**Code:**

const fs = require("fs");

const path = require("path");

function readFile(filePath, callback) {

  fs.readFile(filePath, "utf-8", (err, data) => {

    if (err) {

      return callback(err);

    }

    callback(null, data);

  });

}

function writeFile(filePath, data, callback) {

  fs.writeFile(filePath, data, (err) => {

    if (err) {

      return callback(err);

    }

    callback(null, "File written successfully");

  });

}

function logMessage(message, callback) {

  setTimeout(() => {

    console.log(message);

    callback(null, "Message logged");

  }, 1000);

}

const inputFilePath = path.join(\_\_dirname, "text.txt");

const outputFilePath = path.join(\_\_dirname, "output.txt");

function processFiles() {

  readFile(inputFilePath, (err, data) => {

    if (err) {

      return console.error("Error reading file:", err);

    }

    console.log("File data read successfully:", data);

    writeFile(outputFilePath, data, (err) => {

      if (err) {

        return console.error("Error writing file:", err);

      }

      console.log("File written successfully.");

      logMessage("Process completed successfully!", (err, message) => {

        if (err) {

          return console.error("Error logging message:", err);

        }

        console.log(message);

      });

    });

  });

}

processFiles();

**Primary Functions:**

* readFile: Asynchronously reads a files content
* writeFile: Asynchronously delays writing a files content
* logMessage: Simulates logging a message, after a delay

**Control Flow:**

* processFiles() is the main function which reads the contents of the text.txt file first using readFile().
* When that has completed successfully, it writes to output.txt file using the writeFile() function.
* When that function has written the file, it will log a message saying it was successful using logMessage().

**Error Management:**

* Every function in the code checks for errors. When an error is found to have occurred, it logs the error message, and the process stops there.

**Output:**

