universidade de aveiro deti departamento de electrónica, telecomunicações e informática

# Multi-Threaded Web Server with IPC and Semaphores

## User Manual

Authors:
Alan Marques (125046)
Miguel Sousa (125624)

## Introduction

This project is a high-performance, multi-threaded web server written in C. It is designed to handle multiple concurrent client connections efficiently using a combination of:

- **Master-Worker Architecture**: A master process accepts connections and distributes them to worker processes.
- **Thread Pool**: Each worker process maintains a pool of threads to handle requests.
- **IPC (Inter-Process Communication)**: UNIX domain sockets are used to pass file descriptors between the master and workers.
- **Shared Memory & Semaphores**: Used for global statistics tracking and synchronization.
- **LRU Cache**: An in-memory cache to speed up serving frequently accessed static files.

## Prerequisites

This server is designed for **Linux** environments due to its reliance on POSIX semaphores and other system-specific features.

#### Requirements:

- **OS**: Linux
- **Compiler**: GCC (GNU Compiler Collection)
- **Build Tool**: Make

## Clean Build

To remove compiled object files and binaries:

```
make clean
```

## Configuration

The server is configured via the server.conf file located in the root directory. You can modify this file to tune performance.

**Default `server.conf` settings:**

```
PORT=8080 NUM_WORKERS=4 THREADS_PER_WORKER=8
DOCUMENT_ROOT=./www MAX_QUEUE_SIZE=100 LOG_FILE=server.log
CACHE_SIZE_MB=10 TIMEOUT_SECONDS=10
```

## Running the Server

In order to compile and start the server you should run these in your terminal, in the project directory:

```
make clean

make run
```

The server will start in the foreground. You should see output indicating the Master PID and that it is listening on the configured port.

## Usage

Once the server is running, you can interact with it using a web browser or command-line tools.

**Example 1: Fetch a page using "ab"** Open a new terminal window and run:

```
ab -n 1000 -c 100 http://localhost:8080/index.html
```

After this just insert http://localhost:8080 on your browser and you will be able to see the server up and running!

**Stopping the Server**

To stop the server, press Ctrl+C in the terminal where it is running. The server handles the interrupt signal to perform a graceful shutdown, cleaning up resources and shared memory.

## Testing

The project includes a comprehensive test suite that covers both unit tests (C) and integration/load tests (Bash).

### Run All Tests

To run the full test suite:

```
make test
```

This command will:

1. Compile and run **Concurrency Tests** (test_concurrent): Checks queue logic, cache integrity, and thread safety.
2. Compile and run **Load Tests** (test_load): Checks HTTP responses, error codes, and server stability under load.