Multi-Threaded Web Server with IPC and Semaphores

SO 25/26

User Manual

Department of Electronics, Telecommunications and Informatics

Bernardo Mota Coelho - 125059
Tiago Francisco Crespo do Vale - 125913

9/12/2025

1. Download instructions

Option 1: Online and Public Repository

1. Open your terminal.
2. Clone the repository by typing:
   git clone https://github.com/tiagofcvale/SO_Project2
3. Navigate into the project directory:
   cd SO_Project2

Option 2: `.rar` or `.zip` Archive

1. Download the archive file.
2. Unzip the contents.
3. Navigate into the project directory using your terminal:
   cd SO_Project2

2. Compilation and Execution instructions

1. **Compile the code** by typing the following command in the terminal:
   make
2. **Start the server** with this command:
   ./server
3. **To stop the server**, press `CTRL+C` in the terminal session where the server is active. The server will perform a graceful shutdown, ensuring all Inter-Process Communication (IPC) resources are properly cleaned up.

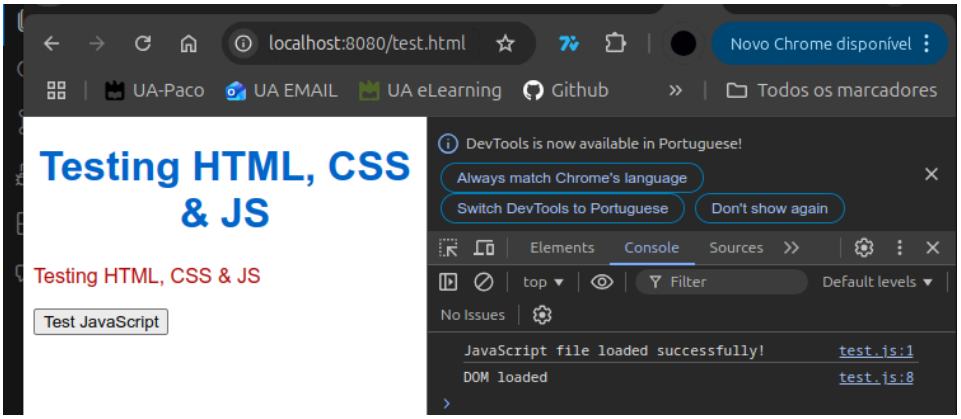
3. Examples

3.1 Test case 1

Objective: Ensure the server correctly handles GET requests for fundamental web resources (HTML, CSS, and JavaScript files).

- **Steps:**
  - Add HTML, CSS, and JavaScript files to the www folder.
  - Execute: make
  - Execute: ./server
  - Access: http://localhost:8080/name_of_your_html.html
  - Verify that all web components load and function correctly.
- **Expected Result:**
  - The files are successfully placed in the correct directory.
  - The URL loads and displays the complete web page correctly.
- **Actual Result:** A website successfully appears.
- **Status: Passed**

Screenshot:

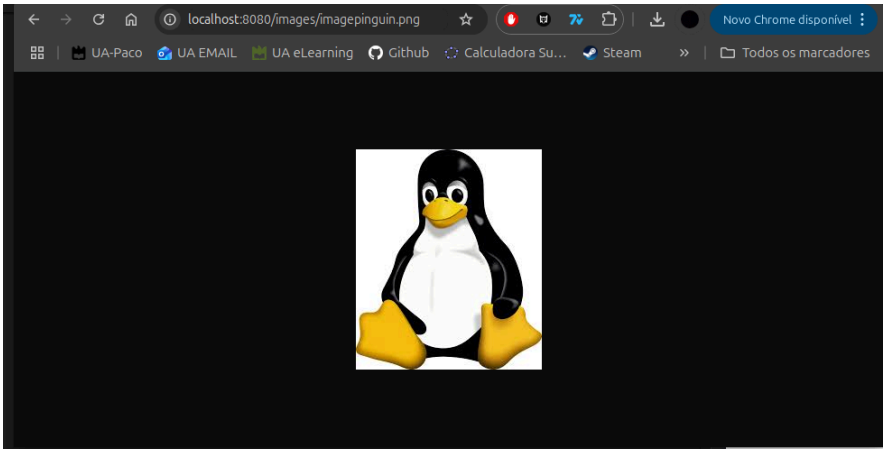


3.2 Test case 2

Objective: Verify the server's capability to serve image files via a GET request.

- **Steps:**
  - Add an image file to the www/images folder.
  - Execute: make
  - Execute: ./server
  - Access: http://localhost:8080/images/name_of_your_image
  - Verify that the image is displayed correctly.
- **Expected Result:**
  - The image file is successfully placed in the correct directory.
  - The URL correctly displays the image.
- **Actual Result:** An image successfully appears.
- **Status: Passed**

Screenshot:



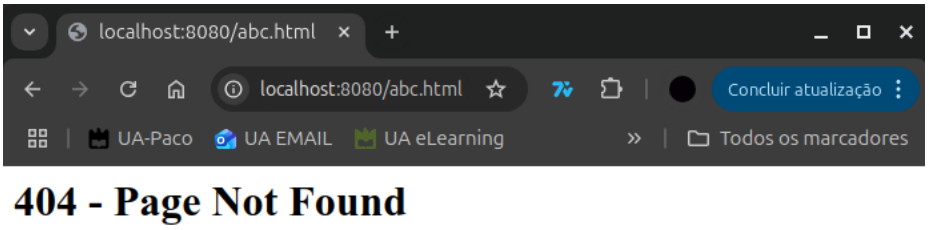
3.3 Test case 3

Objective: Confirm that the server returns the correct 404 status code and response

page for non-existent resources.

- **Steps:**
  - Execute: make
  - Execute: ./server
  - Access: http://localhost:8080/images/unknown_file
  - Verify that the page displays the message 404 - File Not Found.
- **Expected Result:**
  - The program detects a file not found error.
  - The URL displays the customized 404.html error page with the relevant message.
- **Actual Result:** An error appears and the server program continues to run.
- **Status: Passed**

Screenshot:

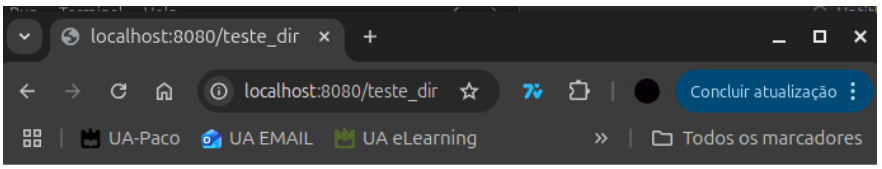


3.4 Test case 4

Objective: Confirm that the server returns the correct 403 status code and response page for forbidden or unauthorized resources.

- **Steps:**
  - Create a directory without adequate access permissions.
  - Execute: make
  - Execute: ./server
  - Access: http://localhost:8080/images/dir_without_perms
  - Verify that the page displays the message 403 - Access Denied.
- **Expected Result:**
  - The program detects an access denied error.
  - The URL displays the customized 403.html error page with the relevant message.
- **Actual Result:** An error appears and the server program continues to run.
- **Status: Passed**

Screenshot:



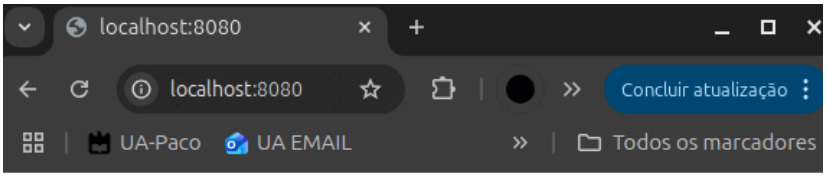
403 - Acess Denied

3.5 Test case 5

Objective: Ensure the server automatically serves the index.html file when a directory URL is accessed (e.g., the root path).

- **Steps:**
  - Execute: make
  - Execute: ./server
  - Access: http://localhost:8080
  - Verify that the index.html file is displayed.
- **Expected Result:**
  - The server successfully routes the root request to index.html.
  - The URL displays the content of index.html.
- **Actual Result:** index.html is shown when no specific file is requested.
- **Status: Passed**

Screenshot:



HELLO WORLD!

3.6 Test case 6

Objective: Confirm that the server correctly identifies and sets the Content-Type header for various file types.

- **Steps:**
    - Execute: make
    - Execute: ./server
    - Execute the terminal command: curl -I http://localhost:8080/your_file_name
    - Verify that the output headers include the correct file type.
- **Expected Result:**
    - The program accurately identifies the file type.
    - The server correctly handles various file types and sets the corresponding Content-Type in the headers.
- **Actual Result:** The program successfully handles and reports various file types.
- **Status: Passed**

Screenshot: