Multi-threaded Web Scraping System

**Title:** Multi-threaded Web Scraping System  
**Course:** Operating Systems Project

**Group Members:**  
- 23K-2080  
- 23P-0658  
- 23K-2024  
- 23K-2037

# Table of Contents

1. Introduction  
2. Objective  
3. Tools and Technologies Used  
4. Project Overview  
5. Detailed Workflow and Dry Run  
6. Use Cases  
7. Synchronization and Multithreading Analysis  
8. Conclusion  
9. References

# 1. Introduction

The "Web Scraping System" project demonstrates how real-world data can be extracted from multiple web pages using advanced concepts of operating systems like process creation, multithreading, inter-process communication, and synchronization mechanisms.

**Solutions Provided:**

* Solves the **Producer-Consumer Problem** by managing a producer thread (reading URLs) and multiple consumer threads (scraping).
* Handles **Race Conditions** through **mutex locks** ensuring data consistency.
* Addresses aspects of the **Reader-Writer Problem** where multiple threads read/write shared files carefully.
* Implements **Parallelism** through multithreading to improve performance and efficiency.
* Manages **Synchronization** between threads using **semaphores**.

This project is a complete implementation of how critical OS problems are solved through coordinated process/thread management, ensuring data safety, process efficiency, and system stability.

# 2. Objective

* Implement a user authentication system using inter-process communication.
* Solve synchronization problems like Producer-Consumer and Race Conditions.
* Perform concurrent web scraping on multiple URLs.
* Store and manage scraped data efficiently into CSV files.
* Allow flexible data filtering and analysis through a user-friendly menu.

# 3. Tools and Technologies Used

* **Programming Language:** C
* **Libraries and Headers:**
  + <pthread.h> — for multithreading.
  + <curl/curl.h> — For HTTP web requests.
  + <libxml/HTMLparser.h> and <libxml/xpath.h> — For HTML parsing and XPath queries.
  + <stdio.h>, <stdlib.h>, <string.h> — for standard input/output and memory operations.
  + <sys/types.h>, <sys/wait.h>, <unistd.h> — For process management (fork, pipe, wait).
  + <termios.h> — for secure password input without echo.
  + <semaphore.h> — for synchronization between threads.
* **OS Environment:** Linux (Ubuntu preferred)
* **Compiler:** GCC
* **IDE:** Visual Studio Code
* **External Resources:**
  + Website: <https://scrapeme.live/product-category/pokemon/>

# 4. Project Overview

The project consists of the following phases:

* **User Authentication:** Parent and child processes communicate through a pipe to validate login credentials.
* **Producer-Consumer Model:**
  + Producer thread reads URLs and XPath expressions.
  + Consumer threads fetch and parse HTML pages concurrently.
* **Data Storage:** Scraped product information is saved into separate CSV files.
* **User Interaction:** User can filter products based on price, find lowest/highest priced products, and display all products using a menu system.

# 5. Detailed Workflow and Dry Run

**Step 1: Program Initialization**

* Clear screen.
* Create IPC pipe.
* Fork child process.

**Step 2: Login Authentication**

* Child process asks for username and password.
* If login fails after 4 attempts, the parent process exits.
* Otherwise, login is successful, and the program proceeds.

**Step 3: Thread Setup**

* Initialize CURL library and semaphores.
* Create **one Producer thread** to read from urls1.txt and store scraping data.
* Create **four Consumer threads** to perform parallel web scraping.

**Use of Producer-Consumer Model:**

* Producer fills the task list.
* Consumers process the tasks concurrently.

**Use of Mutexes:**

* Protect shared variables like urls\_to\_process, threadDataIndex, and urls\_processed from race conditions.

**Use of Semaphores:**

* Coordinate the availability of URLs between producer and consumers.

**Step 4: Web Scraping and CSV Storage**

* Each consumer thread fetches an HTML page.
* Parses product names and prices using XPath.
* Saves the extracted data into a separate CSV file.

**Step 5: User Menu Interaction**

* After scraping, a menu is displayed:
  1. Display products within a user-defined price range.
  2. Show the product with the lowest price.
  3. Show the product with the highest price.
  4. Display all scraped products.
* Each operation is implemented using new threads for efficient parallel processing.

**Step 6: Termination**

* After the user exits, CURL resources and semaphores are cleaned up.
* A thank you message is displayed with group member IDs.

**Sample Output Reports:**  
  
**1. Price Filtering Output (≤£15)**  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 file name: scrapeme1.csv is opening  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Product: Bulbasaur, Price: 12.50  
Product: Charmander, Price: 14.99  
Product: Squirtle, Price: 13.75  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 file name: scrapeme2.csv is opening  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Product: Pikachu, Price: 15.00  
...  
  
**2. Lowest Price Output**  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 LOWEST PRODUCT  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Product: Diglett, Price: 8.99 **3. Highest Price Output**  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 HIGHEST PRODUCT  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Product: Mewtwo, Price: 45.50  
  
**4. Complete Display Output**  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 WEBSITE NAME : scrapeme1.csv  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
PRODUCT NAME PRODUCT PRICE  
Product: Bulbasaur, Price: £12.50  
Product: Charmander, Price: £14.99

# 6. Use Cases

1. Pokemon Price Comparison Tool

* Find cheapest Pokemon.

2. Pokemon Inventory Monitoring

* Monitor competitor prices.

3. Pokemon Market Research

* Analyze market pricing trends.

# 7. Synchronization and Multithreading Analysis

**Multithreading Efficiency:**- Sequential scraping: ~6 seconds  
- Parallel scraping: ~1.5 seconds  
- Speedup: ~4x  
  
**Synchronization Techniques:**  
- Mutex Locks: Protect shared indexes.  
- Semaphores: Coordinate Producer-Consumer operations.  
- Thread-safe file writes.  
**Error Handling:**  
- CURL errors handled.  
- Missing elements safely skipped.  
- File open errors handled.

**Data Consistency**

* Synchronization mechanisms ensured accurate results
* No duplicate or missing data points observed

# 8. Conclusion

This project successfully demonstrates the use of Operating System principles like synchronization, multithreading, inter-process communication, and process management. It shows how classic OS problems like the Producer-Consumer problem, Race Conditions, and aspects of the Reader-Writer Problem can be solved practically.

The use of multithreading improves program speed by parallelizing web scraping tasks, while mutexes and semaphores ensure data integrity and proper coordination between threads.

# 9. References

- Scrapeme Live: <https://scrapeme.live/product-category/pokemon/>  
- CURL Documentation: <https://curl.se/libcurl/>  
- libxml2 Documentation: <http://xmlsoft.org/>











