Multithreaded Web Crawler in C (Due date: 4/25/2024)

Project Overview

In this project, you will develop a multithreaded web crawler in C that can efficiently fetch, parse, and process web pages from the internet. The goal is to utilize multiple threads to improve the performance and efficiency of the web crawling process.

Problem Statement

The internet is vast and ever-growing. Traditional single-threaded web crawlers are not efficient enough to handle the volume of information available. The challenge is to develop a web crawler that can navigate the web efficiently by fetching multiple pages in parallel, thus significantly reducing the time required to crawl a set of URLs.

Functionality Description

Your web crawler should include the following functionalities:

- **Multithreading:** Use multiple threads to fetch web pages concurrently.
- **URL Queue:** Implement a thread-safe queue to manage URLs that are pending to be fetched.
- **HTML Parsing:** Extract links from the fetched web pages to find new URLs to crawl.
- **Depth Control:** Allow the crawler to limit the depth of the crawl to prevent infinite recursion.
- **Synchronization:** Implement synchronization mechanisms to manage access to shared resources among threads.
- **Error Handling:** Handle possible errors gracefully, including network errors, parsing errors, and dead links.
- **Logging:** Log the crawler's activity, including fetched URLs and encountered errors.

Features Documentation

1. Thread Management

- Create and manage worker threads that fetch and process web pages in parallel.
- Implement a mechanism for terminating threads gracefully once their task is complete.

2. URL Queue

- Design a thread-safe queue that holds URLs to be crawled.
- Ensure that multiple threads can add to and fetch from the queue without causing data corruption.

3. HTML Parsing

- Parse HTML content to extract links.

- Use a library or write a custom parser to identify and follow links within HTML documents.

4. Depth Control

- Implement depth control to limit how deep the crawler goes into a website.
- The user should be able to specify the maximum depth as a parameter.

5. Synchronization

- Use mutexes, semaphores, or other synchronization primitives to ensure that shared resources like the URL queue are accessed safely.

6. Error Handling

- Implement error handling to manage network failures, invalid URLs, and other exceptions.
- Ensure that the crawler can recover from errors and continue operating.

7. Logging

- Log the progress of the web crawler, including which URLs have been visited and any errors encountered.
- Optionally, implement a verbosity level for the logging system.

Project Deliverables

- **Source Code:** All project source code, including a Makefile for building the project.
- **Documentation:** A detailed project report documenting your design decisions, challenges faced, and how they were overcome.

Submission Guidelines

What to Submit

Your group is required to submit a single zip file containing the following items: 1. **README.md** - A markdown file that must include: - The names and RUID numbers of all group members. - A comprehensive description of your implementation, focusing on its architecture, the multithreading approach, the specific roles, and contributions of each group member, and any libraries used. This should serve as a detailed overview of your web crawler, akin to a manpage.

2. **crawler.c** - The source code file(s) for your multithreaded web crawler.

3. **Makefile** - A makefile to facilitate the building of your web crawler's binary with appropriate commands.

Optional

1. Submit your GitHub (or any Git) repository link if you have been using Git version control for your project. This is not mandatory, but it is highly recommended. Make sure that your repository is public and accessible to the TA.

Compilation Requirements

Ensure your project is compilable on Linux using the `gcc` compiler, conforming at least to the ISO C11 standard. The following command should be used to compile your project, ensuring compatibility with the required flags and standards:

```
gcc -std=c11 -pedantic -pthread crawler.c -o crawler
```

Note 1: The inclusion of the `-pthread` flag is crucial for compiling programs that employ pthreads for multithreading. **Note 2**: You can use the C POSIX library for developing your multithreaded web crawler.

Makefile Targets

Your Makefile must support the following targets:

- **all**: Compiles the web crawler binary.
- **clean**: Removes the web crawler binary and any other files generated during compilation.
- **run**: Executes the web crawler binary. This target may use default arguments for demonstration purposes, but it should allow for flexible configuration to accommodate various scenarios.

Good luck, and happy coding!