HTML CSS Selector 实验报告

姓名：王松宸　学号：2024201594

2025 年 12 月 8 日

1　系统概述与功能亮点

1.1　简介

我设计并实现了高性能、功能完备的 HTML 解析器和 CSS 选择器，能够将非结构化的 HTML 文本解析为结构化的树，并提供 CSS 选择器和 XPath 两种强大的查询接口，支持对网页内容的精准定位、提取与修改。

1.2　功能亮点

程序在完成所有基本功能的同时，**完整实现了实验规定的所有选做内容**，功能特性如下：

- **HTML 解析**：支持标准 HTML 标签、属性解析，能够正确处理自闭合标签及文本结点，构建完整的树。
- **双路径查询**：
  1. **CSS 选择器**：
     - **基础支持**：标签、ID、类、通配符选择器。
     - **组合支持**：后代（空格）、分组（逗号）、相邻兄弟（`+`）、通用兄弟（`~`）、子元素（`>`）。
     - **高级伪类与伪元素**：实现了 `:empty`、`:first-child`、`::first-letter`、`::first-line`、`::before`、`::after` 六个选做的功能。
  2. **XPath 查询**：
     - 实现了 XPath 核心语法，支持绝对路径与相对路径。
     - 支持属性谓语过滤（如 `[@class='news']`）及索引定位（如 `//li[1]`）。
     - 支持 `text()` 函数直接提取文本结点。

1.3　输入与输出内容

- **输入**：HTML 源文件路径及用户交互式的查询指令。
- **输出**：结果结点列表、InnerText 文本提取结果、OuterHTML 源码结果、A 标签所有链接。

2　程序架构

2.1　解析模块

该模块负责将非结构化的 HTML 源码转换为一棵完整的树，主要包含预处理和解析两个阶段：

- **预处理**：读取 HTML 文件，去除无关字符，处理编码问题。

- **解析**：采用基于栈的单遍扫描算法，能够正确处理标签的嵌套关系、自闭合标签（如 `<img>`）以及文本，将各部分正确地存储在结点内。同时具备一定的容错能力，能够忽略注释与 DOCTYPE 声明。

2.2 查询模块

该模块实现了基础功能和选做功能，包含两个独立的查询引擎：

- **CSS 选择器引擎**：支持从基础选择器到复杂组合器（后代、兄弟、子元素）的解析与执行，并集成了伪类/伪元素的高级逻辑。
- **XPath 引擎**：支持路径导航（`/` 与 `//`）、属性谓语过滤以及索引定位。

2.3 结果处理模块

该模块负责对查询模块返回的内部结点列表进行进一步的处理。

- **再查询**：支持对结果列表里的特定结点进行进一步的 CSS / XPath 查询。
- **文本提取**：实现了 `innerText`，能够提取特定结点下的所有文本内容。
- **源码提取**：实现了 `outerHTML`，能够输出特定结点的完整 HTML 源码。
- **链接提取**：针对 `<a>` 标签，能够批量提取所有链接地址。

3 核心设计

3.1 数据结构定义

```
struct Node
{
    NodeType type;              // 结点类型
    char *tag_name;             // 标签名：div, p, a等 (仅元素结点)
    char *id;                   // id属性
    char **classes;             // class列表 (字符串数组)
    int class_count;            // class数量
    char *href;                 // href属性 (仅a标签)
    char *text;                 // 文本内容 (仅文本结点)
    struct Node *parent;        // 父结点指针
    struct Node *children;      // 第一个子结点指针 (左孩子)
    struct Node *next_sibling; // 下一个兄弟结点指针 (右兄弟)
};
```

3.2 关键算法实现

3.2.1 基于栈的 HTML 解析算法

采用一次遍历策略，利用栈结构维护标签嵌套关系：

- 遇到开始标签入栈，遇到结束标签出栈。
- 遇到自闭合标签（如 `img`, `meta`）直接挂载到当前栈顶元素下，不入栈。
- 解析器能够自动忽略 HTML 中的注释、DOCTYPE 声明，并保留必要的空白文本结点以维持排版格式。

3.2.2 CSS 选择器设计

CSS 查询引擎采用“分词-过滤”的结构：

- **解析阶段**：将复杂选择器（如 `div#main > p.active`）分解为原子选择器序列和组合符序列。
- **执行阶段**：维护一个“当前候选结点集合”。每处理一个组合符（如 `>`），就基于当前集合中的结点，查找其符合条件的子结点，生成新的候选集合。
- **高级特性实现**：
  - **兄弟选择器 (`+` / `~`)**：在遍历 `next_sibling` 链表时，增加了智能跳过 `TEXT_NODE`（空白换行符）的逻辑，确保准确匹配下一个元素结点。
  - **伪元素 (`::first-line`)**：实现了文本自动截断算法，能够智能识别并提取首行文本，忽略前导空白。

3.2.3 XPath 递归下降搜索

XPath 引擎基于递归思想实现：

- **路径解析**：将完整路径解析为路径段和谓语条件，比如对于 `//div[@id='a']`，路径段为 `div`，谓语条件为 `id='a'`。
- **递归搜索**：
  - 若路径以 `/` 开头，仅在当前上下文的直接子结点中匹配。
  - 若路径以 `//` 开头，则遍历当前上下文的整棵子树（深度优先）。
- **索引支持**：在匹配过程中维护计数器，支持 `[n]` 语法精确定位同级元素中的第 n 个。

4 调试分析

4.1 HTML 源码中空白结点的干扰

- **问题**:HTML 源码格式化产生的换行符和缩进会被解析为 `TEXT_NODE`。这导致 `node->next_sibling` 往往是空白文本而非下一个标签，直接破坏了 `+` 选择器和 `:first-child` 的判定逻辑。
- **解决**:在底层遍历逻辑中封装了 `get_next_element_sibling()` 和 `get_first_element_child()` 辅助函数，显式跳过非 `ELEMENT_NODE` 类型的结点。

4.2 伪元素和伪类相关功能的实现

- **问题**：扩展的六个功能完全无法套用原先的处理逻辑
- **解决**：解析命令时补充了对伪类和伪元素的操作，并在 `match_part` 和 `traverse_and_match` 函数中都进行了特判处理，保证所有功能都得到具体实现。

5 功能测试

5.1 测试结论

经过对标准网页及复杂新闻页面的全面测试，我们可以得出结论：

- 程序解析功能稳定，无内存泄漏。
- CSS 选择器支持所有基础及选做的高级语法。
- XPath 选择器能够准确执行路径查找和属性过滤。

5.2 功能测试报告

详细测试报告请参阅 功能测试报告**.pdf**。

5.3 使用手册

详细的操作指南请参阅 使用手册**.pdf**。

6 附录

- `lab3.cpp`
- 功能测试报告.pdf
- 使用手册.pdf