OS Report

Introduction:

This code represents a basic implementation of a multithreaded web crawler in C, designed to traverse web pages starting from a given URL, fetch their HTML content using libcurl, and extract hyperlinks embedded in the HTML. The crawler employs a thread-safe queue data structure to manage the URLs to be processed and utilizes POSIX threads for concurrent URL fetching and processing.

Code:

1. Header Files:

A computer code with green text

Description automatically generated

* 1. stdio.h, stdlib.h, pthread.h, string.h and time.h are standard C library headers for input / output, memory location, threading, string operation and time function.
  2. curl/curl.h is header files for libcurl, it is used for transferring data with URL syntax.
  3. Libxml/HTMLparser.h is header file for libxml2, it is a library for parsing XML and HTML documents.

1. Data Structures:

A screenshot of a computer code

Description automatically generated

* 1. URLQueueNode is a structure representing a node in a queue. It contains:
     1. char \*url: A pointer to a string storing the URL.
     2. int depth: An integer showing the depth of the URL in the crawling process. ie how many links far away are we from starting point.
     3. URLQueueNode \*next: A pointer for the next node in the queue.
  2. URLQueue is a structure representing a thread-safe queue of URLs.
     1. URLQueueNode \*head: A pointer for the first node in the queue.
     2. URLQueueNode \*tail: A pointer for the last node in the queue.
     3. Pthread\_mutex\_t lock: a lock to ensure thread safety when we are accessing the queue.

1. Queue Operations:

A screenshot of a computer program

Description automatically generated

* 1. initQueue is a function that initializes a URL queue.
     1. It sets both the head and tail pointer to the queue to NULL.
     2. It initializes the mutex lock associated with queue using pthread\_mutex\_init() to ensure thread safety.
  2. Enqueue is a function that adds URL to the queue along with its depth information.
     1. It locks the queue's mutex to ensure thread safety during modification of the queue.
     2. It checks if the queue is empty:
        1. If it's not empty, then it links the new node to the end of the queue
        2. Else it sets both head and tail pointer to new node.
     3. Then it updates tail pointer to point to the newly added node and unlocks the mutex.
  3. Dequeue is a function that removes and return the URL from the front of the queue.
     1. It locks the queue's mutex to ensure thread safety.
     2. It checks if the queue is empty, If it is empty then it unlocks the mutex and returns NULL.
     3. It retrieves the URL and depth from the first node in the queue.
     4. It updates the head pointer to point to the next node in the queue. If the queue becomes empty after dequeuing then it updates the tail pointer accordingly.

1. Chunk Function for libcurl:

A screenshot of a computer code

Description automatically generated

* 1. write\_chunk function is designed to handle data received in chunks during an HTTP transfer, commonly used in conjunction with libcurl.
  2. It takes four parameters: data, size, nmemb and userdata.
  3. Inside the function, it calculates the total size of the received data chunk by multiplying size and nmemb.
  4. The function reallocates memory for the response buffer to accommodate the new data chunk.
  5. If memory reallocation is successful, it copies the received data chunk into the response buffer using memcy.

1. Function for Fetching and Parsing HTML:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer code

Description automatically generated

* 1. Extract\_html is function is designed to fetch HTML content from a given URL using libcurl and return it as a dynamically allocated C-style string.
     1. It starts by initializing a libcurl easy handle and checking if initialization was successful.
     2. If curl\_easy\_init returns NULL then it prints an error message.
     3. It initializes a response structure, which likely contains a buffer for storing the received data and its size.
     4. it performs the HTTP request using curl\_easy\_perform. If the request fails it prints an error message.
     5. If there is a successful completion of the request, the function allocates memory for a new buffer to hold the HTML content received.
  2. Parse\_html is designed to extract URLs from HTML content.
     1. It initializes an HTML parser context htmlParserCtxPtr using htmlCreateMemotyParserCtxt which is responsible for parsing HTML content stored in memory. If the initialization fails, indicating a problem with the HTML parser, it prints an error message.
     2. it reads the HTML content into an HTML document htmlDocPtr using htmlCtxtReadMemory. This function call parses the HTML content using the parser context created earlier. If parsing the HTML content fails, it prints an error.
     3. After successfully parsing the HTML content into a document, the function retrieves the root element of the document using xmlDocGetRootElement. If the root element is NULL which means an empty HTML document, it prints an error message.
     4. If the HTML document is not empty then the function traverses the HTML Document Object Model to find anchor tags using a loop. For each element found, it checks if it's an anchor tag and if its name is "a". If both conditions are met then it extracts the value of the href attribute using xmlgetProp.
     5. If a valid href attribute value is found it adds the link to the URL queue.

1. Logging Function:

A computer code with text

Description automatically generated

* 1. log\_message is designed to facilitate logging messages to a file named "crawler.log".
     1. It attempts to open the file in append mode using fopen.
     2. If the file pointer logfile is not NULL then the function proceeds to log the message.
     3. It fetches the current time using the time function. It returns the current calendar time in seconds since the Epoch. This time value is then converted to year, month, day, hour using localtime function.
     4. The function formats and writes the log message to the file.
     5. The file is closed using fclose. If the file opening operation fails, meaning logfile is NULL and prints an error message.

1. Cleanup Function:

A screen shot of a computer program

Description automatically generated

* 1. Cleanup is designed for releasing resources associated with the URL queue and ensuring proper program termination.
     1. It starts by initializing two pointers current and next to traverse the linked list of URL queue nodes starting from the head of the queue. While loop iterates over each node in the queue until reaching the end.
     2. During each iteration, it frees the memory allocated for the URL stored in the current node using free function.
     3. Before moving to the next node in the queue, it stores a reference to the next node next = current->next to avoid losing track of the remaining nodes.
     4. Once all nodes in the queue have been processed and freed, the function proceeds to destroy the mutex associated with the URL queue using pthread\_mutex\_destroy function.

1. Thread Function:

A screenshot of a computer code

Description automatically generated

* 1. \*fetch\_url serves as a placeholder for fetching and processing URLs concurrently in a threaded environment.
     1. It takes a void pointer arg to be cast to a pointer to a URL Queue structure representing the queue of URLs to be processed.
     2. The function then enters a loop where it repeatedly dequeues URLs from the provided URL queue using dequeue function.
     3. For each URL dequeued, the function proceeds to fetch the corresponding HTML content using the fetch\_html function.
     4. If the HTML content is successfully fetched ie it is not NULL the function then parses the HTML content to extract links using the parse\_html function.
     5. After parsing, the memory allocated for the HTML content is freed to prevent memory leaks using free function.

1. Main Function:

A screen shot of a computer code

Description automatically generated

* 1. main function serves as the entry point of the program for the execution of the web crawler. It starts by checking the number of command-line arguments.
     1. If valid starting URL is provided as a command-line argument, the function proceeds to initialize a URL queue. This queue will hold the URLs to be crawled.
     2. The program defines the number of threads to be used for concurrent URL processing.
     3. The program then enters a loop to create threads for fetching and processing URLs concurrently.
     4. After creating all threads, the program enters another loop to wait for each thread to finish its execution. This ensures that the main thread waits for all worker threads to complete their tasks before proceeding.
     5. Once all threads have finished executing, the program calls the cleanup function to release any resources associated with the URL queue.