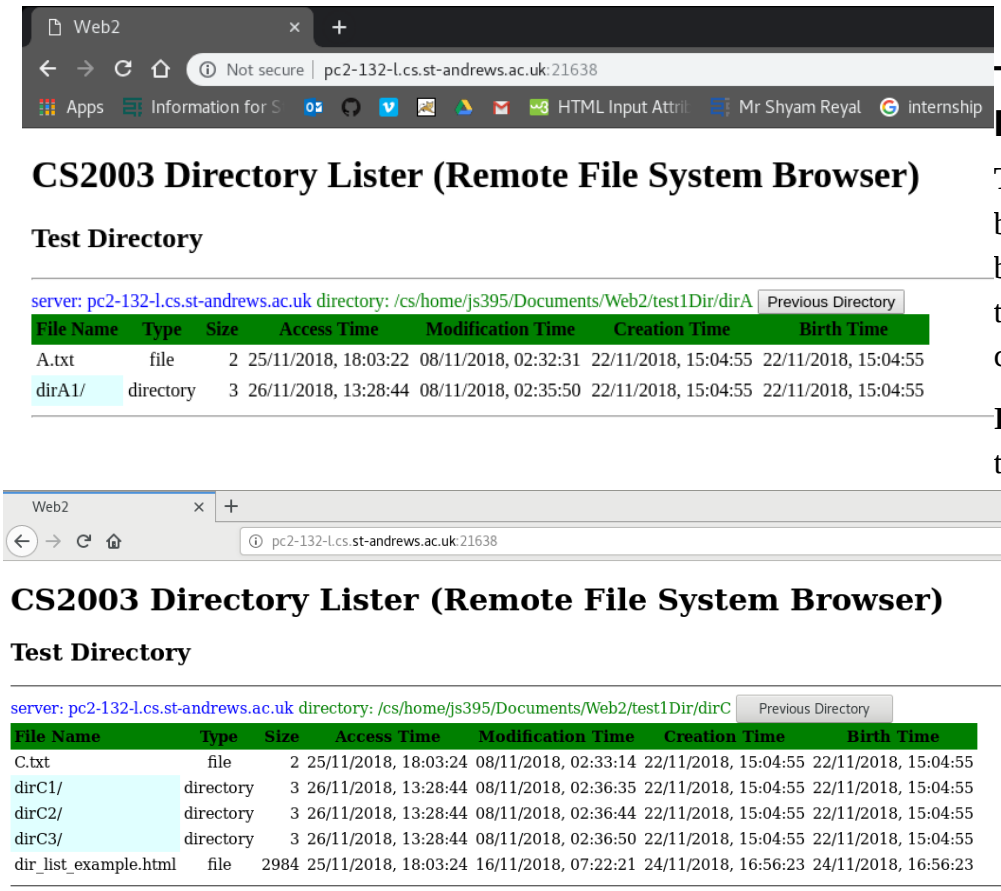


Figure 4. Chrome dirA.



Test 3 – Parent Directory

Test Brief: Clicking on the back 'Previous Directory' button should return the user to the parent directory of the current folder.

Input: From Figure 4 & 5 the client should press the back button.

Expected Result: Figure 4 & 5 should return to how they looked in figure 2 & 3.

Actual Result: Successful. Both browsers displayed the parent directory correctly.

Figure 5. Firefox dirC.

Test 4 – Access beyond root directory

Test Brief: If a user presses the 'Previous Directory' button while in the root folder, they should be informed access isn't allowed.

Inputs: While on the root directory the user should 'Previous Directory' button. (For this test we will also test the back-end by commenting out the code that prevents requests for beyond the root directory.(Test 4b)

Expected Results: 4a. Successful. Alert box displayed in both browsers.

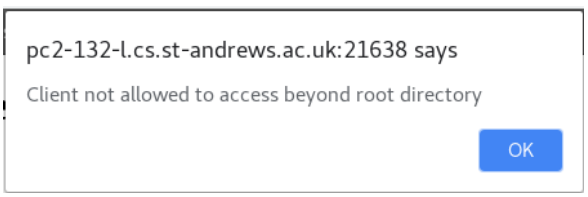
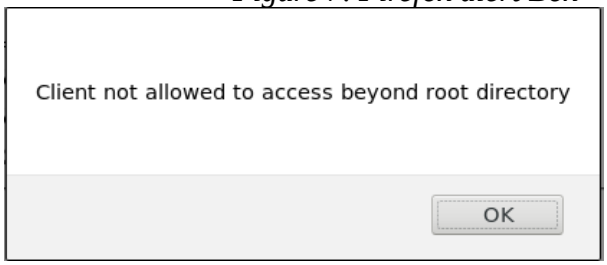


Figure 6. Chrome Alert Box

Figure 7. Firefox alert Box



4b. Successful. Figure 8.

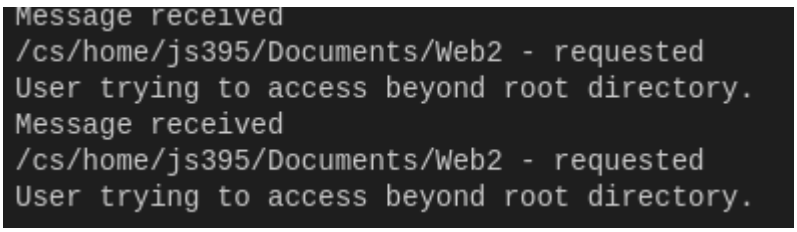


Figure 8. Console of Server as parent folders requested

Test 5 – Hide File Details

Test Brief: When a user clicks on the header of a table column that table column should become hidden from view.

Expected Result: When each header is clicked on the column should become hidden and directories shouldn't be accessible. When a hidden column header is clicked on the column should become visible again.

Actual Result: Successful. For brevity only a handful of the screen-shots from the Firefox browser are shown however results were similarly successful on Chrome, and for columns not shown hidden.

Figure 9. Filename Hidden. Firefox

File Name	Type	Size	Access Time	Modification Time	Creation Time	Birth Time
	file	2	25/11/2018, 18:03:23	15/11/2018, 20:43:54	22/11/2018, 15:04:55	22/11/2018, 15:04:55
	file	2	25/11/2018, 18:03:23	15/11/2018, 20:43:59	22/11/2018, 15:04:55	22/11/2018, 15:04:55
	file	2	25/11/2018, 18:03:24	15/11/2018, 20:44:11	22/11/2018, 15:11:40	22/11/2018, 15:11:40
	directory	4	26/11/2018, 14:39:30	08/11/2018, 02:35:50	22/11/2018, 15:04:55	22/11/2018, 15:04:55
	directory	5	26/11/2018, 14:39:30	08/11/2018, 02:36:14	22/11/2018, 15:04:55	22/11/2018, 15:04:55
	directory	7	26/11/2018, 14:39:30	24/11/2018, 16:56:23	24/11/2018, 16:56:23	24/11/2018, 16:56:23

Figure 9. Filename, type, size, access time and birth time hidden. Firefox.

File Name	Type	Size	Access Time	Modification Time	Creation Time	Birth Time
				15/11/2018, 20:43:54	22/11/2018, 15:04:55	
				15/11/2018, 20:43:59	22/11/2018, 15:04:55	
				15/11/2018, 20:44:11	22/11/2018, 15:11:40	
				08/11/2018, 02:35:50	22/11/2018, 15:04:55	
				08/11/2018, 02:36:14	22/11/2018, 15:04:55	
				24/11/2018, 16:56:23	24/11/2018, 16:56:23	

Figure 9. All details restored. Firefox.

File Name	Type	Size	Access Time	Modification Time	Creation Time	Birth Time
A.txt	file	2	25/11/2018, 18:03:23	15/11/2018, 20:43:54	22/11/2018, 15:04:55	22/11/2018, 15:04:55
B.txt	file	2	25/11/2018, 18:03:23	15/11/2018, 20:43:59	22/11/2018, 15:04:55	22/11/2018, 15:04:55
Change.txt	file	2	25/11/2018, 18:03:24	15/11/2018, 20:44:11	22/11/2018, 15:11:40	22/11/2018, 15:11:40
dirA/	directory	4	26/11/2018, 14:39:30	08/11/2018, 02:35:50	22/11/2018, 15:04:55	22/11/2018, 15:04:55
dirB/	directory	5	26/11/2018, 14:39:30	08/11/2018, 02:36:14	22/11/2018, 15:04:55	22/11/2018, 15:04:55
dirC/	directory	7	26/11/2018, 14:39:30	24/11/2018, 16:56:23	24/11/2018, 16:56:23	24/11/2018, 16:56:23