苏 州 大 学 实 验 报 告

院系	计算机学院	年级专业	21 计科	姓名	方浩楠	学号	2127405048
课程名称	编译原理课程实践					成绩	
指导教师	王中卿	同组实验者	无		实验日期	2023.9.22	

实 验 名 称 实验二

一. 实验目的

将给定的 LaTex 文件（example.tex）转换为 PDF 文件

二. 实验内容

- 基于正则表达式将 LaTex 源文件进行解析
- 将解析结果构建为 HTML 字符串
- 通过 pyfPDF 生成 pdf

需完成以下标记的解析如下标记：

- document
- title
- abstract
- section
- subsection
- itemize/item
- tabular
- emph
- textbf

三. 实验步骤和结果

最终运行结果:

运行结果文件位置:实验 2/example/example.pdf

How to Structure a Latex Document

In this article, I shall discuss some of the fundamental topics in producing a structured document. This document itself does not go into much depth, but is instead the output of an example of how to implement structure. Its Latex source, when in used with my tutorial provides all the relevant information.

Introduction

This small document is designed to illustrate how easy it is to create a well structured document within \LaTeX\cite{lamport94}. You should quickly be able to see how the article looks very professional, despite the content being far from academic. Titles, section headings, justified text, text formatting etc., is all there, and you would be surprised when you see just how little markup was required to get this output.

Top Matter

The first thing you normally have is a title of the document, as well as information about the author and date of publication. In \LaTeX{} terms, this is all generally referred to as \emph{top matter}.

Sectioning Commands

The commands for inserting sections are fairly intuitive. Of course, certain commands are appropriate to different document classes. For example, a book has chapters but a article doesn't.

- \texttt{\textbackslash title\{\emph{title}\}} - The title of the article.
- \texttt{\textbackslash date} - The date. Use:
- \texttt{\textbackslash date\{\textbackslash today\}} - to get the
- \texttt{\textbackslash date\{\emph{date}\}} - for a %\emph{} emphasises the specified text. Italics by default.
- \texttt{\textbackslash date\{\}} - for no date.
- \texttt{\textbackslash author} - The author of the document.
- \texttt{\textbackslash address} - The author's address. Use
- \texttt{\textbackslash thanks} - Where you put any acknowledgments.
- \texttt{\textbackslash email} - The author's email address.
- \texttt{\textbackslash urladdr} - The URL for the author's web page.

Command	Level
\part{part}	-1
\chapter{chapter}	0
\section{section}	1
\subsection{subsection}	2
\subsubsection{subsubsection}	3
\paragraph{paragraph}	4
\subparagraph{subparagraph}	5

top matter title date part chapter section subsection subsubsection paragraph subparagraph \LaTeX: A Document Preparation System

函数 re_find 伪代码:

Algorithm 1: Function to Find First Match

```
Data: Pattern p, Text text
Result: First match or empty string
Function re_find(p, text):
    m ← p.findall(text)
    if length of m = 1 then
        return m[0]
    end
    else
        return empty string ;    // Return empty string if no
         match or more than one match
    end
```

函数 re_find 的作用:
此函数接受两个参数：p，是已编译的正则表达式模式，text，是要查找匹配项的输入文本。
它使用正则表达式模式 p 的 findall 方法，在输入文本中查找所有非重叠的匹配项。
如果找到恰好一个匹配项，它将该匹配项作为字符串返回。如果没有找到匹配项或找到多个匹配项，它将返回一个空字符串。

函数 re_findall 伪代码:

Algorithm 2: Function to Find All Matches

```
Data: Pattern p, Text text
Result: List of all matches
Function re_findall(p, text):
    return p.findall(text)
```

函数 re_findall 作用:
此函数同样接受两个参数：p，是已编译的正则表达式模式，text，是要查找匹配项的输入文本。
它使用正则表达式模式 p 的 findall 方法，在输入文本中查找所有非重叠的匹配项。
它返回一个包含所有在输入文本中找到的匹配项的列表

函数 clear_text 伪代码

Algorithm 3: Function to Clear Text

Data: Text *text*

Result: Cleaned text

```
Function clear_text(text):
    lines ← empty list
    for each line in text do
        line ← strip(line)
        if length of line > 0 then
            lines.append(line)
        end
    end
    return join(lines, ' ')
```

函数 clear_text 作用:
此函数接受一个参数 text，即要处理的输入文本。
它使用换行符 ('\n') 将输入文本拆分成多行，然后去除每行的前导和尾随空格，将非空行添加到名为 lines 的列表中。
最后，它将清理后的行连接成一个用空格分隔的单一字符串，并返回该清理后的文本。

提取 latex 文档的各个标签

1. **提取文档内容:**

   使用以下正则表达式来提取整个文档的内容，即位于`\begin{document}`和`\end{document}`之间的部分:

   ```
   p_doc = re.compile(r'\\begin{document}(.+?)\\end{document}', re.S)
   document = re_find(p_doc, content)
   ```

   这个正则表达式将匹配`\\begin{document}`和`\\end{document}`之间的所有内容（`re.S`标志使`.`匹配换行符）。然后，`re_find`函数用于获取匹配的内容。

2. **提取标题:**

   使用以下正则表达式来提取文档中的标题，即位于`\title{}`标签内的内容:

   ```
   p_title = re.compile(r'\\title{(.+?)}', re.S)
   title = re_find(p_title, document)
   ```

3. **提取摘要:**

   使用以下正则表达式来提取文档中的摘要，即位于`\begin{abstract}`和`\end{abstract}`之间的内容:

   ```
   p_abs = re.compile(r'\\begin{abstract}(.+?)\\end{abstract}', re.S)
   abstract = re_find(p_abs, document)
   abstract = clear_text(abstract)
   ```

   提取的摘要内容经过`clear_text`函数清理以去除多余的空格和换行符。

4. **提取章节和子章节：**

   使用以下正则表达式来提取章节的标题和内容，以及子章节的标题和内容：

```
p_sec = re.compile(r'\\section{(.+?)}(.+?)\\section', re.S)  # 只匹配第一章节
section_title, section_content = re_find(p_sec, document)
```

   这个正则表达式匹配`\\section{}`标签和相应章节内容之间的内容。子章节的提取过程类似。

5. **提取itemize环境：**

   使用以下正则表达式来提取文档中的itemize环境，即位于`\begin{itemize}`和`\end{itemize}`之间的内容：

```
p_itemize = re.compile(r'\\begin{itemize}(.+?)\\end{itemize}', re.S)
itemize = re.findall(p_itemize, document)
```

   这将提取一个或多个itemize环境。

6. **提取文本格式化（如粗体和强调）：**

   使用以下正则表达式来提取文档中的粗体和强调标签内容：

```
re_tbf = re.compile(r'\\textbf{(.+?)}', re.S)
tbf = re.findall(re_tbf, document)
```

这个正则表达式匹配`\\textbf{}`标签内的内容，类似地，使用`re_emph`正则表达式来提取强调标签内容。

提取 latex 中的表格的部分:

1. **列表闭合标签：**

```
html_text += '</ul>\n\n'
```

   这一行将HTML列表元素`<ul>`（无序列表）的闭合标签`</ul>`添加到`html_text`，以结束之前提取的itemize环境。

2. **表格开始标签：**

```
html_text += '<table border="1">\n'
html_text += "<tr>\n"
html_text += "<th width='40%'>Command</th>\n"
html_text += "<th width='40%'>Level</th>\n"
html_text += "</tr>\n"
```

   这部分代码添加一个HTML表格`<table>`，其中包括了一个表头行`<tr>`和两个表头单元格`<th>`。这个表格用于呈现从LaTeX文档中提取的表格数据。

   - `border="1"`设置了表格的边框，使其具有边框。
   - `<th>`标签表示表头单元格，这里定义了两个列的表头，分别为 "Command" 和 "Level"。

3. **表格数据行：**

```
for i in range(len(col1)):
    html_text += "<tr>\n"
    html_text += "<td>%s</td>\n" % col1[i]
    html_text += "<td>%s</td>\n" % col2[i]
    html_text += "</tr>\n"
```

这个循环遍历从LaTeX文档中提取的表格数据，并将其添加到HTML表格中。每一行都包括两个数据单元格 `<td>`，分别显示 `col1[i]` 和 `col2[i]` 中的内容。

4. **文本格式化标签（粗体和强调）：**

```
# 16.Write textbf label
for item in tbf:
    html_text += '<strong>%s</strong>\n\n' % item

# 17.Write emph label
for item in emph:
    html_text += '<em>%s</em>\n\n' % item
```

这部分代码将从LaTeX文档中提取的粗体和强调文本格式化标签转换为HTML标签。对于粗体，使用 `<strong>` 标签，对于强调，使用 `<em>` 标签，然后将文本内容包裹在这些HTML标签内。

四. 实验总结

本次实验的主要目的是从一个 LaTeX 文档中提取内容，将其转换为 HTML 格式，最终将 HTML 转换为 PDF 格式。以下本次试验的总结:

1.正则表达式提取内容：

使用正则表达式从 LaTeX 文档中提取标题、摘要、章节、子章节、itemize 环境、tabular 环境以及文本格式化（如粗体和强调）等各种元素。

内容清理和结构化：

2.清理提取的内容，去除多余的空格和换行符。

3.使用 HTML 标签对提取的内容进行结构化，如使用 <h1> 标签表示标题，<p> 标签表示段落， 标签表示无序列表，<table> 标签表示表格，以及 和 标签表示粗体和强调文本。

HTML 生成：

4.生成 HTML 文本，将提取的内容以合适的 HTML 标记嵌入其中，以保持文档的结构和格式。

HTML 转 PDF：

5.最终使用 html2pdf 库将生成的 HTML 文本转换为 PDF 文件。