COP701 A3: Mini-Web Browser

September 30, 2024

1 Problem statement

This assignment aims to design a mini web browser to render simple HTML web pages. The system should support multi-threading and multi-processing for handling multiple tabs, allowing one to open multiple search windows simultaneously.

In this assignment, you will fetch simple HTML pages (no JavaScript) and write your own HTML parser to convert HTML documents into a Document Object Model (DOM), essentially an abstract syntax tree (AST). Later, traverse the AST nodes to deploy a rendering (GUI) using Qt widgets. Use any standard packages like *libcurl* to fetch HTML pages.

1.1 Fetching HTML Pages

Design a network handling process to fetch simple HTML pages using HTTP requests and cache these pages locally in the host machine. The mini-browser will support only restricted HTML tags, which are as follows: *HTML, head, title, body, nav, ul, li, header, h1, h2, h3, h4, h5, p, section, article, aside, footer, img, a, strong, em, u, small, blockquote, pre, code, ol.*

The assignment handout includes *gen_html.py* to generate random HTML pages with the above-mentioned tags. Upon running python *gen_html.py*, the pages will get generated in a directory named *html_dataset*. Launch these pages on an HTTP server using *python -m http.server* command. This command should start a server on *http://localhost:8000/*.

Use this script file, which is designed for your convenience, to create datasets to test the web browser implementation. Use any standard package such as *libcurl* to fetch the launched pages.

1.2 DOM Conversion

Develop a process to parse the HTML page into a Document Object Model (DOM). DOM is a language-independent interface that treats an HTML document as a tree structure (AST). Write the parser from scratch. Do not use any existing libraries to perform this task directly.

A rendering engine will later use this intermediate DOM structure to render the GUI. Use the Qt framework to render the GUI. Direct use of QWebEngineView or QWebView is **not allowed**; Use built-in Qt widgets to render DOM nodes such as text, images, and links.

1.3 Rendering

Write a rendering process using Qt Widgets to render the DOM structure generated in the previous step. Use the following links for your reference:

- Qt Installation Guide
- Refer Qt Widgets Tutorial to understand how to render AST nodes.

Specifically, take a look into QPainter. A example code block to render text is show below.

```
// Class for text-based HTML elements (e.g., <p>, <h1>)
class HTMLTextElement : public HTMLElement {
public:
    QString text;
    void render(QPainter& painter, int& yPosition) override {
        painter.drawText(10, yPosition, text);
        // Place the text at the given coordinates
        yPosition += 20;  // Move down for the next element
    }
};
```

1.4 Multi-Process and Multi-Threaded Support

Design the Mini-browser with multiple processes and multiple threads capability. A separate system process must handle the fetching, parsing and rendering tasks. The system should be able to handle multiple tabs parallelly. To coordinate between processes, use POSIX APIs, standard IPC mechanisms, or any open-source framework that supports multi-processing. Additionally, enable multi-threading using libraries like pthreads, std::threads, or Qthread.

1.5 Bonus

Implementing one or more of the following features will fetch extra credits.

- Maintain a browser history such that when the user reopens the web engine, it displays the most recently or most frequently visited web page.
- Allow the user to move forward or backwards through browser history.
- A option to save the browser session so that the user can restore tabs on accessing the web engine again.

2 Logistics

- The **deadline** for this assignment is **18/11/2024 at 11:59 PM**. It is a hard deadline and will not be extended.
- This assignment can be done in a group of two people. Only one of you must submit. (40 Marks)
- You can only use C/C++ in this assignment. You are not supposed to use any other programming language.
- You need to create a private git repository either on https://git.iitd.ac.in or github. Git commit history will be checked during evaluation.
- ANY form of **plagiarism** will not be tolerated.
- Submission will be made on Moodle. You need to submit all your code and a pdf format report. Compress all these in a tar file with the name $< entry\_number1 > \_ < entry\_number2 >$ *.tar* and upload on Moodle.
- You will be graded on the output of your code, the coding style and your viva/presentation.
- Any doubts regarding the course/assignment should be asked on Piazza.

2.1 Marks distribution

Report	5%
Regular git commits (from both the members)	5%
Coding style	10%
Parser	20%
Rendering Engine	35%
Multi-Process and Multi-Threaded	25%