

Lesson 06 Demo 01

Executing File System Commands

Objective: To execute file system operations like reading, writing, and deleting to manage files within a file system

Tools required: Linux Cent OS, Node Package Manager, and Visual Studio Code

Prerequisites: Basic Linux Commands, NPM commands, JavaScript, and file system module

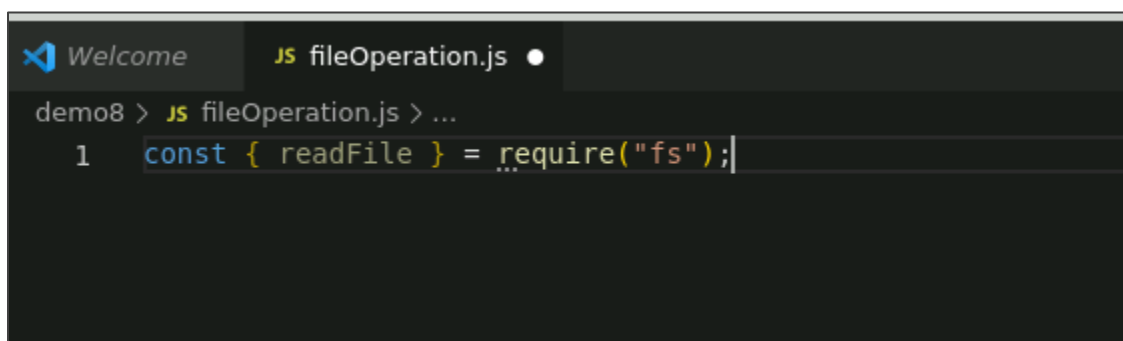
Steps to be followed:

1. Read files using the **fs.readFile()** method
2. Write into the files using the **fs.writeFile()** method
3. Delete files using the **unlink(path, callback)** method

Step 1: Read files using the fs.readFile() method

- 1.1 Import the file system module in the **fileOperation.js** file:

```
const { readFile } = require("fs");
```



The screenshot shows a code editor with a dark theme. At the top, there are two tabs: 'Welcome' and 'JS fileOperation.js'. Below the tabs, the command prompt shows 'demo8 > JS fileOperation.js > ...'. The first line of code is '1 const { readFile } = require("fs");' with the text highlighted in blue.

- 1.2 Write the following non-blocking task that reads the **essay.md** file from the system and print the contents on the console:

```
// Non-Blocking Task
readFile("./essay.md", (error, data) => {
  if(error) {
    console.error(error);
    return;
  }

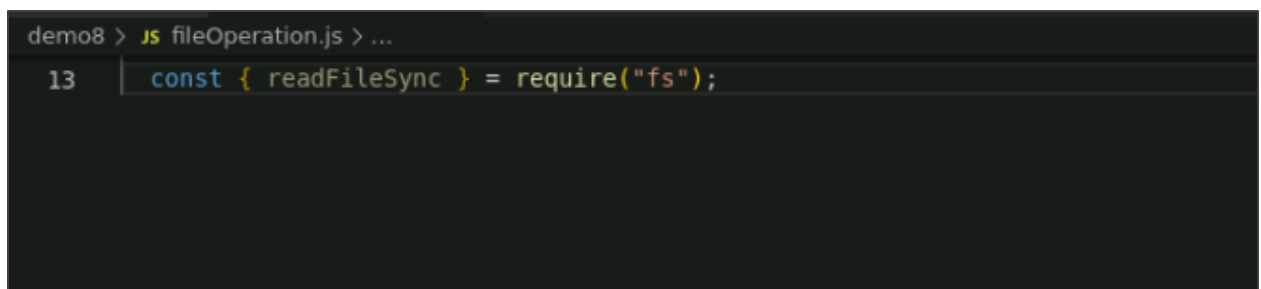
  console.log(">>> File Content\n\b", data.toString());
})
```

A screenshot of a code editor with a dark theme. The editor has a tab labeled 'JS fileOperation.js'. The code is as follows:

```
demo8 > JS fileOperation.js > ...
1  const { readFile } = require("fs");
2
3  // Non-Blocking Task
4  readFile("./essay.md", (error, data) => {
5    if(error) {
6      console.error(error);
7      return;
8    }
9
10     console.log(">>> File Content\n\b", data.toString());
11   })
12
```

- 1.3 Use the following code statement to read the file synchronously:

```
const { readFileSync } = require("fs");
```

A screenshot of a code editor with a dark theme. The editor has a tab labeled 'JS fileOperation.js'. The code is as follows:

```
demo8 > JS fileOperation.js > ...
13  const { readFileSync } = require("fs");
```

- 1.4 Write the blocking task that reads the **essay.md** file from the system using the **readFileSync** function and print the contents on the console

```
// Blocking Task
```

```
async function readFileContent() {
  const data = await readFileSync("./essay.md")
  console.log(">>> File Content\n\b", data.toString());
}
```

```
readFileContent()
```

```
13 const { readFileSync } = require("fs");
14
15 // Blocking Task
16 async function readFileContent() {
17   const data = await readFileSync("./essay.md")
18   console.log(">>> File Content\n\b", data.toString());
19 }
20
21 readFileContent()
22
```

- 1.5 Execute the following code in the terminal to view the output:

```
node fileOperations.js
```

```
demo8 % node fileOperation.js
>>> File Content
Mechanical Engineering Student Sample
In the first sample essay from mechanical engineering, what stands out immediately are the length and the photographs. I
n this case, the student was applying for an engineering scholarship, so he was given room to flesh out technical materi
al as well as address issues such as personal motivations one would expect to read in a personal statement.
demo8 %
```

Step 2: Write into the files using the fs.writeFile() method

2.1 Import the file system module in the **fileOperation.js** file:

```
const { writeFile } = require("fs");
```

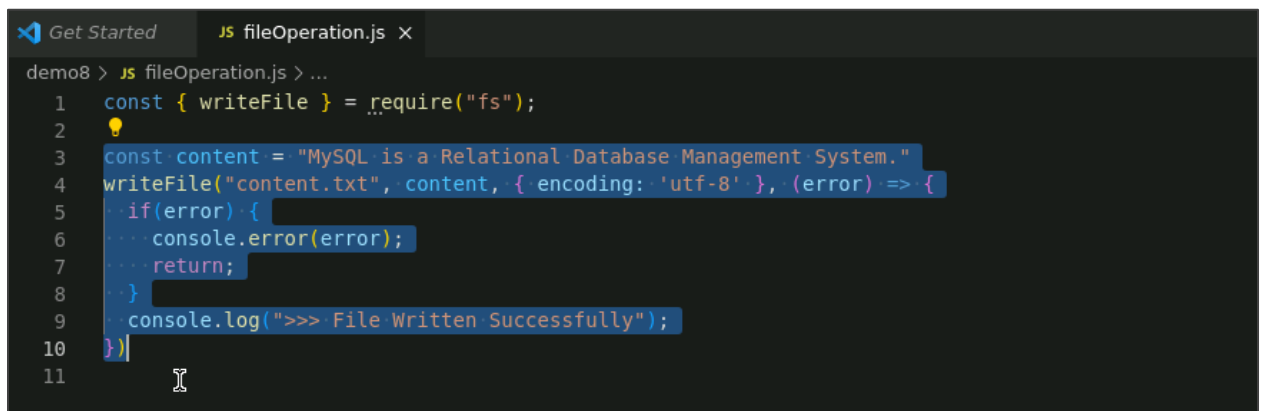


The screenshot shows a VS Code editor window with a tab labeled 'JS fileOperation.js'. The terminal at the bottom shows the command 'demo8 > JS fileOperation.js > ...'. The editor displays the following code on line 1:

```
1  const { writeFile } = require("fs");
```

2.2 Add the following code snippet to write a string into a file in async mode:

```
const content = "MySQL is a Relational Database Management System."
writeFile("content.txt", content, { encoding: 'utf-8' }, (error) => {
  if(error) {
    console.error(error);
    return;
  }
  console.log(">>> File Written Successfully");
})
```



The screenshot shows the same VS Code editor window with the complete code in fileOperation.js. The terminal shows the command 'demo8 > JS fileOperation.js > ...'. The editor displays the following code:

```
1  const { writeFile } = require("fs");
2
3  const content = "MySQL is a Relational Database Management System."
4  writeFile("content.txt", content, { encoding: 'utf-8' }, (error) => {
5    if(error) {
6      console.error(error);
7      return;
8    }
9    console.log(">>> File Written Successfully");
10  })
11
```

2.3 Import the file system module again to write into the a file in synchronous mode:

```
const { writeFileSync } = require("fs");
```

```
demo8 > js fileOperation.js > ...
1  const { writeFile } = require("fs");
2
3  const content = "MySQL is a Relational Database Management System."
4  writeFile("content.txt", content, { encoding: 'utf-8' }, (error) => {
5    if(error) {
6      console.error(error);
7      return;
8    }
9    console.log(">>> File Written Successfully");
10 } )
11
12 const { writeFileSync } = require("fs");
13
```

2.4 Add the following code snippet to write a string into a file in synchronous mode:

```
async function writeContentInFile() {
  try {
    const content = "MySQL is a Relational Database Management System."
    const data = await writeFileSync("content.txt", content)

    console.log(">>> File Written Successfully");
  } catch (error) {
    console.error(error);
  }
}

writeContentInFile();
```

```
demo8 > js fileOperation.js > ...
11
12 const { writeFileSync } = require("fs");
13
14 async function writeContentInFile() {
15   try {
16     const content = "MySQL is a Relational Database Management System."
17     const data = await writeFileSync("content.txt", content)
18
19     console.log(">>> File Written Successfully");
20   } catch (error) {
21     console.error(error);
22   }
23 }
24 writeContentInFile();
25
```

Step 3: Delete files using the unlink(path, callback) method

3.1 Import the file system module:

```
const { unlink } = require("fs");
```

```
demo8 > js fileOperation.js > ...  
1  const { unlink } = require("fs");|
```

3.2 Use the **unlink** function and pass the file as an input parameter for deletion in async mode:

```
const { unlink } = require("fs");
```

```
// Non-Blocking Task
```

```
unlink("./essay.md", (error) => {
```

```
  if (error) {
```

```
    console.error(error);
```

```
    return;
```

```
  }
```

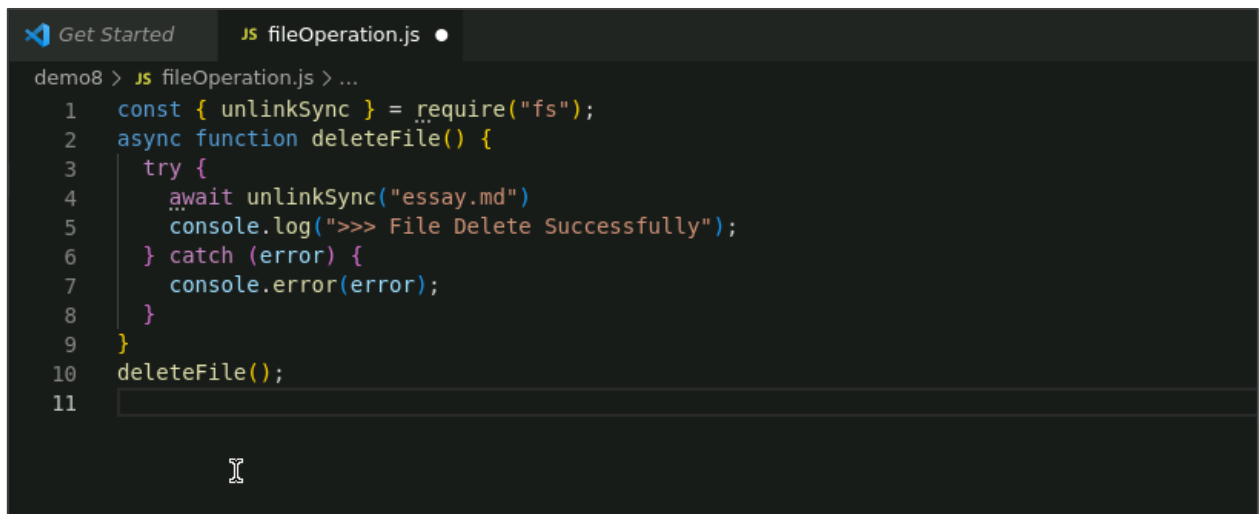
```
  console.log(">>> File Delete Successfully");
```

```
})
```

```
demo8 > js fileOperation.js > ...  
1  const { unlink } = require("fs");  
2    
3  const { unlink } = require("fs");  
4    
5  // Non-Blocking Task  
6  unlink("./essay.md", (error) => {  
7    if (error) {  
8      console.error(error);  
9      return;  
10   }  
11     
12   console.log(">>> File Delete Successfully");  
13 })  
14
```

3.3 Use the following code to delete the file synchronously:

```
const { unlinkSync } = require("fs");
async function deleteFile() {
  try {
    await unlinkSync("essay.md")
    console.log(">>> File Delete Successfully");
  } catch (error) {
    console.error(error);
  }
}
deleteFile();
```

A screenshot of a code editor interface. At the top, there are two tabs: 'Get Started' and 'JS fileOperation.js'. The 'JS fileOperation.js' tab is active. Below the tabs, the code from the previous block is shown, line-by-line, with line numbers 1 through 11 on the left. The code is: 1 const { unlinkSync } = require("fs"); 2 async function deleteFile() { 3 try { 4 await unlinkSync("essay.md") 5 console.log(">>> File Delete Successfully"); 6 } catch (error) { 7 console.error(error); 8 } 9 } 10 deleteFile(); 11 The cursor is at the end of line 11. The background is dark with light-colored text.

By following these steps, you have successfully executed file system operations like reading, writing, and deleting to manage files within a file system.