

README

南方小智

2020 年 11 月 24 日

目录

第一章	概述	2
1.1	处理单个文档	2
1.2	处理多个文档	3
1.3	生成图片文件	4
1.4	所有测试用例	4
第二章	Markdown 支持	7
2.1	特殊字符	7
2.2	导入代码文件	7
2.3	导入 tex 格式的图片文件	11
2.4	导入目标文件格式的文件	12
2.5	导入 dot 格式的图片文件	15
2.6	Markdown 的解析和渲染	18
第三章	TODO	23

第一章 概述

通过运行以下命令，可以将本 Markdown 文档生成 pdf 格式，建议阅读 pdf 版本的 README。

```
1 make readme
```

本工具用于个人辅助写作和作图等，满足各种文档编辑的需求。

- 该系统写作部分以基本的 Markdown 语法为核心，作图部分使用 Latex 和 dot 等通用工具包。
- 该系统会生成 Markdown 语法树，可通过 Latex 渲染生成 pdf 等，也可以通过其他渲染方式生成 Html, Word 等格式，方便发布文章。
- 该系统支持将批量的 Markdown 文档整理成册，比如合成一本书，并设置封面和章节目录等。
- 该系统可以通过 pgfplot, tikz 等工具，快速制作复杂的数学图片。

1.1 处理单个文档

本系统可以将一个 Markdown 文档转化为 tex 文档，并自动生成封面和目录，后续还通过 latex 工具包转化为 pdf。只需要将文档路径传入 path。

```
1 python3 create_book.py --path README.md --name README --author 南方小智
2 @echo 'xelatex cmd support Chinese'
3 xelatex -output-directory log README.tex
4 @echo 'run twice to build toc correctly'
5 xelatex -output-directory log README.tex
```

- name 代表生成的文件名和封面标题。
- author 代表生成的封面作者。

- 生成的 tex 文件在 log 文件夹中。

1.2 处理多个文档

本系统可以将多个 Markdown 文档合成一个 tex 文档，并自动生成封面和目录，后续还通过 latex 工具包转化为 pdf。只需要将所有文档放在同一个文件夹里，并将文件夹路径传入 path。

```
1 python3 create_book.py --path examples/趣题集/ --name 趣题集 --bg images/趣题集/background.jpeg --author 南方小智
2 @echo 'xelatex cmd support Chinese'
3 xelatex -output-directory log 趣题集.tex
4 @echo 'run twice to build toc correctly'
5 xelatex -output-directory log 趣题集.tex
```

- 支持多层文件夹，每层文件夹对应一个目录级别，比如文件夹内第一层目录每个文件夹名字为每章的名字，第二层目录每个文件夹名字为每节的名字。
- MAIN 文件为保留文件名，为该层目录对应章节的导言部分。
- _INDEX 文件为保留文件名，用于对文件夹（章节）进行排序，目前章节的排列方法是较长的章节放在靠前的位置。
- 名字以 “_” 开始的文件夹或者文件会被忽略。

_INDEX 文件中按顺序列举该层目录的标题，未列举的标题将放在 “*” 的位置。如果 “*” 未标注在文件中，则未列举的标题默认排到最后。

```
1 概率统计
2 几何原本
3 逻辑思维
4 博弈游戏
5 数学皇后
6 魔法幻术
7 思想实验
8 思维锦囊
```

1.3 生成图片文件

本系统可以编译单一 tex 文档，并生成图片。只需要使用 simple 参数，就可以省略标题，目录，页码等，生成一个独立的图片形式，后续可以用 convert 命令将 pdf 转为图片。

```
1 python3 create_book.py --path images/趣题集/三角形悖论/image0.tex --name 三角
   形 --simple
2 xelatex -output-directory log 三角形.tex
3 @echo '需要安装brew install imagemagick'
4 convert -density 300 log/三角形.pdf -quality 90 log/三角形.png
```

1.4 所有测试用例

```
1 make readme      # 将本文档生成pdf
2 make all         # 生成一本《趣题集》
3 make image       # 生成一张png图片（需要安装brew install imagemagick）
4 make clean       # 清空log文件
5 make clean_all   # 清空所有生成文件，包括，log文件，tex文件， pdf文件等
```

具体细节请参照 makefile:

```
1 all:
2   python3 create_book.py --path examples/趣题集/ --name 趣题集 --bg images/趣
   题集/background.jpeg --author 南方小智
3   @echo 'xelatex cmd support Chinese'
4   xelatex -output-directory log 趣题集.tex
5   @echo 'run twice to build toc correctly'
6   xelatex -output-directory log 趣题集.tex
7   open log/趣题集.pdf
8
9 algo:
10  python3 create_book.py --path examples/Algorithm/ --name Algorithm --author
   南方小智
11  @echo 'xelatex cmd support Chinese'
12  xelatex -output-directory log Algorithm.tex
13  @echo 'run twice to build toc correctly'
14  xelatex -output-directory log Algorithm.tex
```

```

15  open log/Algorithm.pdf
16
17  readme:
18  python3 create_book.py --path README.md --name README --author 南方小智
19  @echo 'xelatex cmd support Chinese'
20  xelatex -shell-escape -output-directory log README.tex
21  @echo 'run twice to build toc correctly'
22  xelatex -shell-escape -output-directory log README.tex
23  open log/README.pdf
24
25  image:
26  python3 create_book.py --path images/趣题集/三角形悖论/image0.tex --name 三
    角形 --simple
27  xelatex -output-directory log 三角形.tex
28  @echo '需要安装brew install imagemagick'
29  convert -density 300 log/三角形.pdf -quality 90 log/三角形.png
30  open log/三角形.png
31
32  test:
33  python3 test.py --path README.md
34
35  git:
36  git push https://github.com/JimmyFromSYSU/latex.git master
37
38  temp:
39  python3 create_book.py --path books/全栈开发项目实践/ --name 全栈开发项目实
    践 --author "" --output ignore
40  @echo 'xelatex cmd support Chinese'
41  xelatex -output-directory ignore 全栈开发项目实践.tex
42  @echo 'run twice to build toc correctly'
43  xelatex -output-directory ignore 全栈开发项目实践.tex
44  open ignore/全栈开发项目实践.pdf
45
46  clean:
47  rm -f log/*.aux
48  rm -f log/*.toc
49  rm -f log/*.log

```

```
50  rm -f log/*.out
51
52 clean_all:
53  rm -f log/*.aux
54  rm -f log/*.toc
55  rm -f log/*.log
56  rm -f log/*.out
57  rm -f log/*.tex
58  rm -f log/*.png
59  rm -f log/*.pdf
60
61 schedule:
62  python3 create_book.py --path books/每日规划 --name 每日规划 --author 南方小
    智 --is_article --output ignore --level 2 --config RAW_PAGE
63  @echo 'xelatex cmd support Chinese'
64  xelatex -output-directory ignore 每日规划.tex
65  @echo 'run twice to build toc correctly'
66  xelatex -output-directory ignore 每日规划.tex
67  open ignore/每日规划.pdf
```

第二章 Markdown 支持

基本语法请参照：[markdown wiki](#)

2.1 特殊字符

在 Markdown 文件中使用特殊字符时需要进行特殊处理，比如添加转义字符 \

```
1 _ -> \_    # _ 在Markdown中可用于表示斜体
2 * -> \*    # * 在Markdown中可用于表示斜体
3 $ -> \$    # $ 在Markdown中可用于开始数学表达式模式
4 \ -> \\    # \ 是特殊的Latex符号
```

2.2 导入代码文件

```
1 {{create_book.py}}[code:Python] # 导入Python格式的代码文件

1 #!/usr/bin/python
2 # -*- coding: UTF-8 -*-
3 from src.render.latex import DOCUMENT, ABSTRACT, SECTION, NEW_LINE
4 from src.utils.path import read, write, trim_ext
5 from src.converter import md2tex
6 from src.constants import LOGGER_FORMAT
7 from src.config import get_config
8 from os import listdir
9 from os.path import isfile, isdir
10 import markdown
11 import argparse
```



```

12 import logging
13 logger = logging.getLogger()
14 logging.basicConfig(level=logging.INFO, format=LOGGER_FORMAT)
15
16 MAIN_FILE = "MAIN"
17 INDEX_FILE = "_INDEX"
18
19
20 def get_title(section: str) -> str:
21     return trim_ext(section)
22
23
24 def create_by_folder(args) -> str:
25     def travel(path: str, level: int = 0) -> str:
26         # TODO: refactoring into a class
27         content = ""
28         sections = listdir(path)
29
30         # Add main content in this section before add sub sections.
31         if MAIN_FILE in sections:
32             child_path = path + "/" + MAIN_FILE
33             if isfile(child_path):
34                 file_content = md2tex(read(child_path), level + 1)
35                 content += NEW_LINE([file_content])
36
37         str_sections = []
38         sections_titles = []
39         for section in sections:
40             # ignore this section for now if it's prefix with '_'
41             if len(section) > 0 and section[0] in ['_', '.']:
42                 continue
43             child_path = path + "/" + section
44             if isfile(child_path) and section != MAIN_FILE:
45                 file_content = md2tex(read(child_path), level + 1)
46                 str_sections.append(SECTION(section, file_content, level))
47                 sections_titles.append(section)
48             elif isdir(child_path):

```

```

49         str_section = NEW_LINE([
50             SECTION(section, "", level),
51             travel(child_path, level=level+1)
52         ])
53         str_sections.append(str_section)
54         sections_titles.append(section)
55
56     # Sort sections by _INDEX file.
57     if INDEX_FILE in sections:
58         child_path = path + "/" + INDEX_FILE
59         titles = read(child_path).split("\n")
60         orders = {}
61         level = 1
62         for title in titles:
63             orders[title] = level
64             level += 1
65         if "*" not in list(orders.keys()):
66             orders["*"] = level
67         section_levels = [
68             orders[title] if title in list(orders.keys()) else orders["*"]
69             for title in sections_titles
70         ]
71         str_sections = [x for _,x in sorted(zip(section_levels,
str_sections))]
72     else:
73         str_sections = sorted(str_sections, key=len, reverse=True)
74         content += NEW_LINE(str_sections)
75         return content
76     return travel(args.path)
77
78
79
80 def create_by_file(args) -> str:
81     content = md2tex(read(args.path), level=args.level)
82     return content
83
84

```

```

85 def get_args() -> argparse.Namespace:
86     logger.info("Parse arguments.")
87     parser = argparse.ArgumentParser()
88     parser.add_argument(
89         "--path", help="The folder/file path for the book", required=True
90     )
91     parser.add_argument(
92         "--bg", help="Background image for the title page", required=False
93     )
94     parser.add_argument(
95         "--simple", help="Output a simple standalone pdf", required=False,
96         action="store_true"
97     )
98     parser.add_argument(
99         "--is_article", help="Output an article rather than a book (ignore
100         title, toc, etc)", required=False, action="store_true"
101     )
102     parser.add_argument(
103         "--level", help="The first level of the book section, default to 0:
104         chapter", default=0, required=False, type=int,
105     )
106     parser.add_argument(
107         "--config", help="The config name of the book", required=False,
108         default='DEFAULT',
109     )
110     parser.add_argument(
111         "--name", help="The name of the book", required=True
112     )
113     parser.add_argument(
114         "--output", help="The output folder for the pdf and log files, default
115         = 'log'", required=False, default='log',
116     )
117     parser.add_argument(
118         "--author", help="The author of the book", required=False
119     )
120     return parser.parse_args()

```

```

117
118 if __name__ == "__main__":
119     args = get_args()
120     logger.info(args)
121     if args.output:
122         output = f"{args.output}/{args.name}.tex"
123     output_content = ""
124     if isdir(args.path):
125         output_content = create_by_folder(args)
126     elif isfile(args.path):
127         output_content = create_by_file(args)
128
129     write(
130         output,
131         DOCUMENT(
132             config=get_config(args.config),
133             book_title=args.name,
134             content=output_content,
135             bg_image=args.bg,
136             book_author=args.author,
137             is_simple=args.simple,
138             is_article=args.is_article,
139         )
140     )

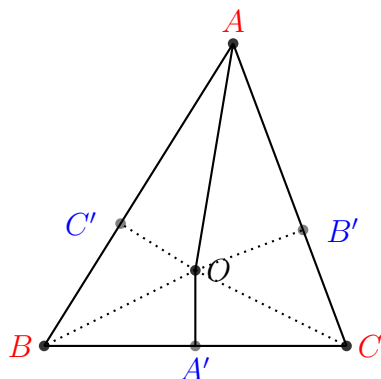
```

2.3 导入 tex 格式的图片文件

```

1 \{\{images/趣题集/三角形悖论/image1.tex\}\}[image] # 导入tex格式的图片文件

```



2.4 导入目标文件格式的文件

有时 Markdown 不支持所有的语法，比如复杂的 latex 表格，这个时候可以在 markdown 文档中导入一段纯文本，渲染器会直接导入纯文本并不做附加处理。

```

1 # 导入tex格式的table文件
2 {{tables/table1.tex}}[text]
3
4 # longtable用于多页表格
5 # longtable不能嵌套在table环境内
6 # | 代表列分隔线，0.2代表该列占宽度的20%。
7 \begin{longtable}{p{0.2\textwidth} | p{0.25\textwidth} p{0.55\textwidth}}
8 \toprule
9 \end{longtable}
10
11 # hline行分隔线，cline部分行分隔线
12 \hline
13 \cline{2-3}

```

Table Name	Column Name	Desc/Type
DiceHistory	dice1	PostiveInt
	dice2	PostiveInt
	player	Player（哪位玩家掷的骰子）
	turn_id	Int（第几轮掷的骰子）

	(action_id)	Int (代表这个操作是该场游戏的第几个 action, 用来重播游戏过程, 用于 Debug)
	game	Game
RobberHistory	x	Int (Robber 所在位置的 x 坐标)
	y	Int (Robber 所在位置的 y 坐标)
	player	Player
	is_knight	Bool (True 代表是由 Knight 技能卡牌触发的 Robber, False 代表通过掷骰子到数字 7 触发的 Robber)
	is_latest	Bool (是否最后一个 Robber 操作, 可用于查询当前 Robber 位置, 每个游戏只有一个 true value)
	victim	Player (受害者玩家)
	(cardset_movement)	CardsetMovement (抽牌转移情况)
	turn_id	Int (第几轮发生的 Robber 事件)
	(action_id)	Int (代表这个操作是该场游戏的第几个 action, 用来重播游戏过程, 用于 Debug)
	game	Game
Cardset	lumber	Int (default=0)
	brick	Int (default=0)
	wool	Int (default=0)
	grain	Int (default=0)
	ore	Int (default=0)
	dev_knight	Int (default=0)
	dev_one_victory_point	Int (default=0)
	dev_road_building	Int (default=0)
	dev_monopoly	Int (default=0)
	dev_year_of_plenty	Int (default=0)
Player	card_set	CardSet
	order	Int (取值范围是 0 玩家数, 代表该玩家是第几个开始行动的玩家。)

	color	Chars (玩家的颜色)
	knight number	(可以根据 RobberHistory 进行计算)
	(user)	Int (用户 id, 外层的 Portal 系统负责用户的注册, 有用户名, 头像等信息。头像信息也可以放到每个 Game 里, 也就是每个游戏可以随时设置不同头像)
	game	Game
Construction	type	Chars (可选 House, Town, Road)
	owner	Player (物件所属玩家)
	x	Int (所在位置 x 坐标)
	y	Int (所在位置 y 坐标)
	z	Int (所在位置 z 坐标)
	game	Game
Tile	type	Chars (可选五种基本资源, Sea, Desert)
	number	Int (每个地块上的数字, 2 ~ 12)
	x	Int (所在位置 x 坐标)
	y	Int (所在位置 y 坐标)
	game	Game
HarborSea	type	Chars (可选五种基本资源, Any3)
	x	Int (所在海洋位置 x 坐标)
	y	Int (所在海洋位置 y 坐标)
	game	Game
HarborLand	x	Int (所在位置 x 坐标)
	y	Int (所在位置 x 坐标)
	z	Int (所在位置 x 坐标)
	sea	HarborSea
	game	Game
Bank	cardset	Cardset
	game	Game
	map_name	Chars (游戏用的地图模版名)

turn_id	Int (当前是该场游戏的第几个回合, base=0, 每个回合可能有多个 action, 前 2N 个回合为 Settle 阶段, N 为玩家数)
status	Chars (游戏当前的阶段, 包括 settle: 放房子阶段。main: 主游戏阶段。end: 游戏结束显示结果阶段。)
number_of_player	Int (总玩家数量, 也可以从 Player 表计算)
(action_id)	Int (当前是该场游戏的第几个 action)
curr_player	Player (当前玩家)

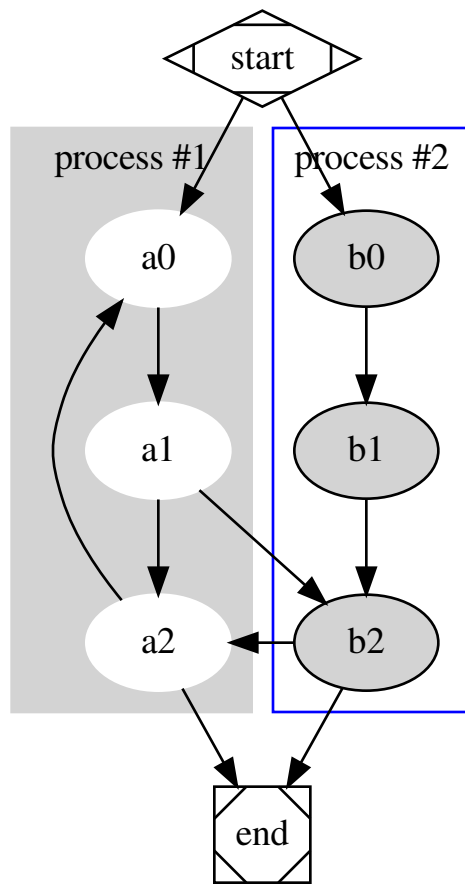
表 2.1: 数据库设计

2.5 导入 dot 格式的图片文件

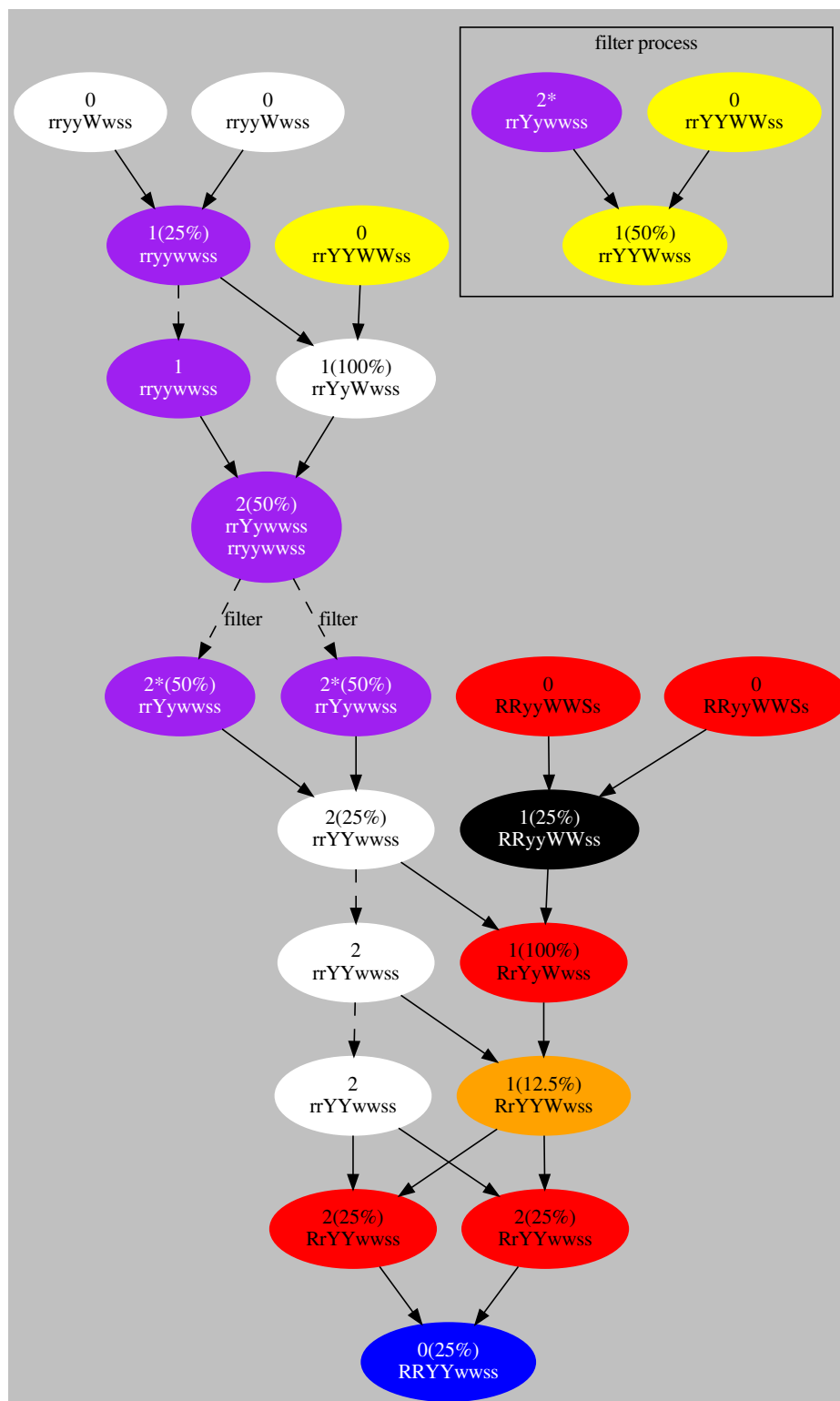
注意: 需要安装[Graphviz](#)。

导入的 dot 文件会通过 dot 命令转化为 eps 格式, 并作为图片导入到 tex 文件中。dot 的简单用法可以参照[GraphViz Documentation](#)。

```
1 {{images/README/image0.dot}}[dot: 测试dot] # 导入dot格式/graphviz语法的图片文件, 并用“测试dot”作为标题。
```

测试 dot

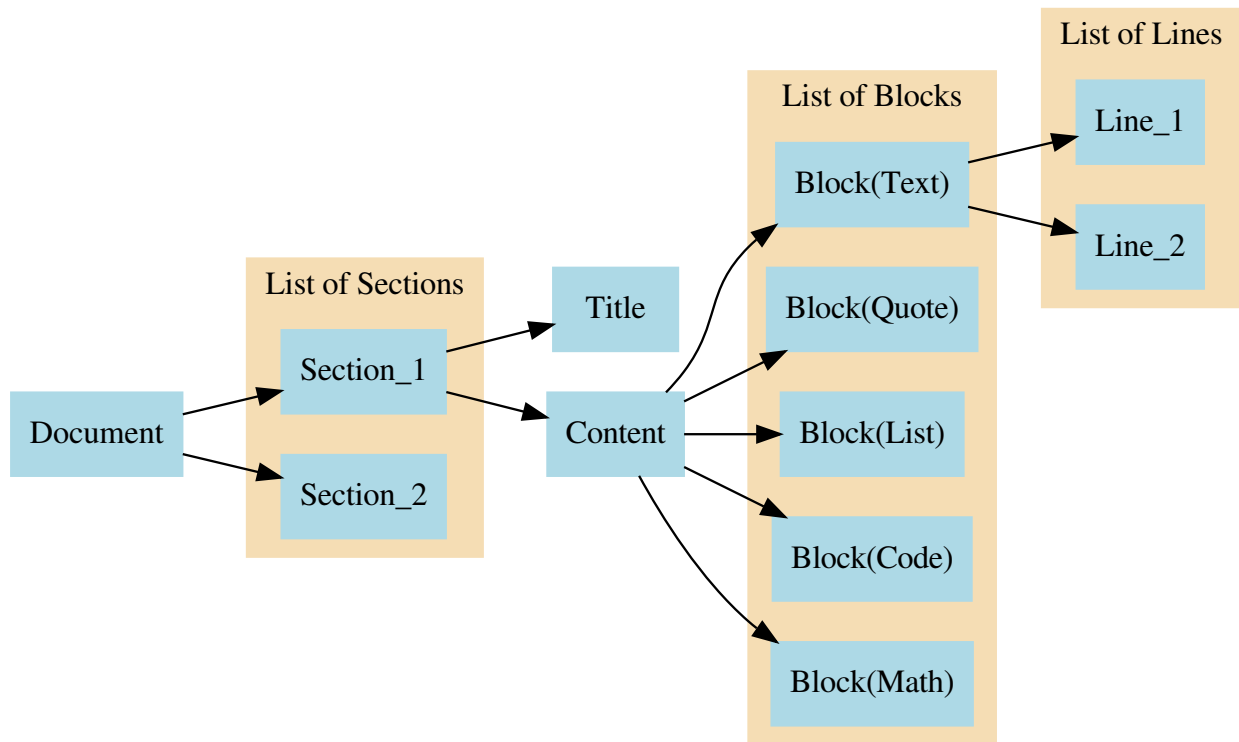


2.6 Markdown 的解析和渲染

该系统自带一个 Markdown 解析器，会将 Markdown 文本转化为一棵语法树。使用如下命令可以打印出本 Readme 文档的语法树：

```
1 # python3 test.py --path README.md
2 make test
```

Markdown 语法树的解析遵循如图的树状结构。从最顶端的 Document 到最底端的 Line，而 inline 部分的语法则较为简单，将由渲染器进行解析。



Markdown 语法树

通过渲染器,可以将 Markdown 渲染成 tex,html,word 等格式。只需要继承 MDRender,实现对应的抽象方法即可添加新的渲染器,比如 src/render/md_tex_render.py。

```
1 #!/usr/bin/python
2 # -*- coding: UTF-8 -*-
3 from typing import List
4 from .constants import ListType, InLineType, IncludeType
5 import abc
6 import re
7
8
9 # use (.*?) to extract the pattern
10 # use (?:) to make the extraction optional
11 # use (.*?) to make a non-greedy match (shortest match)
12 # use (.*?)? to make the match optional
13 class MDRender():
14     def __init__(self, level: int = 0):
15         self.level = level
16         self.render_line_configs = {
17             InLineType.Image: {
18                 'pattern_str': r"!\[([.*?])\]\([.*?])",
19                 'group_num': 2,
20             },
21             InLineType.Link: {
22                 'pattern_str': r"\[([.*?])\]\([.*?])",
23                 'group_num': 2,
24             },
25             InLineType.Bold: {
26                 'pattern_str': r"\*\*([.*?])\*\*",
27                 'group_num': 1,
28             },
29             InLineType.Include: {
30                 'pattern_str': r"\{\{([.*?])\}\}\[([.*?])\]",
31                 'group_num': 2,
32             },
33         }
34
```

```

35     @abc.abstractmethod
36     def render_title(self, title, level) -> str:
37         pass
38
39     @abc.abstractmethod
40     def render_list(self, items: List[str], type_: ListType=ListType.Normal)
-> str:
41         pass
42
43     @abc.abstractmethod
44     def build_inline_link(self, title: str, link: str) -> str:
45         pass
46
47     @abc.abstractmethod
48     def build_inline_image(self, title: str, link: str) -> str:
49         pass
50
51     @abc.abstractmethod
52     def build_inline_bold(self, content: str) -> str:
53         pass
54
55     @abc.abstractmethod
56     def build_inline_include(self, path: str, type_: IncludeType) -> str:
57         pass
58
59     def build_inline_new_pattern(self, match, inline_type: InLineType) -> str:
60         # Image must be before Link, otherwise Image will be treat as Link
because the format
61         # Image: ![]()
62         # Link: []()
63         if inline_type == InLineType.Image:
64             title = match.group(1)
65             link = match.group(2)
66             return self.build_inline_image(title, link)
67         elif inline_type == InLineType.Link:
68             title = match.group(1)
69             link = match.group(2)

```

```

70         return self.build_inline_link(title, link)
71     elif inline_type == InLineType.Bold:
72         content = match.group(1)
73         return self.build_inline_bold(content)
74     elif inline_type == InLineType.Include:
75         path = match.group(1)
76         setting = match.group(2)
77         if len(setting.split(":", 1)) == 2:
78             type_ = setting.split(":", 1)[0]
79             config = setting.split(":", 1)[1]
80         else:
81             type_ = setting
82             config = ""
83         return self.build_inline_include(path, type_, config)
84     return ""
85
86     def render_line_with(self, line, inline_type, config) -> str:
87         changed = True
88         while changed:
89             pattern = re.compile(config['pattern_str'], re.IGNORECASE)
90             match = pattern.search(line)
91             if match and len(match.groups()) == config['group_num']:
92                 old_pattern = match.group(0)
93                 new_pattern = self.build_inline_new_pattern(match, inline_type
94 )
95                 line = line.replace(old_pattern, new_pattern)
96             else:
97                 changed = False
98         return line
99
100     @abc.abstractmethod
101     def render_line(self, line: str) -> str:
102         for inline_type, config in self.render_line_configs.items():
103             line = self.render_line_with(line, inline_type, config)
104         return line
105
106     @abc.abstractmethod

```

```
106     def render_blockquote(self, content) -> str:
107         pass
108
109     @abc.abstractmethod
110     def render_math(self, content) -> str:
111         pass
112
113     @abc.abstractmethod
114     def render_code(self, content, language) -> str:
115         pass
```

第三章 TODO

- [WIP] 目前很多 latex 设置都是 Hard Code 的，应当将这些设置变成可配置的：
 - 目前最高层目录默认为 Chapter，设置 level 参数使其可配置化
 - 可以设置加粗字体颜色和链接颜色。
- 支持表格，以及 merge cell。
- 代码块内无法在行首使用 #，会被解析成 section。
- 目前 Markdown 解析器仍然有许多 Markdown 基本语法并不支持，需要：
 - 支持行内斜体，下划线等。
 - 支持行内代码。
 - [Done] 支持一般图片格式的导入。
 - [Done] 支持嵌套 List。
 - * 测试 3 层嵌套。
 - 支持表格
- 目前 Latex 渲染器仍然有一些 Latex 用法支持不够，需要：
 - 支持更多类似 \$, <, > 等 latex 特殊字符。
- 支持更多的 Features：
 - 使用 folder 模式的时候，兼容.md 等后缀。
 - 添加页面背景水印，这样可以防止发布的材料被盗版使用。
 - 添加程序辅助画图（特别是制作复杂的各类数学图片）。
 - 添加作者介绍。
 - 添加 html 和 word 渲染器。
 - 支持 latex 画思维导图。
- BUG:
 - tex 图片嵌入应该像 dot 图片嵌入一样支持设置标题，如 [image:title]。

- Better Engineering:
 - md_tree 的 Parser 主体可以用自动机的方式实现，从而减少 if/else 的判断和重复代码片。



扫码支持作者