

A Markdown Interpreter for T_EX

Vít Novotný
witiko@mail.muni.cz

Version 2.8.2
March 21, 2020

Contents

1	Introduction	1	2.3	L ^A T _E X Interface	32
1.1	Feedback	2	2.4	ConT _E Xt Interface	42
1.2	Acknowledgements	2	3	Implementation	42
1.3	Requirements	2	3.1	Lua Implementation	43
2	Interfaces	5	3.2	Plain T _E X Implementation	136
2.1	Lua Interface	5	3.3	L ^A T _E X Implementation	145
2.2	Plain T _E X Interface	17	3.4	ConT _E Xt Implementation	157

1 Introduction

The Markdown package¹ converts markdown² markup to T_EX commands. The functionality is provided both as a Lua module and as plain T_EX, L^AT_EX, and ConT_EXt macro packages that can be used to directly typeset T_EX documents containing markdown markup. Unlike other converters, the Markdown package does not require any external programs, and makes it easy to redefine how each and every markdown element is rendered. Creative abuse of the markdown syntax is encouraged. ;-)

This document is a technical documentation for the Markdown package. It consists of three sections. This section introduces the package and outlines its prerequisites. Section 2 describes the interfaces exposed by the package. Section 3 describes the implementation of the package. The technical documentation contains only a limited number of tutorials and code examples. You can find more of these in the user manual.³

```
1 local metadata = {
2   version   = "2.8.2",
3   comment   = "A module for the conversion from markdown to plain TeX",
4   author    = "John MacFarlane, Hans Hagen, Vít Novotný",
5   copyright = {"2009-2016 John MacFarlane, Hans Hagen",
6               "2016-2020 Vít Novotný"},
7   license   = "LPPL 1.3"
8 }
9
```

¹See <https://ctan.org/pkg/markdown>.

²See <https://daringfireball.net/projects/markdown/basics/>.

³See <http://mirrors.ctan.org/macros/generic/markdown/markdown.html>.

```
10 if not modules then modules = { } end
11 modules['markdown'] = metadata
```

1.1 Feedback

Please use the Markdown project page on GitHub⁴ to report bugs and submit feature requests. If you do not want to report a bug or request a feature but are simply in need of assistance, you might want to consider posting your question on the T_EX-L^AT_EX Stack Exchange.⁵

1.2 Acknowledgements

The Lunamark Lua module provides speedy markdown parsing for the package. I would like to thank John Macfarlane, the creator of Lunamark, for releasing Lunamark under a permissive license.

Funding by the the Faculty of Informatics at the Masaryk University in Brno [1] is gratefully acknowledged.

The T_EX implementation of the package draws inspiration from several sources including the source code of L^AT_EX 2_ε, the minted package by Geoffrey M. Poore, which likewise tackles the issue of interfacing with an external interpreter from T_EX, the filecontents package by Scott Pakin and others.

1.3 Requirements

This section gives an overview of all resources required by the package.

1.3.1 Lua Requirements

The Lua part of the package requires that the following Lua modules are available from within the LuaT_EX engine:

LPeg \geq 0.10 A pattern-matching library for the writing of recursive descent parsers via the Parsing Expression Grammars (PEGs). It is used by the Lunamark library to parse the markdown input. LPeg \geq 0.10 is included in LuaT_EX \geq 0.72.0 (T_EXLive \geq 2013).

```
12 local lpeg = require("lpeg")
```

Selene Unicode A library that provides support for the processing of wide strings. It is used by the Lunamark library to cast image, link, and footnote tags to the lower case. Selene Unicode is included in all releases of LuaT_EX (T_EXLive \geq 2008).

⁴See <https://github.com/witiko/markdown/issues>.

⁵See <https://tex.stackexchange.com>.

```
13 local unicode = require("unicode")
```

MD5 A library that provides MD5 crypto functions. It is used by the Lunamark library to compute the digest of the input for caching purposes. MD5 is included in all releases of LuaTeX (TeXLive \geq 2008).

```
14 local md5 = require("md5")
```

All the abovelisted modules are statically linked into the current version of the LuaTeX engine [2, Section 3.3].

1.3.2 Plain TeX Requirements

The plain TeX part of the package requires that the plain TeX format (or its superset) is loaded, all the Lua prerequisites (see Section 1.3.1), and the following Lua module:

Lua File System A library that provides access to the filesystem via OS-specific syscalls. It is used by the plain TeX code to create the cache directory specified by the `\markdownOptionCacheDir` macro before interfacing with the Lunamark library. Lua File System is included in all releases of LuaTeX (TeXLive \geq 2008).

The plain TeX code makes use of the `isdir` method that was added to the Lua File System library by the LuaTeX engine developers [2, Section 3.2].

The Lua File System module is statically linked into the LuaTeX engine [2, Section 3.3].

Unless you convert markdown documents to TeX manually using the Lua command-line interface (see Section 2.1.5), the plain TeX part of the package will require that either the LuaTeX `\directlua` primitive or the shell access file stream 18 is available in your TeX engine. If only the shell access file stream is available in your TeX engine (as is the case with pdfTeX and XeTeX) or if you enforce the use of shell using the `\markdownMode` macro, then unless your TeX engine is globally configured to enable shell access, you will need to provide the `-shell-escape` parameter to your engine when typesetting a document.

1.3.3 L^ATeX Requirements

The L^ATeX part of the package requires that the L^ATeX 2_ε format is loaded,

```
15 \NeedsTeXFormat{LaTeX2e}%
```

all the plain TeX prerequisites (see Section 1.3.2), and the following L^ATeX 2_ε packages:

keyval A package that enables the creation of parameter sets. This package is used to provide the `\markdownSetup` macro, the package options processing, as well as the parameters of the `markdown*` L^ATeX environment.

16 `\RequirePackage{keyval}`

url A package that provides the `\url` macro for the typesetting of URLs. It is used to provide the default token renderer prototype (see Section 2.2.4) for links.

17 `\RequirePackage{url}`

graphicx A package that provides the `\includegraphics` macro for the typesetting of images. It is used to provide the corresponding default token renderer prototype (see Section 2.2.4).

18 `\RequirePackage{graphicx}`

paralist A package that provides the `compactitem`, `compactenum`, and `compactdesc` macros for the typesetting of tight bulleted lists, ordered lists, and definition lists. It is used to provide the corresponding default token renderer prototypes (see Section 2.2.4).

ifthen A package that provides a concise syntax for the inspection of macro values. It is used to determine whether or not the paralist package should be loaded based on the user options.

19 `\RequirePackage{ifthen}`

fancyvrb A package that provides the `\VerbatimInput` macros for the verbatim inclusion of files containing code. It is used to provide the corresponding default token renderer prototype (see Section 2.2.4).

20 `\RequirePackage{fancyvrb}`

csvsimple A package that provides the default token renderer prototype for iA Writer content blocks with the CSV filename extension (see Section 2.2.4).

21 `\RequirePackage{csvsimple}`

gobble A package that provides the `\@gobblethree` \TeX command.

22 `\RequirePackage{gobble}`

1.3.4 Con \TeX T Prerequisites

The Con \TeX T part of the package requires that either the Mark II or the Mark IV format is loaded, all the plain \TeX prerequisites (see Section 1.3.2), and the following Con \TeX T modules:

m-database A module that provides the default token renderer prototype for iA Writer content blocks with the CSV filename extension (see Section 2.2.4).

2 Interfaces

This part of the documentation describes the interfaces exposed by the package along with usage notes and examples. It is aimed at the user of the package.

Since neither \TeX nor Lua provide interfaces as a language construct, the separation to interfaces and implementations is purely abstract. It serves as a means of structuring this documentation and as a promise to the user that if they only access the package through the interface, the future minor versions of the package should remain backwards compatible.

2.1 Lua Interface

The Lua interface provides the conversion from UTF-8 encoded markdown to plain \TeX . This interface is used by the plain \TeX implementation (see Section 3.2) and will be of interest to the developers of other packages and Lua modules.

The Lua interface is implemented by the `markdown` Lua module.

```
23 local M = {metadata = metadata}
```

2.1.1 Conversion from Markdown to Plain \TeX

The Lua interface exposes the `new(options)` method. This method creates converter functions that perform the conversion from markdown to plain \TeX according to the table `options` that contains options recognized by the Lua interface. (see Section 2.1.2). The `options` parameter is optional; when unspecified, the behaviour will be the same as if `options` were an empty table.

The following example Lua code converts the markdown string `Hello *world*!` to a \TeX output using the default options and prints the \TeX output:

```
local md = require("markdown")
local convert = md.new()
print(convert("Hello *world*!"))
```

2.1.2 Options

The Lua interface recognizes the following options. When unspecified, the value of a key is taken from the `defaultOptions` table.

```
24 local defaultOptions = {}
```

2.1.3 File and Directory Names

`cacheDir=<path>` default: .

A path to the directory containing auxiliary cache files. If the last segment of the path does not exist, it will be created by the Lua command-line and plain T_EX implementations. The Lua implementation expects that the entire path already exists.

When iteratively writing and typesetting a markdown document, the cache files are going to accumulate over time. You are advised to clean the cache directory every now and then, or to set it to a temporary filesystem (such as `/tmp` on UN*X systems), which gets periodically emptied.

```
25 defaultOptions.cacheDir = "."
```

2.1.4 Parser Options

`blankBeforeBlockquote=true, false` default: false

- `true` Require a blank line between a paragraph and the following blockquote.
- `false` Do not require a blank line between a paragraph and the following blockquote.

```
26 defaultOptions.blankBeforeBlockquote = false
```

`blankBeforeCodeFence=true, false` default: false

- `true` Require a blank line between a paragraph and the following fenced code block.
- `false` Do not require a blank line between a paragraph and the following fenced code block.

```
27 defaultOptions.blankBeforeCodeFence = false
```

`blankBeforeHeading=true, false` default: false

- `true` Require a blank line between a paragraph and the following header.
- `false` Do not require a blank line between a paragraph and the following header.

```
28 defaultOptions.blankBeforeHeading = false
```

`breakableBlockquotes=true, false` default: false

- `true` A blank line separates block quotes.
- `false` Blank lines in the middle of a block quote are ignored.

```
29 defaultOptions.breakableBlockquotes = false
```

`citationNbsps=true, false` default: false

- `true` Replace regular spaces with non-breakable spaces inside the prenotes and postnotes of citations produced via the pandoc citation syntax extension.
- `false` Do not replace regular spaces with non-breakable spaces inside the prenotes and postnotes of citations produced via the pandoc citation syntax extension.

```
30 defaultOptions.citationNbsps = true
```

`citations=true, false` default: false

- `true` Enable the pandoc citation syntax extension:

Here is a simple parenthetical citation [`@doe99`] and here is a string of several [`see @doe99, pp. 33-35; also @smith04, chap. 1`].

A parenthetical citation can have a [`prenote @doe99`] and a [`@smith04 postnote`]. The name of the author can be suppressed by inserting a dash before the name of an author as follows [`-@smith04`].

Here is a simple text citation `@doe99` and here is a string of several `@doe99` [`pp. 33-35; also @smith04, chap. 1`]. Here is one with the name of the author suppressed `-@doe99`.

- `false` Disable the pandoc citation syntax extension.

```
31 defaultOptions.citations = false
```

`codeSpans=true, false`

default: true

`true` Enable the code span syntax:

```
Use the printf() function.  
``There is a literal backtick (`) here.``
```

`false` Disable the code span syntax. This allows you to easily use the quotation mark ligatures in texts that do not contain code spans:

```
``This is a quote.``
```

```
32 defaultOptions.codeSpans = true
```

`contentBlocks=true, false`

default: false

`true` Enable the iA Writer content blocks syntax extension [3]:

```
http://example.com/minard.jpg (Napoleon's  
disastrous Russian campaign of 1812)  
/Flowchart.png "Engineering Flowchart"  
/Savings Account.csv 'Recent Transactions'  
/Example.swift  
/Lorem Ipsum.txt
```

`false` Disable the iA Writer content blocks syntax extension.

```
33 defaultOptions.contentBlocks = false
```

`contentBlocksLanguageMap=<filename>`

default: `markdown-languages.json`

The filename of the JSON file that maps filename extensions to programming language names in the iA Writer content blocks. See Section 2.2.3.9 for more information.

```
34 defaultOptions.contentBlocksLanguageMap = "markdown-languages.json"
```


`definitionLists=true, false`

default: false

`true` Enable the pandoc definition list syntax extension:

```
Term 1

: Definition 1

Term 2 with inline markup

: Definition 2

    { some code, part of Definition 2 }

Third paragraph of definition 2.
```

`false` Disable the pandoc definition list syntax extension.

```
35 defaultOptions.definitionLists = false
```

`fencedCode=true, false`

default: false

`true` Enable the commonmark fenced code block extension:

```
~~~ js
if (a > 3) {
  moveShip(5 * gravity, DOWN);
}
~~~~~

``` html


```

  <code>
    // Some comments
    line 1 of code
    line 2 of code
    line 3 of code
  </code>
</pre>
```
```


```

`false` Disable the commonmark fenced code block extension.

```
36 defaultOptions.fencedCode = false
```

`footnotes=true, false`

default: `false`

`true` Enable the pandoc footnote syntax extension:

```
Here is a footnote reference, [^1] and another. [^longnote]

[^1]: Here is the footnote.

[^longnote]: Here's one with multiple blocks.

    Subsequent paragraphs are indented to show that they
    belong to the previous footnote.

        { some.code }

    The whole paragraph can be indented, or just the
    first line. In this way, multi-paragraph footnotes
    work like multi-paragraph list items.

This paragraph won't be part of the note, because it
isn't indented.
```

`false` Disable the pandoc footnote syntax extension.

```
37 defaultOptions.footnotes = false
```

`hashEnumerators=true, false`

default: `false`

`true` Enable the use of hash symbols (#) as ordered item list markers:

```
#. Bird
#. McHale
#. Parish
```

`false` Disable the use of hash symbols (#) as ordered item list markers.

```
38 defaultOptions.hashEnumerators = false
```

`headerAttributes=true, false`

default: `false`

`true` Enable the assignment of HTML attributes to headings:

```
# My first heading {#foo}

## My second heading ## {#bar .baz}

Yet another heading {key=value}
=====
```

These HTML attributes have currently no effect other than enabling content slicing, see the `slice` option.

`false` Disable the assignment of HTML attributes to headings.

```
39 defaultOptions.headerAttributes = false
```

`html=true, false`

default: `false`

`true` Enable the recognition of HTML tags, block elements, comments, HTML instructions, and entities in the input. Tags, block elements (along with contents), HTML instructions, and comments will be ignored and HTML entities will be replaced with the corresponding Unicode codepoints.

`false` Disable the recognition of HTML markup. Any HTML markup in the input will be rendered as plain text.

```
40 defaultOptions.html = false
```

`hybrid=true, false`

default: `false`

`true` Disable the escaping of special plain \TeX characters, which makes it possible to intersperse your markdown markup with \TeX code. The intended usage is in documents prepared manually by a human author. In such documents, it can often be desirable to mix \TeX and markdown markup freely.

`false` Enable the escaping of special plain \TeX characters outside verbatim environments, so that they are not interpreted by \TeX . This is encouraged when typesetting automatically generated content or markdown documents that were not prepared with this package in mind.

```
41 defaultOptions.hybrid = false
```

`inlineFootnotes=true, false` default: false

`true` Enable the pandoc inline footnote syntax extension:

Here is an inline note.^[Inlines notes are easier to write, since you don't have to pick an identifier and move down to type the note.]

`false` Disable the pandoc inline footnote syntax extension.

42 `defaultOptions.inlineFootnotes = false`

`pipeTables=true, false` default: false

`true` Enable the PHP Markdown table syntax extension:

| Right | Left | Default | Center |
|-------|------|---------|--------|
| 12 | 12 | 12 | 12 |
| 123 | 123 | 123 | 123 |
| 1 | 1 | 1 | 1 |

`false` Disable the PHP Markdown table syntax extension.

43 `defaultOptions.pipeTables = false`

`preserveTabs=true, false` default: false

`true` Preserve tabs in code block and fenced code blocks.

`false` Convert any tabs in the input to spaces.

44 `defaultOptions.preserveTabs = false`

`shiftHeadings=<shift amount>` default: 0

All headings will be shifted by *<shift amount>*, which can be both positive and negative. Headings will not be shifted beyond level 6 or below level 1. Instead, those headings will be shifted to level 6, when *<shift amount>* is positive, and to level 1, when *<shift amount>* is negative.

45 `defaultOptions.shiftHeadings = 0`

`slice`=*<the beginning and the end of a slice>* default: `^ $`

Two space-separated selectors that specify the slice of a document that will be processed, whereas the remainder of the document will be ignored. The following selectors are recognized:

- The circumflex (`^`) selects the beginning of a document.
- The dollar sign (`$`) selects the end of a document.
- `^<identifier>` selects the beginning of a section with the HTML attribute `#<identifier>` (see the `headerAttributes` option).
- `$<identifier>` selects the end of a section with the HTML attribute `#<identifier>`.
- `<identifier>` corresponds to `^<identifier>` for the first selector and to `$<identifier>` for the second selector.

Specifying only a single selector, `<identifier>`, is equivalent to specifying the two selectors `<identifier> <identifier>`, which is equivalent to `^<identifier> $<identifier>`, i.e. the entire section with the HTML attribute `#<identifier>` will be selected.

```
46 defaultOptions.slice = "^ $"
```

`smartEllipses`=`true, false` default: `false`

- | | |
|--------------------|---|
| <code>true</code> | Convert any ellipses in the input to the <code>\markdownRendererEllipsis</code> \TeX macro. |
| <code>false</code> | Preserve all ellipses in the input. |

```
47 defaultOptions.smartEllipses = false
```

`startNumber`=`true, false` default: `true`

- | | |
|--------------------|---|
| <code>true</code> | Make the number in the first item of an ordered lists significant. The item numbers will be passed to the <code>\markdownRendererOliItemWithNumber</code> \TeX macro. |
| <code>false</code> | Ignore the numbers in the ordered list items. Each item will only produce a <code>\markdownRendererOliItem</code> \TeX macro. |

```
48 defaultOptions.startNumber = true
```

`tableCaptions=true, false`

default: `false`

`true` Enable the Pandoc `table_captions` syntax extension for pipe tables (see the `pipeTables` option).

| Right | Left | Default | Center |
|-------|------|---------|--------|
| 12 | 12 | 12 | 12 |
| 123 | 123 | 123 | 123 |
| 1 | 1 | 1 | 1 |

: Demonstration of pipe table syntax.

`false` Enable the Pandoc `table_captions` syntax extension.

49 `defaultOptions.tableCaptions = false`

`tightLists=true, false`

default: `true`

`true` Lists whose bullets do not consist of multiple paragraphs will be passed to the `\markdownRenderer01BeginTight`, `\markdownRenderer01EndTight`, `\markdownRendererU1BeginTight`, `\markdownRendererU1EndTight`, `\markdownRendererD1BeginTight`, and `\markdownRendererD1EndTight` TeX macros.

`false` Lists whose bullets do not consist of multiple paragraphs will be treated the same way as lists that do consist of multiple paragraphs.

50 `defaultOptions.tightLists = true`

`underscores=true, false`

default: `true`

`true` Both underscores and asterisks can be used to denote emphasis and strong emphasis:

```
*single asterisks*
_single underscores_
**double asterisks**
__double underscores__
```

`false` Only asterisks can be used to denote emphasis and strong emphasis. This makes it easy to write math with the `hybrid` option without the need to constantly escape subscripts.

51 `defaultOptions.underscores = true`

2.1.5 Command-Line Interface

To provide finer control over the conversion and to simplify debugging, a command-line Lua interface for converting a Markdown document to \TeX is also provided.

```
52
53 HELP_STRING = [[
54 Usage: texlua ]] .. arg[0] .. [[ [OPTIONS] -- [INPUT_FILE] [OUTPUT_FILE]
55 where OPTIONS are documented in the Lua interface section of the
56 technical Markdown package documentation.
57
58 When OUTPUT_FILE is unspecified, the result of the conversion will be
59 written to the standard output. When INPUT_FILE is also unspecified, the
60 result of the conversion will be read from the standard input.
61
62 Report bugs to: witiko@mail.muni.cz
63 Markdown package home page: <https://github.com/witiko/markdown>]]
64
65 VERSION_STRING = [[
66 markdown-cli.lua (Markdown) ]] .. metadata.version .. [[
67
68 Copyright (C) ]] .. table.concat(metadata.copyright,
69                                     "\nCopyright (C) ") .. [[
70
71 License: ]] .. metadata.license
72
73 local function warn(s)
74   io.stderr:write("Warning: " .. s .. "\n") end
75
76 local function error(s)
77   io.stderr:write("Error: " .. s .. "\n")
78   os.exit(1) end
79
80 local process_options = true
81 local options = {}
82 local input_filename
83 local output_filename
84 for i = 1, #arg do
85   if process_options then
86     if arg[i] == "--" then
87       process_options = false
88       goto continue
```

Unless the `--` argument has been specified before, an argument containing the equals sign (=) is assumed to be an option specification in a $\langle key \rangle = \langle value \rangle$ format. The available options are listed in Section 2.1.2.

```
89     elseif arg[i]:match("=") then
90         key, value = arg[i]:match("(.)=(.*)")
```

The `defaultOptions` table is consulted to identify whether $\langle value \rangle$ should be parsed as a string or as a boolean.

```
91         default_type = type(defaultOptions[key])
92         if default_type == "boolean" then
93             options[key] = (value == "true")
94         else
95             if default_type ~= "string" then
96                 if default_type == "nil" then
97                     warn('Option "' .. key .. '" not recognized.')
98                 else
99                     warn('Option "' .. key .. '" type not recognized, please file ' ..
100                         'a report to the package maintainer.')
101                 end
102                 warn('Parsing the ' .. 'value "' .. value ..'" of option "' ..
103                     key .. '" as a string.')
104             end
105             options[key] = value
106         end
107         goto continue
```

Unless the `--` argument has been specified before, an argument `--help`, or `-h` causes a brief documentation for how to invoke the program to be printed to the standard output.

```
108     elseif arg[i] == "--help" or arg[i] == "-h" then
109         print(HELP_STRING)
110         os.exit()
```

Unless the `--` argument has been specified before, an argument `--version`, or `-v` causes the program to print information about its name, version, origin and legal status, all on standard output.

```
111     elseif arg[i] == "--version" or arg[i] == "-v" then
112         print(VERSION_STRING)
113         os.exit()
114     end
115 end
```

The first argument that matches none of the above patterns is assumed to be the input filename. The input filename should correspond to the Markdown document that is going to be converted to a TeX document.

```
116 if input_filename == nil then
117     input_filename = arg[i]
```


The first argument that matches none of the above patterns is assumed to be the output filename. The output filename should correspond to the \TeX document that will result from the conversion.

```
118 elseif output_filename == nil then
119     output_filename = arg[i]
120 else
121     error('Unexpected argument: "' .. arg[i] .. "'.')
122 end
123 ::continue::
124 end
```

The command-line Lua interface is implemented by the `markdown-cli.lua` file that can be invoked from the command line as follows:

```
texlua /path/to/markdown-cli.lua cacheDir=. -- hello.md hello.tex
```

to convert the Markdown document `hello.md` to a \TeX document `hello.tex`. After the Markdown package for our \TeX format has been loaded, the converted document can be typeset as follows:

```
\input hello
```

This shows another advantage of using the command-line interface compared to using a higher-level \TeX interface: it is unnecessary to provide shell access for the \TeX engine.

2.2 Plain \TeX Interface

The plain \TeX interface provides macros for the typesetting of markdown input from within plain \TeX , for setting the Lua interface options (see Section 2.1.2) used during the conversion from markdown to plain \TeX and for changing the way markdown the tokens are rendered.

```
125 \def\markdownLastModified{2020/03/20}%
126 \def\markdownVersion{2.8.2}%
```

The plain \TeX interface is implemented by the `markdown.tex` file that can be loaded as follows:

```
\input markdown
```

It is expected that the special plain \TeX characters have the expected category codes, when `\inputting` the file.

2.2.1 Typesetting Markdown

The interface exposes the `\markdownBegin`, `\markdownEnd`, and `\markdownInput` macros.

The `\markdownBegin` macro marks the beginning of a markdown document fragment and the `\markdownEnd` macro marks its end.

```
127 \let\markdownBegin\relax
128 \let\markdownEnd\relax
```

You may prepend your own code to the `\markdownBegin` macro and redefine the `\markdownEnd` macro to produce special effects before and after the markdown block.

There are several limitations to the macros you need to be aware of. The first limitation concerns the `\markdownEnd` macro, which must be visible directly from the input line buffer (it may not be produced as a result of input expansion). Otherwise, it will not be recognized as the end of the markdown string. As a corollary, the `\markdownEnd` string may not appear anywhere inside the markdown input.

Another limitation concerns spaces at the right end of an input line. In markdown, these are used to produce a forced line break. However, any such spaces are removed before the lines enter the input buffer of T_EX [4, p. 46]. As a corollary, the `\markdownBegin` macro also ignores them.

The `\markdownBegin` and `\markdownEnd` macros will also consume the rest of the lines at which they appear. In the following example plain T_EX code, the characters `c`, `e`, and `f` will not appear in the output.

```
\input markdown
a
b \markdownBegin c
d
e \markdownEnd f
g
\bye
```

Note that you may also not nest the `\markdownBegin` and `\markdownEnd` macros.

The following example plain T_EX code showcases the usage of the `\markdownBegin` and `\markdownEnd` macros:

```
\input markdown
\markdownBegin
_Hello_ **world** ...
\markdownEnd
\bye
```

The `\markdownInput` macro accepts a single parameter containing the filename of a markdown document and expands to the result of the conversion of the input markdown document to plain T_EX.

```
129 \let\markdownInput\relax
```

This macro is not subject to the abovelisted limitations of the `\markdownBegin` and `\markdownEnd` macros.

The following example plain T_EX code showcases the usage of the `\markdownInput` macro:

```
\input markdown
\markdownInput{hello.md}
\bye
```

2.2.2 Options

The plain T_EX options are represented by T_EX commands. Some of them map directly to the options recognized by the Lua interface (see Section 2.1.2), while some of them are specific to the plain T_EX interface.

2.2.2.1 File and Directory Names The `\markdownOptionHelperScriptFileName` macro sets the filename of the helper Lua script file that is created during the conversion from markdown to plain T_EX in T_EX engines without the `\directlua` primitive. It defaults to `\jobname.markdown.lua`, where `\jobname` is the base name of the document being typeset.

The expansion of this macro must not contain quotation marks (") or backslash symbols (\). Mind that T_EX engines tend to put quotation marks around `\jobname`, when it contains spaces.

```
130 \def\markdownOptionHelperScriptFileName{\jobname.markdown.lua}%
```

The `\markdownOptionInputTempFileName` macro sets the filename of the temporary input file that is created during the conversion from markdown to plain T_EX in `\markdownMode` other than 2. It defaults to `\jobname.markdown.out`. The same limitations as in the case of the `\markdownOptionHelperScriptFileName` macro apply here.

```
131 \def\markdownOptionInputTempFileName{\jobname.markdown.in}%
```

The `\markdownOptionOutputTempFileName` macro sets the filename of the temporary output file that is created during the conversion from markdown to plain T_EX in `\markdownMode` other than 2. It defaults to `\jobname.markdown.out`. The same limitations apply here as in the case of the `\markdownOptionHelperScriptFileName` macro.

```
132 \def\markdownOptionOutputTempFileName{\jobname.markdown.out}%
```

The `\markdownOptionErrorTempFileName` macro sets the filename of the temporary output file that is created when a Lua error is encountered during the conversion from markdown to plain TeX in `\markdownMode` other than 2. It defaults to `\jobname.markdown.err`. The same limitations apply here as in the case of the `\markdownOptionHelperScriptFileName` macro.

```
133 \def\markdownOptionErrorTempFileName{\jobname.markdown.err}%
```

The `\markdownOptionOutputDir` macro sets the path to the directory that will contain the cache files produced by the Lua implementation and also the auxiliary files produced by the plain TeX implementation. The option defaults to `..`.

The path must be set to the same value as the `-output-directory` option of your TeX engine for the package to function correctly. We need this macro to make the Lua implementation aware where it should store the helper files. The same limitations apply here as in the case of the `\markdownOptionHelperScriptFileName` macro.

```
134 \def\markdownOptionOutputDir{.}%
```

The `\markdownOptionCacheDir` macro corresponds to the Lua interface `cacheDir` option that sets the path to the directory that will contain the produced cache files. The option defaults to `_markdown_\jobname`, which is a similar naming scheme to the one used by the minted L^AT_EX package. The same limitations apply here as in the case of the `\markdownOptionHelperScriptFileName` macro.

```
135 \def\markdownOptionCacheDir{\markdownOptionOutputDir/_markdown_\jobname}%
```

2.2.2.2 Lua Interface Options The following macros map directly to the options recognized by the Lua interface (see Section 2.1.2) and are not processed by the plain TeX implementation, only passed along to Lua. They are undefined, which makes them fall back to the default values provided by the Lua interface.

For the macros that correspond to the non-boolean options recognized by the Lua interface, the same limitations apply here in the case of the `\markdownOptionHelperScriptFileName` macro.

```
136 \let\markdownOptionBlankBeforeBlockquote\undefined
```

```
137 \let\markdownOptionBlankBeforeCodeFence\undefined
```

```
138 \let\markdownOptionBlankBeforeHeading\undefined
```

```
139 \let\markdownOptionBreakableBlockquotes\undefined
```

```
140 \let\markdownOptionCitations\undefined
```

```
141 \let\markdownOptionCitationNbsps\undefined
```

```
142 \let\markdownOptionContentBlocks\undefined
```

```
143 \let\markdownOptionContentBlocksLanguageMap\undefined
```

```
144 \let\markdownOptionDefinitionLists\undefined
```

```
145 \let\markdownOptionFootnotes\undefined
```

```
146 \let\markdownOptionFencedCode\undefined
```

```
147 \let\markdownOptionHashEnumerators\undefined
```

```
148 \let\markdownOptionHeaderAttributes\undefined
```

```
149 \let\markdownOptionHtml\undefined
```

```

150 \let\markdownOptionHybrid\undefined
151 \let\markdownOptionInlineFootnotes\undefined
152 \let\markdownOptionPipeTables\undefined
153 \let\markdownOptionPreserveTabs\undefined
154 \let\markdownOptionShiftHeadings\undefined
155 \let\markdownOptionSlice\undefined
156 \let\markdownOptionSmartEllipses\undefined
157 \let\markdownOptionStartNumber\undefined
158 \let\markdownOptionTableCaptions\undefined
159 \let\markdownOptionTightLists\undefined

```

2.2.2.3 Miscellaneous Options The `\markdownOptionStripPercentSigns` macro controls whether a percent sign (%) at the beginning of a line will be discarded when buffering Markdown input (see Section 3.2.4) or not. Notably, this enables the use of markdown when writing T_EX package documentation using the Doc L^AT_EX package [5] or similar. The recognized values of the macro are `true` (discard) and `false` (retain).

```

160 \def\markdownOptionStripPercentSigns{false}%

```

The `\markdownIfOption{<name>}` macro is provided for testing, whether the value of `\markdownOption{<name>}` is `true` or `false`.

```

161 \def\markdownIfOption#1{%
162   \def\next##1##2##3##4##5{%
163     \expandafter\def\expandafter\next\expandafter{%
164       \csname iffalse\endcsname}%
165     \if##1t\if##2r\if##3u\if##4e
166     \expandafter\def\expandafter\next\expandafter{%
167       \csname iftrue\endcsname}%
168     \fi\fi\fi\fi
169     \next}%
170 \expandafter\expandafter\expandafter\next
171   \csname markdownOption#1\endcsname\relax\relax\relax\relax\relax}

```

2.2.3 Token Renderers

The following T_EX macros may occur inside the output of the converter functions exposed by the Lua interface (see Section 2.1.1) and represent the parsed markdown tokens. These macros are intended to be redefined by the user who is typesetting a document. By default, they point to the corresponding prototypes (see Section 2.2.4).

2.2.3.1 Interblock Separator Renderer The `\markdownRendererInterblockSeparator` macro represents a separator between two markdown block elements. The macro receives no arguments.

```

172 \def\markdownRendererInterblockSeparator{%
173   \markdownRendererInterblockSeparatorPrototype}%

```

2.2.3.2 Line Break Renderer The `\markdownRendererLineBreak` macro represents a forced line break. The macro receives no arguments.

```
174 \def\markdownRendererLineBreak{%
175   \markdownRendererLineBreakPrototype}%
```

2.2.3.3 Ellipsis Renderer The `\markdownRendererEllipsis` macro replaces any occurrence of ASCII ellipses in the input text. This macro will only be produced, when the `smartEllipses` option is `true`. The macro receives no arguments.

```
176 \def\markdownRendererEllipsis{%
177   \markdownRendererEllipsisPrototype}%
```

2.2.3.4 Non-Breaking Space Renderer The `\markdownRendererNbsp` macro represents a non-breaking space.

```
178 \def\markdownRendererNbsp{%
179   \markdownRendererNbspPrototype}%
```

2.2.3.5 Special Character Renderers The following macros replace any special plain \TeX characters, including the active pipe character (`|`) of `Con \TeX t`, in the input text. These macros will only be produced, when the `hybrid` option is `false`.

```
180 \def\markdownRendererLeftBrace{%
181   \markdownRendererLeftBracePrototype}%
182 \def\markdownRendererRightBrace{%
183   \markdownRendererRightBracePrototype}%
184 \def\markdownRendererDollarSign{%
185   \markdownRendererDollarSignPrototype}%
186 \def\markdownRendererPercentSign{%
187   \markdownRendererPercentSignPrototype}%
188 \def\markdownRendererAmpersand{%
189   \markdownRendererAmpersandPrototype}%
190 \def\markdownRendererUnderscore{%
191   \markdownRendererUnderscorePrototype}%
192 \def\markdownRendererHash{%
193   \markdownRendererHashPrototype}%
194 \def\markdownRendererCircumflex{%
195   \markdownRendererCircumflexPrototype}%
196 \def\markdownRendererBackslash{%
197   \markdownRendererBackslashPrototype}%
198 \def\markdownRendererTilde{%
199   \markdownRendererTildePrototype}%
200 \def\markdownRendererPipe{%
201   \markdownRendererPipePrototype}%
```

2.2.3.6 Code Span Renderer The `\markdownRendererCodeSpan` macro represents inlined code span in the input text. It receives a single argument that corresponds to the inlined code span.

```
202 \def\markdownRendererCodeSpan{%  
203   \markdownRendererCodeSpanPrototype}%
```

2.2.3.7 Link Renderer The `\markdownRendererLink` macro represents a hyperlink. It receives four arguments: the label, the fully escaped URI that can be directly typeset, the raw URI that can be used outside typesetting, and the title of the link.

```
204 \def\markdownRendererLink{%  
205   \markdownRendererLinkPrototype}%
```

2.2.3.8 Image Renderer The `\markdownRendererImage` macro represents an image. It receives four arguments: the label, the fully escaped URI that can be directly typeset, the raw URI that can be used outside typesetting, and the title of the link.

```
206 \def\markdownRendererImage{%  
207   \markdownRendererImagePrototype}%
```

2.2.3.9 Content Block Renderers The `\markdownRendererContentBlock` macro represents an iA Writer content block. It receives four arguments: the local file or online image filename extension cast to the lower case, the fully escaped URI that can be directly typeset, the raw URI that can be used outside typesetting, and the title of the content block.

```
208 \def\markdownRendererContentBlock{%  
209   \markdownRendererContentBlockPrototype}%
```

The `\markdownRendererContentBlockOnlineImage` macro represents an iA Writer online image content block. The macro receives the same arguments as `\markdownRendererContentBlock`.

```
210 \def\markdownRendererContentBlockOnlineImage{%  
211   \markdownRendererContentBlockOnlineImagePrototype}%
```

The `\markdownRendererContentBlockCode` macro represents an iA Writer content block that was recognized as a file in a known programming language by its filename extension s . If any `markdown-languages.json` file found by `kpathsea`⁶ contains a record (k, v) , then a non-online-image content block with the filename extension $s, s:\text{lower}() = k$ is considered to be in a known programming language v . The macro receives five arguments: the local file name extension s cast to the lower

⁶Local files take precedence. Filenames other than `markdown-languages.json` may be specified using the `contentBlocksLanguageMap` Lua option.

case, the language v , the fully escaped URI that can be directly typeset, the raw URI that can be used outside typesetting, and the title of the content block.

Note that you will need to place a `markdown-languages.json` file inside your working directory or inside your local \TeX directory structure. In this file, you will define a mapping between filename extensions and the language names recognized by your favorite syntax highlighter; there may exist other creative uses beside syntax highlighting. The `Languages.json` file provided by Sotkov [3] is a good starting point.

```
212 \def\markdownRendererContentBlockCode{%
213   \markdownRendererContentBlockCodePrototype}%
```

2.2.3.10 Bullet List Renderers The `\markdownRendererU1Begin` macro represents the beginning of a bulleted list that contains an item with several paragraphs of text (the list is not tight). The macro receives no arguments.

```
214 \def\markdownRendererU1Begin{%
215   \markdownRendererU1BeginPrototype}%
```

The `\markdownRendererU1BeginTight` macro represents the beginning of a bulleted list that contains no item with several paragraphs of text (the list is tight). This macro will only be produced, when the `tightLists` option is `false`. The macro receives no arguments.

```
216 \def\markdownRendererU1BeginTight{%
217   \markdownRendererU1BeginTightPrototype}%
```

The `\markdownRendererU1Item` macro represents an item in a bulleted list. The macro receives no arguments.

```
218 \def\markdownRendererU1Item{%
219   \markdownRendererU1ItemPrototype}%
```

The `\markdownRendererU1ItemEnd` macro represents the end of an item in a bulleted list. The macro receives no arguments.

```
220 \def\markdownRendererU1ItemEnd{%
221   \markdownRendererU1ItemEndPrototype}%
```

The `\markdownRendererU1End` macro represents the end of a bulleted list that contains an item with several paragraphs of text (the list is not tight). The macro receives no arguments.

```
222 \def\markdownRendererU1End{%
223   \markdownRendererU1EndPrototype}%
```

The `\markdownRendererU1EndTight` macro represents the end of a bulleted list that contains no item with several paragraphs of text (the list is tight). This macro

will only be produced, when the `tightLists` option is `false`. The macro receives no arguments.

```
224 \def\markdownRendererUListEndTight{%  
225   \markdownRendererUListEndTightPrototype}%
```

2.2.3.11 Ordered List Renderers The `\markdownRendererOListBegin` macro represents the beginning of an ordered list that contains an item with several paragraphs of text (the list is not tight). The macro receives no arguments.

```
226 \def\markdownRendererOListBegin{%  
227   \markdownRendererOListBeginPrototype}%
```

The `\markdownRendererOListBeginTight` macro represents the beginning of an ordered list that contains no item with several paragraphs of text (the list is tight). This macro will only be produced, when the `tightLists` option is `false`. The macro receives no arguments.

```
228 \def\markdownRendererOListBeginTight{%  
229   \markdownRendererOListBeginTightPrototype}%
```

The `\markdownRendererOListItem` macro represents an item in an ordered list. This macro will only be produced, when the `startNumber` option is `false`. The macro receives no arguments.

```
230 \def\markdownRendererOListItem{%  
231   \markdownRendererOListItemPrototype}%
```

The `\markdownRendererOListItemEnd` macro represents the end of an item in an ordered list. The macro receives no arguments.

```
232 \def\markdownRendererOListItemEnd{%  
233   \markdownRendererOListItemEndPrototype}%
```

The `\markdownRendererOListItemWithNumber` macro represents an item in an ordered list. This macro will only be produced, when the `startNumber` option is `true`. The macro receives no arguments.

```
234 \def\markdownRendererOListItemWithNumber{%  
235   \markdownRendererOListItemWithNumberPrototype}%
```

The `\markdownRendererOListEnd` macro represents the end of an ordered list that contains an item with several paragraphs of text (the list is not tight). The macro receives no arguments.

```
236 \def\markdownRendererOListEnd{%  
237   \markdownRendererOListEndPrototype}%
```

The `\markdownRendererOListEndTight` macro represents the end of an ordered list that contains no item with several paragraphs of text (the list is tight). This macro

will only be produced, when the `tightLists` option is `false`. The macro receives no arguments.

```
238 \def\markdownRendererDlEndTight{%  
239 \markdownRendererDlEndTightPrototype}%
```

2.2.3.12 Definition List Renderers The following macros are only produced, when the `definitionLists` option is `true`.

The `\markdownRendererDlBegin` macro represents the beginning of a definition list that contains an item with several paragraphs of text (the list is not tight). The macro receives no arguments.

```
240 \def\markdownRendererDlBegin{%  
241 \markdownRendererDlBeginPrototype}%
```

The `\markdownRendererDlBeginTight` macro represents the beginning of a definition list that contains an item with several paragraphs of text (the list is not tight). This macro will only be produced, when the `tightLists` option is `false`. The macro receives no arguments.

```
242 \def\markdownRendererDlBeginTight{%  
243 \markdownRendererDlBeginTightPrototype}%
```

The `\markdownRendererDlItem` macro represents a term in a definition list. The macro receives a single argument that corresponds to the term being defined.

```
244 \def\markdownRendererDlItem{%  
245 \markdownRendererDlItemPrototype}%
```

The `\markdownRendererDlItemEnd` macro represents the end of a list of definitions for a single term.

```
246 \def\markdownRendererDlItemEnd{%  
247 \markdownRendererDlItemEndPrototype}%
```

The `\markdownRendererDlDefinitionBegin` macro represents the beginning of a definition in a definition list. There can be several definitions for a single term.

```
248 \def\markdownRendererDlDefinitionBegin{%  
249 \markdownRendererDlDefinitionBeginPrototype}%
```

The `\markdownRendererDlDefinitionEnd` macro represents the end of a definition in a definition list. There can be several definitions for a single term.

```
250 \def\markdownRendererDlDefinitionEnd{%  
251 \markdownRendererDlDefinitionEndPrototype}%
```

The `\markdownRendererDlEnd` macro represents the end of a definition list that contains an item with several paragraphs of text (the list is not tight). The macro receives no arguments.

```
252 \def\markdownRendererDlEnd{%  
253 \markdownRendererDlEndPrototype}%
```

The `\markdownRendererDlEndTight` macro represents the end of a definition list that contains no item with several paragraphs of text (the list is tight). This macro will only be produced, when the `tightLists` option is `false`. The macro receives no arguments.

```
254 \def\markdownRendererDlEndTight{%
255   \markdownRendererDlEndTightPrototype}%
```

2.2.3.13 Emphasis Renderers The `\markdownRendererEmphasis` macro represents an emphasized span of text. The macro receives a single argument that corresponds to the emphasized span of text.

```
256 \def\markdownRendererEmphasis{%
257   \markdownRendererEmphasisPrototype}%
```

The `\markdownRendererStrongEmphasis` macro represents a strongly emphasized span of text. The macro receives a single argument that corresponds to the emphasized span of text.

```
258 \def\markdownRendererStrongEmphasis{%
259   \markdownRendererStrongEmphasisPrototype}%
```

2.2.3.14 Block Quote Renderers The `\markdownRendererBlockQuoteBegin` macro represents the beginning of a block quote. The macro receives no arguments.

```
260 \def\markdownRendererBlockQuoteBegin{%
261   \markdownRendererBlockQuoteBeginPrototype}%
```

The `\markdownRendererBlockQuoteEnd` macro represents the end of a block quote. The macro receives no arguments.

```
262 \def\markdownRendererBlockQuoteEnd{%
263   \markdownRendererBlockQuoteEndPrototype}%
```

2.2.3.15 Code Block Renderers The `\markdownRendererInputVerbatim` macro represents a code block. The macro receives a single argument that corresponds to the filename of a file containing the code block contents.

```
264 \def\markdownRendererInputVerbatim{%
265   \markdownRendererInputVerbatimPrototype}%
```

The `\markdownRendererInputFencedCode` macro represents a fenced code block. This macro will only be produced, when the `fencedCode` option is `true`. The macro receives two arguments that correspond to the filename of a file containing the code block contents and to the code fence infostring.

```
266 \def\markdownRendererInputFencedCode{%
267   \markdownRendererInputFencedCodePrototype}%
```

2.2.3.16 Heading Renderers The `\markdownRendererHeadingOne` macro represents a first level heading. The macro receives a single argument that corresponds to the heading text.

```
268 \def\markdownRendererHeadingOne{%  
269   \markdownRendererHeadingOnePrototype}%
```

The `\markdownRendererHeadingTwo` macro represents a second level heading. The macro receives a single argument that corresponds to the heading text.

```
270 \def\markdownRendererHeadingTwo{%  
271   \markdownRendererHeadingTwoPrototype}%
```

The `\markdownRendererHeadingThree` macro represents a third level heading. The macro receives a single argument that corresponds to the heading text.

```
272 \def\markdownRendererHeadingThree{%  
273   \markdownRendererHeadingThreePrototype}%
```

The `\markdownRendererHeadingFour` macro represents a fourth level heading. The macro receives a single argument that corresponds to the heading text.

```
274 \def\markdownRendererHeadingFour{%  
275   \markdownRendererHeadingFourPrototype}%
```

The `\markdownRendererHeadingFive` macro represents a fifth level heading. The macro receives a single argument that corresponds to the heading text.

```
276 \def\markdownRendererHeadingFive{%  
277   \markdownRendererHeadingFivePrototype}%
```

The `\markdownRendererHeadingSix` macro represents a sixth level heading. The macro receives a single argument that corresponds to the heading text.

```
278 \def\markdownRendererHeadingSix{%  
279   \markdownRendererHeadingSixPrototype}%
```

2.2.3.17 Horizontal Rule Renderer The `\markdownRendererHorizontalRule` macro represents a horizontal rule. The macro receives no arguments.

```
280 \def\markdownRendererHorizontalRule{%  
281   \markdownRendererHorizontalRulePrototype}%
```

2.2.3.18 Footnote Renderer The `\markdownRendererFootnote` macro represents a footnote. This macro will only be produced, when the `footnotes` option is `true`. The macro receives a single argument that corresponds to the footnote text.

```
282 \def\markdownRendererFootnote{%  
283   \markdownRendererFootnotePrototype}%
```

2.2.3.19 Parenthesized Citations Renderer The `\markdownRendererCite` macro represents a string of one or more parenthetical citations. This macro will only be produced, when the `citations` option is `true`. The macro receives the parameter `{⟨number of citations⟩}` followed by `⟨suppress author⟩{⟨prenote⟩}{⟨postnote⟩}{⟨name⟩}` repeated `⟨number of citations⟩` times. The `⟨suppress author⟩` parameter is either the token `-`, when the author’s name is to be suppressed, or `+` otherwise.

```
284 \def\markdownRendererCite{%
285   \markdownRendererCitePrototype}%
```

2.2.3.20 Text Citations Renderer The `\markdownRendererTextCite` macro represents a string of one or more text citations. This macro will only be produced, when the `citations` option is `true`. The macro receives parameters in the same format as the `\markdownRendererCite` macro.

```
286 \def\markdownRendererTextCite{%
287   \markdownRendererTextCitePrototype}%
```

2.2.3.21 Table Renderer The `\markdownRendererTable` macro represents a table. This macro will only be produced, when the `pipeTables` option is `true`. The macro receives the parameters `{⟨caption⟩}{⟨number of rows⟩}{⟨number of columns⟩}` followed by `{⟨alignments⟩}` and then by `{⟨row⟩}` repeated `⟨number of rows⟩` times, where `⟨row⟩` is `{⟨column⟩}` repeated `⟨number of columns⟩` times, `⟨alignments⟩` is `⟨alignment⟩` repeated `⟨number of columns⟩` times, and `⟨alignment⟩` is one of the following:

- `d` – The corresponding column has an unspecified (default) alignment.
- `l` – The corresponding column is left-aligned.
- `c` – The corresponding column is centered.
- `r` – The corresponding column is right-aligned.

```
288 \def\markdownRendererTable{%
289   \markdownRendererTablePrototype}%
```

2.2.4 Token Renderer Prototypes

The following T_EX macros provide definitions for the token renderers (see Section 2.2.3) that have not been redefined by the user. These macros are intended to be redefined by macro package authors who wish to provide sensible default token renderers. They are also redefined by the L^AT_EX and ConT_EXt implementations (see sections 3.3 and 3.4).

```
290 \def\markdownRendererInterblockSeparatorPrototype{%
291 \def\markdownRendererLineBreakPrototype{%
292 \def\markdownRendererEllipsisPrototype{%
293 \def\markdownRendererNbspPrototype{%
```

```

294 \def\markdownRendererLeftBracePrototype{}%
295 \def\markdownRendererRightBracePrototype{}%
296 \def\markdownRendererDollarSignPrototype{}%
297 \def\markdownRendererPercentSignPrototype{}%
298 \def\markdownRendererAmpersandPrototype{}%
299 \def\markdownRendererUnderscorePrototype{}%
300 \def\markdownRendererHashPrototype{}%
301 \def\markdownRendererCircumflexPrototype{}%
302 \def\markdownRendererBackslashPrototype{}%
303 \def\markdownRendererTildePrototype{}%
304 \def\markdownRendererPipePrototype{}%
305 \def\markdownRendererCodeSpanPrototype#1{}%
306 \def\markdownRendererLinkPrototype#1#2#3#4{}%
307 \def\markdownRendererImagePrototype#1#2#3#4{}%
308 \def\markdownRendererContentBlockPrototype#1#2#3#4{}%
309 \def\markdownRendererContentBlockOnlineImagePrototype#1#2#3#4{}%
310 \def\markdownRendererContentBlockCodePrototype#1#2#3#4#5{}%
311 \def\markdownRendererUlBeginPrototype{}%
312 \def\markdownRendererUlBeginTightPrototype{}%
313 \def\markdownRendererUlItemPrototype{}%
314 \def\markdownRendererUlItemEndPrototype{}%
315 \def\markdownRendererUlEndPrototype{}%
316 \def\markdownRendererUlEndTightPrototype{}%
317 \def\markdownRendererOlBeginPrototype{}%
318 \def\markdownRendererOlBeginTightPrototype{}%
319 \def\markdownRendererOlItemPrototype{}%
320 \def\markdownRendererOlItemWithNumberPrototype#1{}%
321 \def\markdownRendererOlItemEndPrototype{}%
322 \def\markdownRendererOlEndPrototype{}%
323 \def\markdownRendererOlEndTightPrototype{}%
324 \def\markdownRendererDlBeginPrototype{}%
325 \def\markdownRendererDlBeginTightPrototype{}%
326 \def\markdownRendererDlItemPrototype#1{}%
327 \def\markdownRendererDlItemEndPrototype{}%
328 \def\markdownRendererDlDefinitionBeginPrototype{}%
329 \def\markdownRendererDlDefinitionEndPrototype{}%
330 \def\markdownRendererDlEndPrototype{}%
331 \def\markdownRendererDlEndTightPrototype{}%
332 \def\markdownRendererEmphasisPrototype#1{}%
333 \def\markdownRendererStrongEmphasisPrototype#1{}%
334 \def\markdownRendererBlockQuoteBeginPrototype{}%
335 \def\markdownRendererBlockQuoteEndPrototype{}%
336 \def\markdownRendererInputVerbatimPrototype#1{}%
337 \def\markdownRendererInputFencedCodePrototype#1#2{}%
338 \def\markdownRendererHeadingOnePrototype#1{}%
339 \def\markdownRendererHeadingTwoPrototype#1{}%
340 \def\markdownRendererHeadingThreePrototype#1{}%

```

```

341 \def\markdownRendererHeadingFourPrototype#1{}%
342 \def\markdownRendererHeadingFivePrototype#1{}%
343 \def\markdownRendererHeadingSixPrototype#1{}%
344 \def\markdownRendererHorizontalRulePrototype{}%
345 \def\markdownRendererFootnotePrototype#1{}%
346 \def\markdownRendererCitePrototype#1{}%
347 \def\markdownRendererTextCitePrototype#1{}%
348 \def\markdownRendererTablePrototype#1#2#3{}%

```

2.2.5 Logging Facilities

The `\markdownInfo`, `\markdownWarning`, and `\markdownError` macros perform logging for the Markdown package. Their first argument specifies the text of the info, warning, or error message.

```

349 \def\markdownInfo#1{}%
350 \def\markdownWarning#1{}%

```

The `\markdownError` macro receives a second argument that provides a help text.

```

351 \def\markdownError#1#2{}%

```

You may redefine these macros to redirect and process the info, warning, and error messages.

2.2.6 Miscellanea

The `\markdownMakeOther` macro is used by the package, when a \TeX engine that does not support direct Lua access is starting to buffer a text. The plain \TeX implementation changes the category code of plain \TeX special characters to *other*, but there may be other active characters that may break the output. This macro should temporarily change the category of these to *other*.

```

352 \let\markdownMakeOther\relax

```

The `\markdownReadAndConvert` macro implements the `\markdownBegin` macro. The first argument specifies the token sequence that will terminate the markdown input (`\markdownEnd` in the instance of the `\markdownBegin` macro) when the plain \TeX special characters have had their category changed to *other*. The second argument specifies the token sequence that will actually be inserted into the document, when the ending token sequence has been found.

```

353 \let\markdownReadAndConvert\relax
354 \begingroup

```

Locally swap the category code of the backslash symbol (`\`) with the pipe symbol (`|`). This is required in order that all the special symbols in the first argument of the `markdownReadAndConvert` macro have the category code *other*.

```

355 \catcode`\|=0\catcode`\=12%
356 |gdef|markdownBegin{%
357 |markdownReadAndConvert{\markdownEnd}%

```

```

358             {|markdownEnd}}}%
359 |endgroup

```

The macro is exposed in the interface, so that the user can create their own markdown environments. Due to the way the arguments are passed to Lua (see Section 3.2.6), the first argument may not contain the string `]]` (regardless of the category code of the bracket symbol (`]`)).

The `\markdownMode` macro specifies how the plain \TeX implementation interfaces with the Lua interface. The valid values and their meaning are as follows:

- `0` – Shell escape via the 18 output file stream
- `1` – Shell escape via the Lua `os.execute` method
- `2` – Direct Lua access

By defining the macro, the user can coerce the package to use a specific mode. If the user does not define the macro prior to loading the plain \TeX implementation, the correct value will be automatically detected. The outcome of changing the value of `\markdownMode` after the implementation has been loaded is undefined.

```

360 \ifx\markdownMode\undefined
361   \ifx\directlua\undefined
362     \def\markdownMode{0}%
363   \else
364     \def\markdownMode{2}%
365   \fi
366 \fi

```

The following macros are no longer a part of the plain \TeX interface and are only defined for backwards compatibility:

```

367 \def\markdownLuaRegisterIBCallback#1{\relax}%
368 \def\markdownLuaUnregisterIBCallback#1{\relax}%

```

2.3 \LaTeX Interface

The \LaTeX interface provides \LaTeX environments for the typesetting of markdown input from within \LaTeX , facilities for setting Lua interface options (see Section 2.1.2) used during the conversion from markdown to plain \TeX , and facilities for changing the way markdown tokens are rendered. The rest of the interface is inherited from the plain \TeX interface (see Section 2.2).

The \LaTeX interface is implemented by the `markdown.sty` file, which can be loaded from the \LaTeX document preamble as follows:

```
\usepackage[options]{markdown}
```

where `options` are the \LaTeX interface options (see Section 2.3.2). Note that `options` inside the `\usepackage` macro may not set the `markdownRenderers` (see Section 2.3.2.2) and `markdownRendererPrototypes` (see Section 2.3.2.3) keys. This limitation is due to the way $\LaTeX 2_{\epsilon}$ parses package options.

2.3.1 Typesetting Markdown

The interface exposes the `markdown` and `markdown*` \LaTeX environments, and redefines the `\markdownInput` command.

The `markdown` and `markdown*` \LaTeX environments are used to typeset markdown document fragments. The starred version of the `markdown` environment accepts \LaTeX interface options (see Section 2.3.2) as its only argument. These options will only influence this markdown document fragment.

```
369 \newenvironment{markdown}\relax\relax
370 \newenvironment{markdown*}[1]\relax\relax
```

You may prepend your own code to the `\markdown` macro and append your own code to the `\endmarkdown` macro to produce special effects before and after the `markdown` \LaTeX environment (and likewise for the starred version).

Note that the `markdown` and `markdown*` \LaTeX environments are subject to the same limitations as the `\markdownBegin` and `\markdownEnd` macros exposed by the plain \TeX interface.

The following example \LaTeX code showcases the usage of the `markdown` and `markdown*` environments:

```
\documentclass{article}          \documentclass{article}
\usepackage{markdown}           \usepackage{markdown}
\begin{document}                \begin{document}
% ...                            % ...
\begin{markdown}                \begin{markdown*}{smartEllipses}
_Hello_ **world** ...           _Hello_ **world** ...
\end{markdown}                  \end{markdown*}
% ...                            % ...
\end{document}                  \end{document}
```

The `\markdownInput` macro accepts a single mandatory parameter containing the filename of a markdown document and expands to the result of the conversion of the input markdown document to plain \TeX . Unlike the `\markdownInput` macro provided by the plain \TeX interface, this macro also accepts \LaTeX interface options (see Section 2.3.2) as its optional argument. These options will only influence this markdown document.

The following example \LaTeX code showcases the usage of the `\markdownInput` macro:

```
\documentclass{article}
\usepackage{markdown}
\begin{document}
% ...
```

```
\markdownInput[smartEllipses]{hello.md}
% ...
\end{document}
```

2.3.2 Options

The \LaTeX options are represented by a comma-delimited list of $\langle key \rangle = \langle value \rangle$ pairs. For boolean options, the $= \langle value \rangle$ part is optional, and $\langle key \rangle$ will be interpreted as $\langle key \rangle = \text{true}$.

The \LaTeX options map directly to the options recognized by the plain \TeX interface (see Section 2.2.2) and to the markdown token renderers and their prototypes recognized by the plain \TeX interface (see Sections 2.2.3 and 2.2.4).

The \LaTeX options may be specified when loading the \LaTeX package (see Section 2.3), when using the `markdown*` \LaTeX environment, or via the `\markdownSetup` macro. The `\markdownSetup` macro receives the options to set up as its only argument.

```
371 \newcommand\markdownSetup[1]{%
372   \setkeys{markdownOptions}{#1}}%
```

2.3.2.1 Plain \TeX Interface Options The following options map directly to the option macros exposed by the plain \TeX interface (see Section 2.2.2).

```
373 \define@key{markdownOptions}{helperScriptFileName}{%
374   \def\markdownOptionHelperScriptFileName{#1}}%
375 \define@key{markdownOptions}{inputTempFileName}{%
376   \def\markdownOptionInputTempFileName{#1}}%
377 \define@key{markdownOptions}{outputTempFileName}{%
378   \def\markdownOptionOutputTempFileName{#1}}%
379 \define@key{markdownOptions}{errorTempFileName}{%
380   \def\markdownOptionErrorTempFileName{#1}}%
381 \define@key{markdownOptions}{cacheDir}{%
382   \def\markdownOptionCacheDir{#1}}%
383 \define@key{markdownOptions}{outputDir}{%
384   \def\markdownOptionOutputDir{#1}}%
385 \define@key{markdownOptions}{blankBeforeBlockquote}[true]{%
386   \def\markdownOptionBlankBeforeBlockquote{#1}}%
387 \define@key{markdownOptions}{blankBeforeCodeFence}[true]{%
388   \def\markdownOptionBlankBeforeCodeFence{#1}}%
389 \define@key{markdownOptions}{blankBeforeHeading}[true]{%
390   \def\markdownOptionBlankBeforeHeading{#1}}%
391 \define@key{markdownOptions}{breakableBlockquotes}[true]{%
392   \def\markdownOptionBreakableBlockquotes{#1}}%
393 \define@key{markdownOptions}{citations}[true]{%
394   \def\markdownOptionCitations{#1}}%
395 \define@key{markdownOptions}{citationNbsps}[true]{%
```

```

396 \def\markdownOptionCitationNbsps{#1}}%
397 \define@key{markdownOptions}{contentBlocks}[true]{%
398 \def\markdownOptionContentBlocks{#1}}%
399 \define@key{markdownOptions}{codeSpans}[true]{%
400 \def\markdownOptionCodeSpans{#1}}%
401 \define@key{markdownOptions}{contentBlocksLanguageMap}{%
402 \def\markdownOptionContentBlocksLanguageMap{#1}}%
403 \define@key{markdownOptions}{definitionLists}[true]{%
404 \def\markdownOptionDefinitionLists{#1}}%
405 \define@key{markdownOptions}{footnotes}[true]{%
406 \def\markdownOptionFootnotes{#1}}%
407 \define@key{markdownOptions}{fencedCode}[true]{%
408 \def\markdownOptionFencedCode{#1}}%
409 \define@key{markdownOptions}{hashEnumerators}[true]{%
410 \def\markdownOptionHashEnumerators{#1}}%
411 \define@key{markdownOptions}{headerAttributes}[true]{%
412 \def\markdownOptionHeaderAttributes{#1}}%
413 \define@key{markdownOptions}{html}[true]{%
414 \def\markdownOptionHtml{#1}}%
415 \define@key{markdownOptions}{hybrid}[true]{%
416 \def\markdownOptionHybrid{#1}}%
417 \define@key{markdownOptions}{inlineFootnotes}[true]{%
418 \def\markdownOptionInlineFootnotes{#1}}%
419 \define@key{markdownOptions}{pipeTables}[true]{%
420 \def\markdownOptionPipeTables{#1}}%
421 \define@key{markdownOptions}{preserveTabs}[true]{%
422 \def\markdownOptionPreserveTabs{#1}}%
423 \define@key{markdownOptions}{smartEllipses}[true]{%
424 \def\markdownOptionSmartEllipses{#1}}%
425 \define@key{markdownOptions}{shiftHeadings}{%
426 \def\markdownOptionShiftHeadings{#1}}%
427 \define@key{markdownOptions}{slice}{%
428 \def\markdownOptionSlice{#1}}%
429 \define@key{markdownOptions}{startNumber}[true]{%
430 \def\markdownOptionStartNumber{#1}}%
431 \define@key{markdownOptions}{tableCaptions}[true]{%
432 \def\markdownOptionTableCaptions{#1}}%
433 \define@key{markdownOptions}{tightLists}[true]{%
434 \def\markdownOptionTightLists{#1}}%
435 \define@key{markdownOptions}{underscores}[true]{%
436 \def\markdownOptionUnderscores{#1}}%
437 \define@key{markdownOptions}{stripPercentSigns}[true]{%
438 \def\markdownOptionStripPercentSigns{#1}}%

```

The following example L^AT_EX code showcases a possible configuration of plain T_EX interface options `\markdownOptionHybrid`, `\markdownOptionSmartEllipses`, and `\markdownOptionCacheDir`.

```

\markdownSetup{
  hybrid,
  smartEllipses,
  cacheDir = /tmp,
}

```

2.3.2.2 Plain T_EX Markdown Token Renderers

The L^AT_EX interface recognizes an option with the `renderers` key, whose value must be a list of options that map directly to the markdown token renderer macros exposed by the plain T_EX interface (see Section 2.2.3).

```

439 \define@key{markdownRenderers}{interblockSeparator}{%
440   \renewcommand\markdownRendererInterblockSeparator{#1}}%
441 \define@key{markdownRenderers}{lineBreak}{%
442   \renewcommand\markdownRendererLineBreak{#1}}%
443 \define@key{markdownRenderers}{ellipsis}{%
444   \renewcommand\markdownRendererEllipsis{#1}}%
445 \define@key{markdownRenderers}{nbsp}{%
446   \renewcommand\markdownRendererNbsp{#1}}%
447 \define@key{markdownRenderers}{leftBrace}{%
448   \renewcommand\markdownRendererLeftBrace{#1}}%
449 \define@key{markdownRenderers}{rightBrace}{%
450   \renewcommand\markdownRendererRightBrace{#1}}%
451 \define@key{markdownRenderers}{dollarSign}{%
452   \renewcommand\markdownRendererDollarSign{#1}}%
453 \define@key{markdownRenderers}{percentSign}{%
454   \renewcommand\markdownRendererPercentSign{#1}}%
455 \define@key{markdownRenderers}{ampersand}{%
456   \renewcommand\markdownRendererAmpersand{#1}}%
457 \define@key{markdownRenderers}{underscore}{%
458   \renewcommand\markdownRendererUnderscore{#1}}%
459 \define@key{markdownRenderers}{hash}{%
460   \renewcommand\markdownRendererHash{#1}}%
461 \define@key{markdownRenderers}{circumflex}{%
462   \renewcommand\markdownRendererCircumflex{#1}}%
463 \define@key{markdownRenderers}{backslash}{%
464   \renewcommand\markdownRendererBackslash{#1}}%
465 \define@key{markdownRenderers}{tilde}{%
466   \renewcommand\markdownRendererTilde{#1}}%
467 \define@key{markdownRenderers}{pipe}{%
468   \renewcommand\markdownRendererPipe{#1}}%
469 \define@key{markdownRenderers}{codeSpan}{%
470   \renewcommand\markdownRendererCodeSpan[1]{#1}}%
471 \define@key{markdownRenderers}{link}{%
472   \renewcommand\markdownRendererLink[4]{#1}}%

```

```

473 \define@key{markdownRenderers}{contentBlock}{%
474   \renewcommand\markdownRendererContentBlock[4]{#1}}%
475 \define@key{markdownRenderers}{contentBlockOnlineImage}{%
476   \renewcommand\markdownRendererContentBlockOnlineImage[4]{#1}}%
477 \define@key{markdownRenderers}{contentBlockCode}{%
478   \renewcommand\markdownRendererContentBlockCode[5]{#1}}%
479 \define@key{markdownRenderers}{image}{%
480   \renewcommand\markdownRendererImage[4]{#1}}%
481 \define@key{markdownRenderers}{ulBegin}{%
482   \renewcommand\markdownRendererUlBegin{#1}}%
483 \define@key{markdownRenderers}{ulBeginTight}{%
484   \renewcommand\markdownRendererUlBeginTight{#1}}%
485 \define@key{markdownRenderers}{ulItem}{%
486   \renewcommand\markdownRendererUlItem{#1}}%
487 \define@key{markdownRenderers}{ulItemEnd}{%
488   \renewcommand\markdownRendererUlItemEnd{#1}}%
489 \define@key{markdownRenderers}{ulEnd}{%
490   \renewcommand\markdownRendererUlEnd{#1}}%
491 \define@key{markdownRenderers}{ulEndTight}{%
492   \renewcommand\markdownRendererUlEndTight{#1}}%
493 \define@key{markdownRenderers}{olBegin}{%
494   \renewcommand\markdownRendererOlBegin{#1}}%
495 \define@key{markdownRenderers}{olBeginTight}{%
496   \renewcommand\markdownRendererOlBeginTight{#1}}%
497 \define@key{markdownRenderers}{olItem}{%
498   \renewcommand\markdownRendererOlItem{#1}}%
499 \define@key{markdownRenderers}{olItemWithNumber}{%
500   \renewcommand\markdownRendererOlItemWithNumber[1]{#1}}%
501 \define@key{markdownRenderers}{olItemEnd}{%
502   \renewcommand\markdownRendererOlItemEnd{#1}}%
503 \define@key{markdownRenderers}{olEnd}{%
504   \renewcommand\markdownRendererOlEnd{#1}}%
505 \define@key{markdownRenderers}{olEndTight}{%
506   \renewcommand\markdownRendererOlEndTight{#1}}%
507 \define@key{markdownRenderers}{dlBegin}{%
508   \renewcommand\markdownRendererDlBegin{#1}}%
509 \define@key{markdownRenderers}{dlBeginTight}{%
510   \renewcommand\markdownRendererDlBeginTight{#1}}%
511 \define@key{markdownRenderers}{dlItem}{%
512   \renewcommand\markdownRendererDlItem[1]{#1}}%
513 \define@key{markdownRenderers}{dlItemEnd}{%
514   \renewcommand\markdownRendererDlItemEnd{#1}}%
515 \define@key{markdownRenderers}{dlDefinitionBegin}{%
516   \renewcommand\markdownRendererDlDefinitionBegin{#1}}%
517 \define@key{markdownRenderers}{dlDefinitionEnd}{%
518   \renewcommand\markdownRendererDlDefinitionEnd{#1}}%
519 \define@key{markdownRenderers}{dlEnd}{%

```

```

520 \renewcommand\markdownRendererDlEnd{#1}}%
521 \define@key{markdownRenderers}{dlEndTight}{%
522 \renewcommand\markdownRendererDlEndTight{#1}}%
523 \define@key{markdownRenderers}{emphasis}{%
524 \renewcommand\markdownRendererEmphasis[1]{#1}}%
525 \define@key{markdownRenderers}{strongEmphasis}{%
526 \renewcommand\markdownRendererStrongEmphasis[1]{#1}}%
527 \define@key{markdownRenderers}{blockquoteBegin}{%
528 \renewcommand\markdownRendererBlockQuoteBegin{#1}}%
529 \define@key{markdownRenderers}{blockquoteEnd}{%
530 \renewcommand\markdownRendererBlockQuoteEnd{#1}}%
531 \define@key{markdownRenderers}{inputVerbatim}{%
532 \renewcommand\markdownRendererInputVerbatim[1]{#1}}%
533 \define@key{markdownRenderers}{inputFencedCode}{%
534 \renewcommand\markdownRendererInputFencedCode[2]{#1}}%
535 \define@key{markdownRenderers}{headingOne}{%
536 \renewcommand\markdownRendererHeadingOne[1]{#1}}%
537 \define@key{markdownRenderers}{headingTwo}{%
538 \renewcommand\markdownRendererHeadingTwo[1]{#1}}%
539 \define@key{markdownRenderers}{headingThree}{%
540 \renewcommand\markdownRendererHeadingThree[1]{#1}}%
541 \define@key{markdownRenderers}{headingFour}{%
542 \renewcommand\markdownRendererHeadingFour[1]{#1}}%
543 \define@key{markdownRenderers}{headingFive}{%
544 \renewcommand\markdownRendererHeadingFive[1]{#1}}%
545 \define@key{markdownRenderers}{headingSix}{%
546 \renewcommand\markdownRendererHeadingSix[1]{#1}}%
547 \define@key{markdownRenderers}{horizontalRule}{%
548 \renewcommand\markdownRendererHorizontalRule{#1}}%
549 \define@key{markdownRenderers}{footnote}{%
550 \renewcommand\markdownRendererFootnote[1]{#1}}%
551 \define@key{markdownRenderers}{cite}{%
552 \renewcommand\markdownRendererCite[1]{#1}}%
553 \define@key{markdownRenderers}{textCite}{%
554 \renewcommand\markdownRendererTextCite[1]{#1}}%
555 \define@key{markdownRenderers}{table}{%
556 \renewcommand\markdownRendererTable[3]{#1}}%

```

The following example L^AT_EX code showcases a possible configuration of the `\markdownRendererLink` and `\markdownRendererEmphasis` markdown token renderers.

```

\markdownSetup{
  renderers = {
    link = {#4}, % Render links as the link title.
    emphasis = {\emph{#1}}, % Render emphasized text via \emph`.

```

```
}  
}
```

2.3.2.3 Plain T_EX Markdown Token Renderer Prototypes The L^AT_EX interface recognizes an option with the `rendererPrototypes` key, whose value must be a list of options that map directly to the markdown token renderer prototype macros exposed by the plain T_EX interface (see Section 2.2.4).

```
557 \define@key{markdownRendererPrototypes}{interblockSeparator}{%  
558   \renewcommand\markdownRendererInterblockSeparatorPrototype{#1}}%  
559 \define@key{markdownRendererPrototypes}{lineBreak}{%  
560   \renewcommand\markdownRendererLineBreakPrototype{#1}}%  
561 \define@key{markdownRendererPrototypes}{ellipsis}{%  
562   \renewcommand\markdownRendererEllipsisPrototype{#1}}%  
563 \define@key{markdownRendererPrototypes}{nbsp}{%  
564   \renewcommand\markdownRendererNbspPrototype{#1}}%  
565 \define@key{markdownRendererPrototypes}{leftBrace}{%  
566   \renewcommand\markdownRendererLeftBracePrototype{#1}}%  
567 \define@key{markdownRendererPrototypes}{rightBrace}{%  
568   \renewcommand\markdownRendererRightBracePrototype{#1}}%  
569 \define@key{markdownRendererPrototypes}{dollarSign}{%  
570   \renewcommand\markdownRendererDollarSignPrototype{#1}}%  
571 \define@key{markdownRendererPrototypes}{percentSign}{%  
572   \renewcommand\markdownRendererPercentSignPrototype{#1}}%  
573 \define@key{markdownRendererPrototypes}{ampersand}{%  
574   \renewcommand\markdownRendererAmpersandPrototype{#1}}%  
575 \define@key{markdownRendererPrototypes}{underscore}{%  
576   \renewcommand\markdownRendererUnderscorePrototype{#1}}%  
577 \define@key{markdownRendererPrototypes}{hash}{%  
578   \renewcommand\markdownRendererHashPrototype{#1}}%  
579 \define@key{markdownRendererPrototypes}{circumflex}{%  
580   \renewcommand\markdownRendererCircumflexPrototype{#1}}%  
581 \define@key{markdownRendererPrototypes}{backslash}{%  
582   \renewcommand\markdownRendererBackslashPrototype{#1}}%  
583 \define@key{markdownRendererPrototypes}{tilde}{%  
584   \renewcommand\markdownRendererTildePrototype{#1}}%  
585 \define@key{markdownRendererPrototypes}{pipe}{%  
586   \renewcommand\markdownRendererPipePrototype{#1}}%  
587 \define@key{markdownRendererPrototypes}{codeSpan}{%  
588   \renewcommand\markdownRendererCodeSpanPrototype[1]{#1}}%  
589 \define@key{markdownRendererPrototypes}{link}{%  
590   \renewcommand\markdownRendererLinkPrototype[4]{#1}}%  
591 \define@key{markdownRendererPrototypes}{contentBlock}{%  
592   \renewcommand\markdownRendererContentBlockPrototype[4]{#1}}%  
593 \define@key{markdownRendererPrototypes}{contentBlockOnlineImage}{%  
594   \renewcommand\markdownRendererContentBlockOnlineImagePrototype[4]{#1}}%
```

```

595 \define@key{markdownRendererPrototypes}{contentBlockCode}{%
596   \renewcommand\markdownRendererContentBlockCodePrototype[5]{#1}}%
597 \define@key{markdownRendererPrototypes}{image}{%
598   \renewcommand\markdownRendererImagePrototype[4]{#1}}%
599 \define@key{markdownRendererPrototypes}{ulBegin}{%
600   \renewcommand\markdownRendererUlBeginPrototype{#1}}%
601 \define@key{markdownRendererPrototypes}{ulBeginTight}{%
602   \renewcommand\markdownRendererUlBeginTightPrototype{#1}}%
603 \define@key{markdownRendererPrototypes}{ulItem}{%
604   \renewcommand\markdownRendererUlItemPrototype{#1}}%
605 \define@key{markdownRendererPrototypes}{ulItemEnd}{%
606   \renewcommand\markdownRendererUlItemEndPrototype{#1}}%
607 \define@key{markdownRendererPrototypes}{ulEnd}{%
608   \renewcommand\markdownRendererUlEndPrototype{#1}}%
609 \define@key{markdownRendererPrototypes}{ulEndTight}{%
610   \renewcommand\markdownRendererUlEndTightPrototype{#1}}%
611 \define@key{markdownRendererPrototypes}{olBegin}{%
612   \renewcommand\markdownRendererOlBeginPrototype{#1}}%
613 \define@key{markdownRendererPrototypes}{olBeginTight}{%
614   \renewcommand\markdownRendererOlBeginTightPrototype{#1}}%
615 \define@key{markdownRendererPrototypes}{olItem}{%
616   \renewcommand\markdownRendererOlItemPrototype{#1}}%
617 \define@key{markdownRendererPrototypes}{olItemWithNumber}{%
618   \renewcommand\markdownRendererOlItemWithNumberPrototype[1]{#1}}%
619 \define@key{markdownRendererPrototypes}{olItemEnd}{%
620   \renewcommand\markdownRendererOlItemEndPrototype{#1}}%
621 \define@key{markdownRendererPrototypes}{olEnd}{%
622   \renewcommand\markdownRendererOlEndPrototype{#1}}%
623 \define@key{markdownRendererPrototypes}{olEndTight}{%
624   \renewcommand\markdownRendererOlEndTightPrototype{#1}}%
625 \define@key{markdownRendererPrototypes}{dlBegin}{%
626   \renewcommand\markdownRendererDlBeginPrototype{#1}}%
627 \define@key{markdownRendererPrototypes}{dlBeginTight}{%
628   \renewcommand\markdownRendererDlBeginTightPrototype{#1}}%
629 \define@key{markdownRendererPrototypes}{dlItem}{%
630   \renewcommand\markdownRendererDlItemPrototype[1]{#1}}%
631 \define@key{markdownRendererPrototypes}{dlItemEnd}{%
632   \renewcommand\markdownRendererDlItemEndPrototype{#1}}%
633 \define@key{markdownRendererPrototypes}{dlDefinitionBegin}{%
634   \renewcommand\markdownRendererDlDefinitionBeginPrototype{#1}}%
635 \define@key{markdownRendererPrototypes}{dlDefinitionEnd}{%
636   \renewcommand\markdownRendererDlDefinitionEndPrototype{#1}}%
637 \define@key{markdownRendererPrototypes}{dlEnd}{%
638   \renewcommand\markdownRendererDlEndPrototype{#1}}%
639 \define@key{markdownRendererPrototypes}{dlEndTight}{%
640   \renewcommand\markdownRendererDlEndTightPrototype{#1}}%
641 \define@key{markdownRendererPrototypes}{emphasis}{%

```



```

642 \renewcommand\markdownRendererEmphasisPrototype[1]{#1}%
643 \define@key{markdownRendererPrototypes}{strongEmphasis}{%
644 \renewcommand\markdownRendererStrongEmphasisPrototype[1]{#1}}%
645 \define@key{markdownRendererPrototypes}{blockQuoteBegin}{%
646 \renewcommand\markdownRendererBlockQuoteBeginPrototype{#1}}%
647 \define@key{markdownRendererPrototypes}{blockQuoteEnd}{%
648 \renewcommand\markdownRendererBlockQuoteEndPrototype{#1}}%
649 \define@key{markdownRendererPrototypes}{inputVerbatim}{%
650 \renewcommand\markdownRendererInputVerbatimPrototype[1]{#1}}%
651 \define@key{markdownRendererPrototypes}{inputFencedCode}{%
652 \renewcommand\markdownRendererInputFencedCodePrototype[2]{#1}}%
653 \define@key{markdownRendererPrototypes}{headingOne}{%
654 \renewcommand\markdownRendererHeadingOnePrototype[1]{#1}}%
655 \define@key{markdownRendererPrototypes}{headingTwo}{%
656 \renewcommand\markdownRendererHeadingTwoPrototype[1]{#1}}%
657 \define@key{markdownRendererPrototypes}{headingThree}{%
658 \renewcommand\markdownRendererHeadingThreePrototype[1]{#1}}%
659 \define@key{markdownRendererPrototypes}{headingFour}{%
660 \renewcommand\markdownRendererHeadingFourPrototype[1]{#1}}%
661 \define@key{markdownRendererPrototypes}{headingFive}{%
662 \renewcommand\markdownRendererHeadingFivePrototype[1]{#1}}%
663 \define@key{markdownRendererPrototypes}{headingSix}{%
664 \renewcommand\markdownRendererHeadingSixPrototype[1]{#1}}%
665 \define@key{markdownRendererPrototypes}{horizontalRule}{%
666 \renewcommand\markdownRendererHorizontalRulePrototype{#1}}%
667 \define@key{markdownRendererPrototypes}{footnote}{%
668 \renewcommand\markdownRendererFootnotePrototype[1]{#1}}%
669 \define@key{markdownRendererPrototypes}{cite}{%
670 \renewcommand\markdownRendererCitePrototype[1]{#1}}%
671 \define@key{markdownRendererPrototypes}{textCite}{%
672 \renewcommand\markdownRendererTextCitePrototype[1]{#1}}%
673 \define@key{markdownRendererPrototypes}{table}{%
674 \renewcommand\markdownRendererTablePrototype[3]{#1}}%

```

The following example \LaTeX code showcases a possible configuration of the `\markdownRendererImagePrototype` and `\markdownRendererCodeSpanPrototype` markdown token renderer prototypes.

```

\markdownSetup{
  rendererPrototypes = {
    image = {\includegraphics{#2}},
    codeSpan = {\texttt{#1}}, % Render inline code via \texttt.
  }
}

```

2.4 ConTeXt Interface

The ConTeXt interface provides a start-stop macro pair for the typesetting of markdown input from within ConTeXt. The rest of the interface is inherited from the plain TeX interface (see Section 2.2).

```
675 \writestatus{loading}{ConTeXt User Module / markdown}%
676 \unprotect
```

The ConTeXt interface is implemented by the `t-markdown.tex` ConTeXt module file that can be loaded as follows:

```
\usemodule[t][markdown]
```

It is expected that the special plain TeX characters have the expected category codes, when `\inputting` the file.

2.4.1 Typesetting Markdown

The interface exposes the `\startmarkdown` and `\stopmarkdown` macro pair for the typesetting of a markdown document fragment.

```
677 \let\startmarkdown\relax
678 \let\stopmarkdown\relax
```

You may prepend your own code to the `\startmarkdown` macro and redefine the `\stopmarkdown` macro to produce special effects before and after the markdown block.

Note that the `\startmarkdown` and `\stopmarkdown` macros are subject to the same limitations as the `\markdownBegin` and `\markdownEnd` macros exposed by the plain TeX interface.

The following example ConTeXt code showcases the usage of the `\startmarkdown` and `\stopmarkdown` macros:

```
\usemodule[t][markdown]
\starttext
\startmarkdown
_Hello_ **world** ...
\stopmarkdown
\stoptext
```

3 Implementation

This part of the documentation describes the implementation of the interfaces exposed by the package (see Section 2) and is aimed at the developers of the package, as well as the curious users.

3.1 Lua Implementation

The Lua implementation implements `writer` and `reader` objects that provide the conversion from markdown to plain $\text{T}_{\text{E}}\text{X}$.

The Lunamark Lua module implements writers for the conversion to various other formats, such as DocBook, Groff, or HTML. These were stripped from the module and the remaining markdown reader and plain $\text{T}_{\text{E}}\text{X}$ writer were hidden behind the converter functions exposed by the Lua interface (see Section 2.1).

```
679 local upper, gsub, format, length =
680   string.upper, string.gsub, string.format, string.len
681 local concat = table.concat
682 local P, R, S, V, C, Cg, Cb, Cmt, Cc, Ct, B, Cs, any =
683   lpeg.P, lpeg.R, lpeg.S, lpeg.V, lpeg.C, lpeg.Cg, lpeg.Cb,
684   lpeg.Cmt, lpeg.Cc, lpeg.Ct, lpeg.B, lpeg.Cs, lpeg.P(1)
```

3.1.1 Utility Functions

This section documents the utility functions used by the plain $\text{T}_{\text{E}}\text{X}$ writer and the markdown reader. These functions are encapsulated in the `util` object. The functions were originally located in the `lunamark/util.lua` file in the Lunamark Lua module.

```
685 local util = {}
```

The `util.err` method prints an error message `msg` and exits. If `exit_code` is provided, it specifies the exit code. Otherwise, the exit code will be 1.

```
686 function util.err(msg, exit_code)
687   io.stderr:write("markdown.lua: " .. msg .. "\n")
688   os.exit(exit_code or 1)
689 end
```

The `util.cache` method computes the digest of `string` and `salt`, adds the `suffix` and looks into the directory `dir`, whether a file with such a name exists. If it does not, it gets created with `transform(string)` as its content. The filename is then returned.

```
690 function util.cache(dir, string, salt, transform, suffix)
691   local digest = md5.sumhexa(string .. (salt or ""))
692   local name = util.pathname(dir, digest .. suffix)
693   local file = io.open(name, "r")
694   if file == nil then -- If no cache entry exists, then create a new one.
695     local file = assert(io.open(name, "w"))
696     local result = string
697     if transform ~= nil then
698       result = transform(result)
699     end
700     assert(file:write(result))
701     assert(file:close())
```

```

702 end
703 return name
704 end

```

The `util.table_copy` method creates a shallow copy of a table `t` and its metatable.

```

705 function util.table_copy(t)
706   local u = { }
707   for k, v in pairs(t) do u[k] = v end
708   return setmetatable(u, getmetatable(t))
709 end

```

The `util.expand_tabs_in_line` expands tabs in string `s`. If `tabstop` is specified, it is used as the tab stop width. Otherwise, the tab stop width of 4 characters is used. The method is a copy of the tab expansion algorithm from Ierusalimschy [6, Chapter 21].

```

710 function util.expand_tabs_in_line(s, tabstop)
711   local tab = tabstop or 4
712   local corr = 0
713   return (s:gsub("\t", function(p)
714     local sp = tab - (p - 1 + corr) % tab
715     corr = corr - 1 + sp
716     return string.rep(" ", sp)
717   end))
718 end

```

The `util.walk` method walks a rope `t`, applying a function `f` to each leaf element in order. A rope is an array whose elements may be ropes, strings, numbers, or functions. If a leaf element is a function, call it and get the return value before proceeding.

```

719 function util.walk(t, f)
720   local typ = type(t)
721   if typ == "string" then
722     f(t)
723   elseif typ == "table" then
724     local i = 1
725     local n
726     n = t[i]
727     while n do
728       util.walk(n, f)
729       i = i + 1
730       n = t[i]
731     end
732   elseif typ == "function" then
733     local ok, val = pcall(t)
734     if ok then
735       util.walk(val, f)
736     end
737   else

```

```

738     f(tostring(t))
739   end
740 end

```

The `util.flatten` method flattens an array `ary` that does not contain cycles and returns the result.

```

741 function util.flatten(ary)
742   local new = {}
743   for _,v in ipairs(ary) do
744     if type(v) == "table" then
745       for _,w in ipairs(util.flatten(v)) do
746         new[#new + 1] = w
747       end
748     else
749       new[#new + 1] = v
750     end
751   end
752   return new
753 end

```

The `util.rope_to_string` method converts a rope `rope` to a string and returns it. For the definition of a rope, see the definition of the `util.walk` method.

```

754 function util.rope_to_string(rope)
755   local buffer = {}
756   util.walk(rope, function(x) buffer[#buffer + 1] = x end)
757   return table.concat(buffer)
758 end

```

The `util.rope_last` method retrieves the last item in a rope. For the definition of a rope, see the definition of the `util.walk` method.

```

759 function util.rope_last(rope)
760   if #rope == 0 then
761     return nil
762   else
763     local l = rope[#rope]
764     if type(l) == "table" then
765       return util.rope_last(l)
766     else
767       return l
768     end
769   end
770 end

```

Given an array `ary` and a string `x`, the `util.intersperse` method returns an array `new`, such that `ary[i] == new[2*(i-1)+1]` and `new[2*i] == x` for all $1 \leq i \leq \#ary$.

```

771 function util.intersperse(ary, x)
772   local new = {}

```

```

773 local l = #ary
774 for i,v in ipairs(ary) do
775     local n = #new
776     new[n + 1] = v
777     if i ~= l then
778         new[n + 2] = x
779     end
780 end
781 return new
782 end

```

Given an array `ary` and a function `f`, the `util.map` method returns an array `new`, such that `new[i] == f(ary[i])` for all $1 \leq i \leq \#ary$.

```

783 function util.map(ary, f)
784     local new = {}
785     for i,v in ipairs(ary) do
786         new[i] = f(v)
787     end
788     return new
789 end

```

Given a table `char_escapes` mapping escapable characters to escaped strings and optionally a table `string_escapes` mapping escapable strings to escaped strings, the `util.escaper` method returns an escaper function that escapes all occurrences of escapable strings and characters (in this order).

The method uses LPeg, which is faster than the Lua `string.gsub` built-in method.

```

790 function util.escaper(char_escapes, string_escapes)

```

Build a string of escapable characters.

```

791 local char_escapes_list = ""
792 for i,_ in pairs(char_escapes) do
793     char_escapes_list = char_escapes_list .. i
794 end

```

Create an LPeg capture `escapable` that produces the escaped string corresponding to the matched escapable character.

```

795 local escapable = S(char_escapes_list) / char_escapes

```

If `string_escapes` is provided, turn `escapable` into the

$$\sum_{(k,v) \in \text{string_escapes}} P(k) / v + \text{escapable}$$

capture that replaces any occurrence of the string `k` with the string `v` for each $(k, v) \in \text{string_escapes}$. Note that the pattern summation is not commutative and its operands are inspected in the summation order during the matching. As a corollary, the strings always take precedence over the characters.

```

796 if string_escapes then

```

```

797     for k,v in pairs(string_escapes) do
798         escapable = P(k) / v + escapable
799     end
800 end

```

Create an LPeg capture `escape_string` that captures anything `escapable` does and matches any other unmatched characters.

```

801 local escape_string = Cs((escapable + any)^0)

```

Return a function that matches the input string `s` against the `escape_string` capture.

```

802 return function(s)
803     return lpeg.match(escape_string, s)
804 end
805 end

```

The `util.pathname` method produces a pathname out of a directory name `dir` and a filename `file` and returns it.

```

806 function util.pathname(dir, file)
807     if #dir == 0 then
808         return file
809     else
810         return dir .. "/" .. file
811     end
812 end

```

3.1.2 HTML Entities

This section documents the HTML entities recognized by the markdown reader. These functions are encapsulated in the `entities` object. The functions were originally located in the `lunamark/entities.lua` file in the Lunamark Lua module.

```

813 local entities = {}
814
815 local character_entities = {
816     ["Tab"] = 9,
817     ["NewLine"] = 10,
818     ["excl"] = 33,
819     ["quot"] = 34,
820     ["QUOT"] = 34,
821     ["num"] = 35,
822     ["dollar"] = 36,
823     ["percent"] = 37,
824     ["amp"] = 38,
825     ["AMP"] = 38,
826     ["apos"] = 39,
827     ["lpar"] = 40,
828     ["rpar"] = 41,

```

829 ["ast"] = 42,
830 ["midast"] = 42,
831 ["plus"] = 43,
832 ["comma"] = 44,
833 ["period"] = 46,
834 ["sol"] = 47,
835 ["colon"] = 58,
836 ["semi"] = 59,
837 ["lt"] = 60,
838 ["LT"] = 60,
839 ["equals"] = 61,
840 ["gt"] = 62,
841 ["GT"] = 62,
842 ["quest"] = 63,
843 ["commat"] = 64,
844 ["lsqb"] = 91,
845 ["lbrack"] = 91,
846 ["bsol"] = 92,
847 ["rsqb"] = 93,
848 ["rbrack"] = 93,
849 ["Hat"] = 94,
850 ["lowbar"] = 95,
851 ["grave"] = 96,
852 ["DiacriticalGrave"] = 96,
853 ["lcub"] = 123,
854 ["lbrace"] = 123,
855 ["verbar"] = 124,
856 ["vert"] = 124,
857 ["VerticalLine"] = 124,
858 ["rcub"] = 125,
859 ["rbrace"] = 125,
860 ["nbsp"] = 160,
861 ["NonBreakingSpace"] = 160,
862 ["iexcl"] = 161,
863 ["cent"] = 162,
864 ["pound"] = 163,
865 ["curren"] = 164,
866 ["yen"] = 165,
867 ["brvbar"] = 166,
868 ["sect"] = 167,
869 ["Dot"] = 168,
870 ["die"] = 168,
871 ["DoubleDot"] = 168,
872 ["uml"] = 168,
873 ["copy"] = 169,
874 ["COPY"] = 169,
875 ["ordf"] = 170,

876 ["laquo"] = 171,
877 ["not"] = 172,
878 ["shy"] = 173,
879 ["reg"] = 174,
880 ["circledR"] = 174,
881 ["REG"] = 174,
882 ["macr"] = 175,
883 ["OverBar"] = 175,
884 ["strns"] = 175,
885 ["deg"] = 176,
886 ["plusmn"] = 177,
887 ["pm"] = 177,
888 ["PlusMinus"] = 177,
889 ["sup2"] = 178,
890 ["sup3"] = 179,
891 ["acute"] = 180,
892 ["DiacriticalAcute"] = 180,
893 ["micro"] = 181,
894 ["para"] = 182,
895 ["middot"] = 183,
896 ["centerdot"] = 183,
897 ["CenterDot"] = 183,
898 ["cedil"] = 184,
899 ["Cedilla"] = 184,
900 ["sup1"] = 185,
901 ["ordm"] = 186,
902 ["raquo"] = 187,
903 ["frac14"] = 188,
904 ["frac12"] = 189,
905 ["half"] = 189,
906 ["frac34"] = 190,
907 ["iquest"] = 191,
908 ["Agrave"] = 192,
909 ["Aacute"] = 193,
910 ["Acirc"] = 194,
911 ["Atilde"] = 195,
912 ["Auml"] = 196,
913 ["Aring"] = 197,
914 ["AElig"] = 198,
915 ["Ccedil"] = 199,
916 ["Egrave"] = 200,
917 ["Eacute"] = 201,
918 ["Ecirc"] = 202,
919 ["Euml"] = 203,
920 ["Igrave"] = 204,
921 ["Iacute"] = 205,
922 ["Icirc"] = 206,

923 ["Iuml"] = 207,
924 ["ETH"] = 208,
925 ["Ntilde"] = 209,
926 ["Ograve"] = 210,
927 ["Oacute"] = 211,
928 ["Ocirc"] = 212,
929 ["Otilde"] = 213,
930 ["Ouml"] = 214,
931 ["times"] = 215,
932 ["Oslash"] = 216,
933 ["Ugrave"] = 217,
934 ["Uacute"] = 218,
935 ["Ucirc"] = 219,
936 ["Uuml"] = 220,
937 ["Yacute"] = 221,
938 ["THORN"] = 222,
939 ["szlig"] = 223,
940 ["agrave"] = 224,
941 ["aacute"] = 225,
942 ["acirc"] = 226,
943 ["atilde"] = 227,
944 ["auml"] = 228,
945 ["aring"] = 229,
946 ["aelig"] = 230,
947 ["ccedil"] = 231,
948 ["egrave"] = 232,
949 ["eacute"] = 233,
950 ["ecirc"] = 234,
951 ["euml"] = 235,
952 ["igrave"] = 236,
953 ["iacute"] = 237,
954 ["icirc"] = 238,
955 ["iuml"] = 239,
956 ["eth"] = 240,
957 ["ntilde"] = 241,
958 ["ograve"] = 242,
959 ["oacute"] = 243,
960 ["ocirc"] = 244,
961 ["otilde"] = 245,
962 ["ouml"] = 246,
963 ["divide"] = 247,
964 ["div"] = 247,
965 ["oslash"] = 248,
966 ["ugrave"] = 249,
967 ["uacute"] = 250,
968 ["ucirc"] = 251,
969 ["uuml"] = 252,

970 ["yacute"] = 253,
971 ["thorn"] = 254,
972 ["yuml"] = 255,
973 ["Amacr"] = 256,
974 ["amacr"] = 257,
975 ["Abreve"] = 258,
976 ["abreve"] = 259,
977 ["Aogon"] = 260,
978 ["aogon"] = 261,
979 ["Cacute"] = 262,
980 ["cacute"] = 263,
981 ["Ccirc"] = 264,
982 ["ccirc"] = 265,
983 ["Cdot"] = 266,
984 ["cdot"] = 267,
985 ["Ccaron"] = 268,
986 ["ccaron"] = 269,
987 ["Dcaron"] = 270,
988 ["dcaron"] = 271,
989 ["Dstrok"] = 272,
990 ["dstrok"] = 273,
991 ["Emacr"] = 274,
992 ["emacr"] = 275,
993 ["Edot"] = 278,
994 ["edot"] = 279,
995 ["Eogon"] = 280,
996 ["eogon"] = 281,
997 ["Ecaron"] = 282,
998 ["ecaron"] = 283,
999 ["Gcirc"] = 284,
1000 ["gcirc"] = 285,
1001 ["Gbreve"] = 286,
1002 ["gbreve"] = 287,
1003 ["Gdot"] = 288,
1004 ["gdot"] = 289,
1005 ["Gcedil"] = 290,
1006 ["Hcirc"] = 292,
1007 ["hcirc"] = 293,
1008 ["Hstrok"] = 294,
1009 ["hstrok"] = 295,
1010 ["Itilde"] = 296,
1011 ["itilde"] = 297,
1012 ["Imacr"] = 298,
1013 ["imacr"] = 299,
1014 ["Iogon"] = 302,
1015 ["iogon"] = 303,
1016 ["Idot"] = 304,

1017 ["imath"] = 305,
1018 ["inodot"] = 305,
1019 ["IJlig"] = 306,
1020 ["ijlig"] = 307,
1021 ["Jcirc"] = 308,
1022 ["jcirc"] = 309,
1023 ["Kcedil"] = 310,
1024 ["kcedil"] = 311,
1025 ["kgreen"] = 312,
1026 ["Lacute"] = 313,
1027 ["lacute"] = 314,
1028 ["Lcedil"] = 315,
1029 ["lcedil"] = 316,
1030 ["Lcaron"] = 317,
1031 ["lcaron"] = 318,
1032 ["Lmidot"] = 319,
1033 ["lmidot"] = 320,
1034 ["Lstrok"] = 321,
1035 ["lstrok"] = 322,
1036 ["Nacute"] = 323,
1037 ["nacute"] = 324,
1038 ["Ncedil"] = 325,
1039 ["ncedil"] = 326,
1040 ["Ncaron"] = 327,
1041 ["ncaron"] = 328,
1042 ["napos"] = 329,
1043 ["ENG"] = 330,
1044 ["eng"] = 331,
1045 ["Omacr"] = 332,
1046 ["omacr"] = 333,
1047 ["Odblac"] = 336,
1048 ["odblac"] = 337,
1049 ["OElig"] = 338,
1050 ["oelig"] = 339,
1051 ["Racute"] = 340,
1052 ["racute"] = 341,
1053 ["Rcedil"] = 342,
1054 ["rcedil"] = 343,
1055 ["Rcaron"] = 344,
1056 ["rcaron"] = 345,
1057 ["Sacute"] = 346,
1058 ["sacute"] = 347,
1059 ["Scirc"] = 348,
1060 ["scirc"] = 349,
1061 ["Scedil"] = 350,
1062 ["scedil"] = 351,
1063 ["Scaron"] = 352,

1064 ["scaron"] = 353,
1065 ["Tcedil"] = 354,
1066 ["tcedil"] = 355,
1067 ["Tcaron"] = 356,
1068 ["tcaron"] = 357,
1069 ["Tstrok"] = 358,
1070 ["tstrok"] = 359,
1071 ["Utilde"] = 360,
1072 ["utilde"] = 361,
1073 ["Umacr"] = 362,
1074 ["umacr"] = 363,
1075 ["Ubreve"] = 364,
1076 ["ubreve"] = 365,
1077 ["Uring"] = 366,
1078 ["uring"] = 367,
1079 ["Udblac"] = 368,
1080 ["udblac"] = 369,
1081 ["Uogon"] = 370,
1082 ["uogon"] = 371,
1083 ["Wcirc"] = 372,
1084 ["wcirc"] = 373,
1085 ["Ycirc"] = 374,
1086 ["ycirc"] = 375,
1087 ["Yuml"] = 376,
1088 ["Zacute"] = 377,
1089 ["zacute"] = 378,
1090 ["Zdot"] = 379,
1091 ["zdot"] = 380,
1092 ["Zcaron"] = 381,
1093 ["zcaron"] = 382,
1094 ["fnof"] = 402,
1095 ["imped"] = 437,
1096 ["gacute"] = 501,
1097 ["jmath"] = 567,
1098 ["circ"] = 710,
1099 ["caron"] = 711,
1100 ["Hacek"] = 711,
1101 ["breve"] = 728,
1102 ["Breve"] = 728,
1103 ["dot"] = 729,
1104 ["DiacriticalDot"] = 729,
1105 ["ring"] = 730,
1106 ["ogon"] = 731,
1107 ["tilde"] = 732,
1108 ["DiacriticalTilde"] = 732,
1109 ["dblac"] = 733,
1110 ["DiacriticalDoubleAcute"] = 733,

1111 ["DownBreve"] = 785,
1112 ["UnderBar"] = 818,
1113 ["Alpha"] = 913,
1114 ["Beta"] = 914,
1115 ["Gamma"] = 915,
1116 ["Delta"] = 916,
1117 ["Epsilon"] = 917,
1118 ["Zeta"] = 918,
1119 ["Eta"] = 919,
1120 ["Theta"] = 920,
1121 ["Iota"] = 921,
1122 ["Kappa"] = 922,
1123 ["Lambda"] = 923,
1124 ["Mu"] = 924,
1125 ["Nu"] = 925,
1126 ["Xi"] = 926,
1127 ["Omicron"] = 927,
1128 ["Pi"] = 928,
1129 ["Rho"] = 929,
1130 ["Sigma"] = 931,
1131 ["Tau"] = 932,
1132 ["Upsilon"] = 933,
1133 ["Phi"] = 934,
1134 ["Chi"] = 935,
1135 ["Psi"] = 936,
1136 ["Omega"] = 937,
1137 ["alpha"] = 945,
1138 ["beta"] = 946,
1139 ["gamma"] = 947,
1140 ["delta"] = 948,
1141 ["epsiv"] = 949,
1142 ["varepsilon"] = 949,
1143 ["epsilon"] = 949,
1144 ["zeta"] = 950,
1145 ["eta"] = 951,
1146 ["theta"] = 952,
1147 ["iota"] = 953,
1148 ["kappa"] = 954,
1149 ["lambda"] = 955,
1150 ["mu"] = 956,
1151 ["nu"] = 957,
1152 ["xi"] = 958,
1153 ["omicron"] = 959,
1154 ["pi"] = 960,
1155 ["rho"] = 961,
1156 ["sigmav"] = 962,
1157 ["varsigma"] = 962,

1158 ["sigmaf"] = 962,
1159 ["sigma"] = 963,
1160 ["tau"] = 964,
1161 ["upsilon"] = 965,
1162 ["upsilon"] = 965,
1163 ["phi"] = 966,
1164 ["phiv"] = 966,
1165 ["varphi"] = 966,
1166 ["chi"] = 967,
1167 ["psi"] = 968,
1168 ["omega"] = 969,
1169 ["thetav"] = 977,
1170 ["vartheta"] = 977,
1171 ["thetasym"] = 977,
1172 ["Upsilon"] = 978,
1173 ["upsih"] = 978,
1174 ["straightphi"] = 981,
1175 ["piv"] = 982,
1176 ["varpi"] = 982,
1177 ["Gammad"] = 988,
1178 ["gammad"] = 989,
1179 ["digamma"] = 989,
1180 ["kappav"] = 1008,
1181 ["varkappa"] = 1008,
1182 ["rhov"] = 1009,
1183 ["varrho"] = 1009,
1184 ["epsi"] = 1013,
1185 ["straightepsilon"] = 1013,
1186 ["bepsi"] = 1014,
1187 ["backepsilon"] = 1014,
1188 ["IOcy"] = 1025,
1189 ["DJcy"] = 1026,
1190 ["GJcy"] = 1027,
1191 ["Jukcy"] = 1028,
1192 ["DScy"] = 1029,
1193 ["Iukcy"] = 1030,
1194 ["YIcy"] = 1031,
1195 ["Jsercy"] = 1032,
1196 ["LJcy"] = 1033,
1197 ["NJcy"] = 1034,
1198 ["TSHcy"] = 1035,
1199 ["KJcy"] = 1036,
1200 ["Ubrcy"] = 1038,
1201 ["DZcy"] = 1039,
1202 ["Acy"] = 1040,
1203 ["Bcy"] = 1041,
1204 ["Vcy"] = 1042,

1205 ["Gcy"] = 1043,
1206 ["Dcy"] = 1044,
1207 ["IEcy"] = 1045,
1208 ["ZHcy"] = 1046,
1209 ["Zcy"] = 1047,
1210 ["Icy"] = 1048,
1211 ["Jcy"] = 1049,
1212 ["Kcy"] = 1050,
1213 ["Lcy"] = 1051,
1214 ["Mcy"] = 1052,
1215 ["Ncy"] = 1053,
1216 ["Ocy"] = 1054,
1217 ["Pcy"] = 1055,
1218 ["Rcy"] = 1056,
1219 ["Scy"] = 1057,
1220 ["Tcy"] = 1058,
1221 ["Ucy"] = 1059,
1222 ["Fcy"] = 1060,
1223 ["KHcy"] = 1061,
1224 ["TScy"] = 1062,
1225 ["CHcy"] = 1063,
1226 ["SHcy"] = 1064,
1227 ["SHCHcy"] = 1065,
1228 ["HARDcy"] = 1066,
1229 ["Ycy"] = 1067,
1230 ["SOFTcy"] = 1068,
1231 ["Ecy"] = 1069,
1232 ["YUcy"] = 1070,
1233 ["YAcy"] = 1071,
1234 ["acy"] = 1072,
1235 ["bcy"] = 1073,
1236 ["vcy"] = 1074,
1237 ["gcy"] = 1075,
1238 ["dcy"] = 1076,
1239 ["iecy"] = 1077,
1240 ["zhcy"] = 1078,
1241 ["zcy"] = 1079,
1242 ["icy"] = 1080,
1243 ["jcy"] = 1081,
1244 ["kcy"] = 1082,
1245 ["lcy"] = 1083,
1246 ["mcy"] = 1084,
1247 ["ncy"] = 1085,
1248 ["ocy"] = 1086,
1249 ["pcy"] = 1087,
1250 ["rcy"] = 1088,
1251 ["scy"] = 1089,

1252 ["tcy"] = 1090,
1253 ["ucy"] = 1091,
1254 ["fcy"] = 1092,
1255 ["khcy"] = 1093,
1256 ["tscy"] = 1094,
1257 ["chcy"] = 1095,
1258 ["shcy"] = 1096,
1259 ["shchcy"] = 1097,
1260 ["hardcy"] = 1098,
1261 ["ycy"] = 1099,
1262 ["softcy"] = 1100,
1263 ["ecy"] = 1101,
1264 ["yucy"] = 1102,
1265 ["yacy"] = 1103,
1266 ["iocy"] = 1105,
1267 ["djcy"] = 1106,
1268 ["gjcy"] = 1107,
1269 ["jukcy"] = 1108,
1270 ["dscy"] = 1109,
1271 ["iukcy"] = 1110,
1272 ["yicy"] = 1111,
1273 ["jsercy"] = 1112,
1274 ["ljcy"] = 1113,
1275 ["njcy"] = 1114,
1276 ["tshcy"] = 1115,
1277 ["kjcy"] = 1116,
1278 ["ubrscy"] = 1118,
1279 ["dzcy"] = 1119,
1280 ["ensp"] = 8194,
1281 ["emsp"] = 8195,
1282 ["emsp13"] = 8196,
1283 ["emsp14"] = 8197,
1284 ["numsp"] = 8199,
1285 ["puncsp"] = 8200,
1286 ["thinsp"] = 8201,
1287 ["ThinSpace"] = 8201,
1288 ["hairsp"] = 8202,
1289 ["VeryThinSpace"] = 8202,
1290 ["ZeroWidthSpace"] = 8203,
1291 ["NegativeVeryThinSpace"] = 8203,
1292 ["NegativeThinSpace"] = 8203,
1293 ["NegativeMediumSpace"] = 8203,
1294 ["NegativeThickSpace"] = 8203,
1295 ["zwnj"] = 8204,
1296 ["zwj"] = 8205,
1297 ["lrm"] = 8206,
1298 ["rlm"] = 8207,

1299 ["hyphen"] = 8208,
1300 ["dash"] = 8208,
1301 ["ndash"] = 8211,
1302 ["mdash"] = 8212,
1303 ["horbar"] = 8213,
1304 ["Verbar"] = 8214,
1305 ["Vert"] = 8214,
1306 ["lsquo"] = 8216,
1307 ["OpenCurlyQuote"] = 8216,
1308 ["rsquo"] = 8217,
1309 ["rsquor"] = 8217,
1310 ["CloseCurlyQuote"] = 8217,
1311 ["lsquor"] = 8218,
1312 ["sbquo"] = 8218,
1313 ["ldquo"] = 8220,
1314 ["OpenCurlyDoubleQuote"] = 8220,
1315 ["rdquo"] = 8221,
1316 ["rdquor"] = 8221,
1317 ["CloseCurlyDoubleQuote"] = 8221,
1318 ["ldquor"] = 8222,
1319 ["bdquo"] = 8222,
1320 ["dagger"] = 8224,
1321 ["Dagger"] = 8225,
1322 ["ddagger"] = 8225,
1323 ["bull"] = 8226,
1324 ["bullet"] = 8226,
1325 ["nldr"] = 8229,
1326 ["hellip"] = 8230,
1327 ["mldr"] = 8230,
1328 ["permil"] = 8240,
1329 ["pertenk"] = 8241,
1330 ["prime"] = 8242,
1331 ["Prime"] = 8243,
1332 ["tprime"] = 8244,
1333 ["bprime"] = 8245,
1334 ["backprime"] = 8245,
1335 ["lsaquo"] = 8249,
1336 ["rsaquo"] = 8250,
1337 ["oline"] = 8254,
1338 ["caret"] = 8257,
1339 ["hybull"] = 8259,
1340 ["frasl"] = 8260,
1341 ["bsemi"] = 8271,
1342 ["qprime"] = 8279,
1343 ["MediumSpace"] = 8287,
1344 ["NoBreak"] = 8288,
1345 ["ApplyFunction"] = 8289,

1346 ["af"] = 8289,
1347 ["InvisibleTimes"] = 8290,
1348 ["it"] = 8290,
1349 ["InvisibleComma"] = 8291,
1350 ["ic"] = 8291,
1351 ["euro"] = 8364,
1352 ["tdot"] = 8411,
1353 ["TripleDot"] = 8411,
1354 ["DotDot"] = 8412,
1355 ["Copf"] = 8450,
1356 ["complexes"] = 8450,
1357 ["incare"] = 8453,
1358 ["gscr"] = 8458,
1359 ["hamilt"] = 8459,
1360 ["HilbertSpace"] = 8459,
1361 ["Hscr"] = 8459,
1362 ["Hfr"] = 8460,
1363 ["Poincareplane"] = 8460,
1364 ["quaternions"] = 8461,
1365 ["Hopf"] = 8461,
1366 ["planckh"] = 8462,
1367 ["planck"] = 8463,
1368 ["hbar"] = 8463,
1369 ["plankv"] = 8463,
1370 ["hslash"] = 8463,
1371 ["Iscr"] = 8464,
1372 ["imagline"] = 8464,
1373 ["image"] = 8465,
1374 ["Im"] = 8465,
1375 ["imagpart"] = 8465,
1376 ["Ifr"] = 8465,
1377 ["Lscr"] = 8466,
1378 ["lagran"] = 8466,
1379 ["Laplacetrif"] = 8466,
1380 ["ell"] = 8467,
1381 ["Nopf"] = 8469,
1382 ["naturals"] = 8469,
1383 ["numero"] = 8470,
1384 ["copysr"] = 8471,
1385 ["weierp"] = 8472,
1386 ["wp"] = 8472,
1387 ["Popf"] = 8473,
1388 ["primes"] = 8473,
1389 ["rationals"] = 8474,
1390 ["Qopf"] = 8474,
1391 ["Rscr"] = 8475,
1392 ["realine"] = 8475,

1393 ["real"] = 8476,
1394 ["Re"] = 8476,
1395 ["realpart"] = 8476,
1396 ["Rfr"] = 8476,
1397 ["reals"] = 8477,
1398 ["Ropf"] = 8477,
1399 ["rx"] = 8478,
1400 ["trade"] = 8482,
1401 ["TRADE"] = 8482,
1402 ["integers"] = 8484,
1403 ["Zopf"] = 8484,
1404 ["ohm"] = 8486,
1405 ["mho"] = 8487,
1406 ["Zfr"] = 8488,
1407 ["zeetrf"] = 8488,
1408 ["iiota"] = 8489,
1409 ["angst"] = 8491,
1410 ["bernou"] = 8492,
1411 ["Bernoullis"] = 8492,
1412 ["Bscr"] = 8492,
1413 ["Cfr"] = 8493,
1414 ["Cayleys"] = 8493,
1415 ["escr"] = 8495,
1416 ["Escr"] = 8496,
1417 ["expectation"] = 8496,
1418 ["Fscr"] = 8497,
1419 ["Fouriertrf"] = 8497,
1420 ["phmmat"] = 8499,
1421 ["Mellintrf"] = 8499,
1422 ["Mscr"] = 8499,
1423 ["order"] = 8500,
1424 ["orderof"] = 8500,
1425 ["oscr"] = 8500,
1426 ["alefsym"] = 8501,
1427 ["aleph"] = 8501,
1428 ["beth"] = 8502,
1429 ["gimel"] = 8503,
1430 ["daleth"] = 8504,
1431 ["CapitalDifferentialD"] = 8517,
1432 ["DD"] = 8517,
1433 ["DifferentialD"] = 8518,
1434 ["dd"] = 8518,
1435 ["ExponentialE"] = 8519,
1436 ["exponentiale"] = 8519,
1437 ["ee"] = 8519,
1438 ["ImaginaryI"] = 8520,
1439 ["ii"] = 8520,

1440 ["frac13"] = 8531,
1441 ["frac23"] = 8532,
1442 ["frac15"] = 8533,
1443 ["frac25"] = 8534,
1444 ["frac35"] = 8535,
1445 ["frac45"] = 8536,
1446 ["frac16"] = 8537,
1447 ["frac56"] = 8538,
1448 ["frac18"] = 8539,
1449 ["frac38"] = 8540,
1450 ["frac58"] = 8541,
1451 ["frac78"] = 8542,
1452 ["larr"] = 8592,
1453 ["leftarrow"] = 8592,
1454 ["LeftArrow"] = 8592,
1455 ["slarr"] = 8592,
1456 ["ShortLeftArrow"] = 8592,
1457 ["uarr"] = 8593,
1458 ["uparrow"] = 8593,
1459 ["UpArrow"] = 8593,
1460 ["ShortUpArrow"] = 8593,
1461 ["rarr"] = 8594,
1462 ["rightarrow"] = 8594,
1463 ["RightArrow"] = 8594,
1464 ["srarr"] = 8594,
1465 ["ShortRightArrow"] = 8594,
1466 ["darr"] = 8595,
1467 ["downarrow"] = 8595,
1468 ["DownArrow"] = 8595,
1469 ["ShortDownArrow"] = 8595,
1470 ["harr"] = 8596,
1471 ["leftrightarrow"] = 8596,
1472 ["LeftRightArrow"] = 8596,
1473 ["varr"] = 8597,
1474 ["updownarrow"] = 8597,
1475 ["UpDownArrow"] = 8597,
1476 ["nwarr"] = 8598,
1477 ["UpperLeftArrow"] = 8598,
1478 ["nwarrow"] = 8598,
1479 ["nearr"] = 8599,
1480 ["UpperRightArrow"] = 8599,
1481 ["nearrow"] = 8599,
1482 ["searr"] = 8600,
1483 ["searrow"] = 8600,
1484 ["LowerRightArrow"] = 8600,
1485 ["swarr"] = 8601,
1486 ["swarrow"] = 8601,

1487 ["LowerLeftArrow"] = 8601,
1488 ["nlarr"] = 8602,
1489 ["nleftarrow"] = 8602,
1490 ["nrarr"] = 8603,
1491 ["nrightarrow"] = 8603,
1492 ["rarrw"] = 8605,
1493 ["rightsquigarrow"] = 8605,
1494 ["Larr"] = 8606,
1495 ["twoheadleftarrow"] = 8606,
1496 ["Uarr"] = 8607,
1497 ["Rarr"] = 8608,
1498 ["twoheadrightarrow"] = 8608,
1499 ["Darr"] = 8609,
1500 ["larrtl"] = 8610,
1501 ["leftarrowtail"] = 8610,
1502 ["rarrtl"] = 8611,
1503 ["rightarrowtail"] = 8611,
1504 ["LeftTeeArrow"] = 8612,
1505 ["mapstoleft"] = 8612,
1506 ["UpTeeArrow"] = 8613,
1507 ["mapstoup"] = 8613,
1508 ["map"] = 8614,
1509 ["RightTeeArrow"] = 8614,
1510 ["mapsto"] = 8614,
1511 ["DownTeeArrow"] = 8615,
1512 ["mapstodown"] = 8615,
1513 ["larrhk"] = 8617,
1514 ["hookleftarrow"] = 8617,
1515 ["rarrhk"] = 8618,
1516 ["hookrightarrow"] = 8618,
1517 ["larrlp"] = 8619,
1518 ["looparrowleft"] = 8619,
1519 ["rarrlp"] = 8620,
1520 ["looparrowright"] = 8620,
1521 ["harrw"] = 8621,
1522 ["leftrightsquigarrow"] = 8621,
1523 ["nharr"] = 8622,
1524 ["nleftrightarrow"] = 8622,
1525 ["lsh"] = 8624,
1526 ["Lsh"] = 8624,
1527 ["rsh"] = 8625,
1528 ["Rsh"] = 8625,
1529 ["ldsh"] = 8626,
1530 ["rdsh"] = 8627,
1531 ["crarr"] = 8629,
1532 ["cularr"] = 8630,
1533 ["curvearrowleft"] = 8630,

1534 ["curarr"] = 8631,
1535 ["curvearrowright"] = 8631,
1536 ["olarr"] = 8634,
1537 ["circlearrowleft"] = 8634,
1538 ["orarr"] = 8635,
1539 ["circlearrowright"] = 8635,
1540 ["lharu"] = 8636,
1541 ["LeftVector"] = 8636,
1542 ["leftharpoonup"] = 8636,
1543 ["lhard"] = 8637,
1544 ["leftharpoondown"] = 8637,
1545 ["DownLeftVector"] = 8637,
1546 ["uharr"] = 8638,
1547 ["upharpoonright"] = 8638,
1548 ["RightUpVector"] = 8638,
1549 ["uharl"] = 8639,
1550 ["upharpoonleft"] = 8639,
1551 ["LeftUpVector"] = 8639,
1552 ["rharu"] = 8640,
1553 ["RightVector"] = 8640,
1554 ["rightharpoonup"] = 8640,
1555 ["rhard"] = 8641,
1556 ["rightharpoondown"] = 8641,
1557 ["DownRightVector"] = 8641,
1558 ["dharr"] = 8642,
1559 ["RightDownVector"] = 8642,
1560 ["downharpoonright"] = 8642,
1561 ["dharl"] = 8643,
1562 ["LeftDownVector"] = 8643,
1563 ["downharpoonleft"] = 8643,
1564 ["rlarr"] = 8644,
1565 ["rightleftarrows"] = 8644,
1566 ["RightArrowLeftArrow"] = 8644,
1567 ["udarr"] = 8645,
1568 ["UpArrowDownArrow"] = 8645,
1569 ["lrarr"] = 8646,
1570 ["leftrightarrows"] = 8646,
1571 ["LeftArrowRightArrow"] = 8646,
1572 ["llarr"] = 8647,
1573 ["leftleftarrows"] = 8647,
1574 ["uuarr"] = 8648,
1575 ["upuparrows"] = 8648,
1576 ["rrarr"] = 8649,
1577 ["rightrightarrows"] = 8649,
1578 ["ddarr"] = 8650,
1579 ["downdownarrows"] = 8650,
1580 ["lrhar"] = 8651,

1581 ["ReverseEquilibrium"] = 8651,
1582 ["leftrightharpoons"] = 8651,
1583 ["rlhar"] = 8652,
1584 ["rightleftharpoons"] = 8652,
1585 ["Equilibrium"] = 8652,
1586 ["nlArr"] = 8653,
1587 ["nLeftarrow"] = 8653,
1588 ["nhArr"] = 8654,
1589 ["nLeftrightarrow"] = 8654,
1590 ["nrArr"] = 8655,
1591 ["nrightarrow"] = 8655,
1592 ["lArr"] = 8656,
1593 ["Leftarrow"] = 8656,
1594 ["DoubleLeftArrow"] = 8656,
1595 ["uArr"] = 8657,
1596 ["Uparrow"] = 8657,
1597 ["DoubleUpArrow"] = 8657,
1598 ["rArr"] = 8658,
1599 ["Rightarrow"] = 8658,
1600 ["Implies"] = 8658,
1601 ["DoubleRightArrow"] = 8658,
1602 ["dArr"] = 8659,
1603 ["Downarrow"] = 8659,
1604 ["DoubleDownArrow"] = 8659,
1605 ["hArr"] = 8660,
1606 ["Leftrightarrow"] = 8660,
1607 ["DoubleLeftRightArrow"] = 8660,
1608 ["iff"] = 8660,
1609 ["vArr"] = 8661,
1610 ["Updownarrow"] = 8661,
1611 ["DoubleUpDownArrow"] = 8661,
1612 ["nwArr"] = 8662,
1613 ["neArr"] = 8663,
1614 ["seArr"] = 8664,
1615 ["swArr"] = 8665,
1616 ["lAarr"] = 8666,
1617 ["Lleftarrow"] = 8666,
1618 ["rAarr"] = 8667,
1619 ["Rrightarrow"] = 8667,
1620 ["zigrarr"] = 8669,
1621 ["larrb"] = 8676,
1622 ["LeftArrowBar"] = 8676,
1623 ["rarrb"] = 8677,
1624 ["RightArrowBar"] = 8677,
1625 ["duarr"] = 8693,
1626 ["DownArrowUpArrow"] = 8693,
1627 ["loarr"] = 8701,

1628 ["roarr"] = 8702,
1629 ["hoarr"] = 8703,
1630 ["forall"] = 8704,
1631 ["ForAll"] = 8704,
1632 ["comp"] = 8705,
1633 ["complement"] = 8705,
1634 ["part"] = 8706,
1635 ["PartialD"] = 8706,
1636 ["exist"] = 8707,
1637 ["Exists"] = 8707,
1638 ["nexist"] = 8708,
1639 ["NotExists"] = 8708,
1640 ["nexists"] = 8708,
1641 ["empty"] = 8709,
1642 ["emptyset"] = 8709,
1643 ["emptyv"] = 8709,
1644 ["varnothing"] = 8709,
1645 ["nabla"] = 8711,
1646 ["Del"] = 8711,
1647 ["isin"] = 8712,
1648 ["isinv"] = 8712,
1649 ["Element"] = 8712,
1650 ["in"] = 8712,
1651 ["notin"] = 8713,
1652 ["NotElement"] = 8713,
1653 ["notinva"] = 8713,
1654 ["niv"] = 8715,
1655 ["ReverseElement"] = 8715,
1656 ["ni"] = 8715,
1657 ["SuchThat"] = 8715,
1658 ["notni"] = 8716,
1659 ["notniva"] = 8716,
1660 ["NotReverseElement"] = 8716,
1661 ["prod"] = 8719,
1662 ["Product"] = 8719,
1663 ["coprod"] = 8720,
1664 ["Coproduct"] = 8720,
1665 ["sum"] = 8721,
1666 ["Sum"] = 8721,
1667 ["minus"] = 8722,
1668 ["mnplus"] = 8723,
1669 ["mp"] = 8723,
1670 ["MinusPlus"] = 8723,
1671 ["plusdo"] = 8724,
1672 ["dotplus"] = 8724,
1673 ["setmn"] = 8726,
1674 ["setminus"] = 8726,

1675 ["Backslash"] = 8726,
1676 ["ssetmn"] = 8726,
1677 ["smallsetminus"] = 8726,
1678 ["lowast"] = 8727,
1679 ["compfn"] = 8728,
1680 ["SmallCircle"] = 8728,
1681 ["radic"] = 8730,
1682 ["Sqrt"] = 8730,
1683 ["prop"] = 8733,
1684 ["propto"] = 8733,
1685 ["Proportional"] = 8733,
1686 ["vprop"] = 8733,
1687 ["varpropto"] = 8733,
1688 ["infin"] = 8734,
1689 ["angrt"] = 8735,
1690 ["ang"] = 8736,
1691 ["angle"] = 8736,
1692 ["angmsd"] = 8737,
1693 ["measuredangle"] = 8737,
1694 ["angsph"] = 8738,
1695 ["mid"] = 8739,
1696 ["VerticalBar"] = 8739,
1697 ["smid"] = 8739,
1698 ["shortmid"] = 8739,
1699 ["nmid"] = 8740,
1700 ["NotVerticalBar"] = 8740,
1701 ["nsmid"] = 8740,
1702 ["nshortmid"] = 8740,
1703 ["par"] = 8741,
1704 ["parallel"] = 8741,
1705 ["DoubleVerticalBar"] = 8741,
1706 ["spar"] = 8741,
1707 ["shortparallel"] = 8741,
1708 ["npar"] = 8742,
1709 ["nparallel"] = 8742,
1710 ["NotDoubleVerticalBar"] = 8742,
1711 ["nspar"] = 8742,
1712 ["nshortparallel"] = 8742,
1713 ["and"] = 8743,
1714 ["wedge"] = 8743,
1715 ["or"] = 8744,
1716 ["vee"] = 8744,
1717 ["cap"] = 8745,
1718 ["cup"] = 8746,
1719 ["int"] = 8747,
1720 ["Integral"] = 8747,
1721 ["Int"] = 8748,

1722 ["tint"] = 8749,
1723 ["iiint"] = 8749,
1724 ["conint"] = 8750,
1725 ["oint"] = 8750,
1726 ["ContourIntegral"] = 8750,
1727 ["Conint"] = 8751,
1728 ["DoubleContourIntegral"] = 8751,
1729 ["Cconint"] = 8752,
1730 ["cwint"] = 8753,
1731 ["cwconint"] = 8754,
1732 ["ClockwiseContourIntegral"] = 8754,
1733 ["awconint"] = 8755,
1734 ["CounterClockwiseContourIntegral"] = 8755,
1735 ["there4"] = 8756,
1736 ["therefore"] = 8756,
1737 ["Therefore"] = 8756,
1738 ["becaus"] = 8757,
1739 ["because"] = 8757,
1740 ["Because"] = 8757,
1741 ["ratio"] = 8758,
1742 ["Colon"] = 8759,
1743 ["Proportion"] = 8759,
1744 ["minusd"] = 8760,
1745 ["dotminus"] = 8760,
1746 ["mDDot"] = 8762,
1747 ["homtht"] = 8763,
1748 ["sim"] = 8764,
1749 ["Tilde"] = 8764,
1750 ["thksim"] = 8764,
1751 ["thicksim"] = 8764,
1752 ["bsim"] = 8765,
1753 ["backsim"] = 8765,
1754 ["ac"] = 8766,
1755 ["mstpos"] = 8766,
1756 ["acd"] = 8767,
1757 ["wreath"] = 8768,
1758 ["VerticalTilde"] = 8768,
1759 ["wr"] = 8768,
1760 ["nsim"] = 8769,
1761 ["NotTilde"] = 8769,
1762 ["esim"] = 8770,
1763 ["EqualTilde"] = 8770,
1764 ["eqsim"] = 8770,
1765 ["sime"] = 8771,
1766 ["TildeEqual"] = 8771,
1767 ["simeq"] = 8771,
1768 ["nsime"] = 8772,

1769 ["nsimeq"] = 8772,
1770 ["NotTildeEqual"] = 8772,
1771 ["cong"] = 8773,
1772 ["TildeFullEqual"] = 8773,
1773 ["simne"] = 8774,
1774 ["ncong"] = 8775,
1775 ["NotTildeFullEqual"] = 8775,
1776 ["asymp"] = 8776,
1777 ["ap"] = 8776,
1778 ["TildeTilde"] = 8776,
1779 ["approx"] = 8776,
1780 ["thkap"] = 8776,
1781 ["thickapprox"] = 8776,
1782 ["nap"] = 8777,
1783 ["NotTildeTilde"] = 8777,
1784 ["naprox"] = 8777,
1785 ["ape"] = 8778,
1786 ["approxpeq"] = 8778,
1787 ["apid"] = 8779,
1788 ["bcong"] = 8780,
1789 ["backcong"] = 8780,
1790 ["asympeq"] = 8781,
1791 ["CupCap"] = 8781,
1792 ["bump"] = 8782,
1793 ["HumpDownHump"] = 8782,
1794 ["Bumpeq"] = 8782,
1795 ["bumpe"] = 8783,
1796 ["HumpEqual"] = 8783,
1797 ["bumpeq"] = 8783,
1798 ["esdot"] = 8784,
1799 ["DotEqual"] = 8784,
1800 ["doteq"] = 8784,
1801 ["eDot"] = 8785,
1802 ["doteqdot"] = 8785,
1803 ["efDot"] = 8786,
1804 ["fallingdotseq"] = 8786,
1805 ["erDot"] = 8787,
1806 ["risingdotseq"] = 8787,
1807 ["colone"] = 8788,
1808 ["coloneq"] = 8788,
1809 ["Assign"] = 8788,
1810 ["ecolon"] = 8789,
1811 ["eqcolon"] = 8789,
1812 ["ecir"] = 8790,
1813 ["eqcirc"] = 8790,
1814 ["cire"] = 8791,
1815 ["circeq"] = 8791,

1816 ["wedgeq"] = 8793,
1817 ["veeeq"] = 8794,
1818 ["trie"] = 8796,
1819 ["triangleq"] = 8796,
1820 ["equest"] = 8799,
1821 ["questeq"] = 8799,
1822 ["ne"] = 8800,
1823 ["NotEqual"] = 8800,
1824 ["equiv"] = 8801,
1825 ["Congruent"] = 8801,
1826 ["nequiv"] = 8802,
1827 ["NotCongruent"] = 8802,
1828 ["le"] = 8804,
1829 ["leq"] = 8804,
1830 ["ge"] = 8805,
1831 ["GreaterEqual"] = 8805,
1832 ["geq"] = 8805,
1833 ["lE"] = 8806,
1834 ["LessFullEqual"] = 8806,
1835 ["leqq"] = 8806,
1836 ["gE"] = 8807,
1837 ["GreaterFullEqual"] = 8807,
1838 ["geqq"] = 8807,
1839 ["lnE"] = 8808,
1840 ["lneqq"] = 8808,
1841 ["gnE"] = 8809,
1842 ["gneqq"] = 8809,
1843 ["Lt"] = 8810,
1844 ["NestedLessLess"] = 8810,
1845 ["ll"] = 8810,
1846 ["Gt"] = 8811,
1847 ["NestedGreaterGreater"] = 8811,
1848 ["gg"] = 8811,
1849 ["twixt"] = 8812,
1850 ["between"] = 8812,
1851 ["NotCupCap"] = 8813,
1852 ["nlt"] = 8814,
1853 ["NotLess"] = 8814,
1854 ["nless"] = 8814,
1855 ["ngt"] = 8815,
1856 ["NotGreater"] = 8815,
1857 ["ngtr"] = 8815,
1858 ["nle"] = 8816,
1859 ["NotLessEqual"] = 8816,
1860 ["nleq"] = 8816,
1861 ["nge"] = 8817,
1862 ["NotGreaterEqual"] = 8817,

1863 ["ngeq"] = 8817,
1864 ["lsim"] = 8818,
1865 ["LessTilde"] = 8818,
1866 ["lesssim"] = 8818,
1867 ["gsim"] = 8819,
1868 ["gtrsim"] = 8819,
1869 ["GreaterTilde"] = 8819,
1870 ["nlsim"] = 8820,
1871 ["NotLessTilde"] = 8820,
1872 ["ngsim"] = 8821,
1873 ["NotGreaterTilde"] = 8821,
1874 ["lg"] = 8822,
1875 ["lessgtr"] = 8822,
1876 ["LessGreater"] = 8822,
1877 ["gl"] = 8823,
1878 ["gtrless"] = 8823,
1879 ["GreaterLess"] = 8823,
1880 ["ntlg"] = 8824,
1881 ["NotLessGreater"] = 8824,
1882 ["ntgl"] = 8825,
1883 ["NotGreaterLess"] = 8825,
1884 ["pr"] = 8826,
1885 ["Precedes"] = 8826,
1886 ["prec"] = 8826,
1887 ["sc"] = 8827,
1888 ["Succeeds"] = 8827,
1889 ["succ"] = 8827,
1890 ["prcue"] = 8828,
1891 ["PrecedesSlantEqual"] = 8828,
1892 ["preccurlyeq"] = 8828,
1893 ["sccue"] = 8829,
1894 ["SucceedsSlantEqual"] = 8829,
1895 ["succcurlyeq"] = 8829,
1896 ["prsim"] = 8830,
1897 ["precsim"] = 8830,
1898 ["PrecedesTilde"] = 8830,
1899 ["scsim"] = 8831,
1900 ["succsim"] = 8831,
1901 ["SucceedsTilde"] = 8831,
1902 ["npr"] = 8832,
1903 ["nprec"] = 8832,
1904 ["NotPrecedes"] = 8832,
1905 ["nsc"] = 8833,
1906 ["nsucc"] = 8833,
1907 ["NotSucceeds"] = 8833,
1908 ["sub"] = 8834,
1909 ["subset"] = 8834,

1910 ["sup"] = 8835,
 1911 ["supset"] = 8835,
 1912 ["Superset"] = 8835,
 1913 ["nsub"] = 8836,
 1914 ["nsup"] = 8837,
 1915 ["sube"] = 8838,
 1916 ["SubsetEqual"] = 8838,
 1917 ["subseteq"] = 8838,
 1918 ["supe"] = 8839,
 1919 ["supseteq"] = 8839,
 1920 ["SupersetEqual"] = 8839,
 1921 ["nsube"] = 8840,
 1922 ["nsubseteq"] = 8840,
 1923 ["NotSubsetEqual"] = 8840,
 1924 ["nsupe"] = 8841,
 1925 ["nsupseteq"] = 8841,
 1926 ["NotSupersetEqual"] = 8841,
 1927 ["subne"] = 8842,
 1928 ["subsetneq"] = 8842,
 1929 ["supne"] = 8843,
 1930 ["supsetneq"] = 8843,
 1931 ["cupdot"] = 8845,
 1932 ["uplus"] = 8846,
 1933 ["UnionPlus"] = 8846,
 1934 ["sqsub"] = 8847,
 1935 ["SquareSubset"] = 8847,
 1936 ["sqsubset"] = 8847,
 1937 ["sqsup"] = 8848,
 1938 ["SquareSuperset"] = 8848,
 1939 ["sqsupset"] = 8848,
 1940 ["sqsube"] = 8849,
 1941 ["SquareSubsetEqual"] = 8849,
 1942 ["sqsubseteq"] = 8849,
 1943 ["sqsupe"] = 8850,
 1944 ["SquareSupersetEqual"] = 8850,
 1945 ["sqsupseteq"] = 8850,
 1946 ["sqcap"] = 8851,
 1947 ["SquareIntersection"] = 8851,
 1948 ["sqcup"] = 8852,
 1949 ["SquareUnion"] = 8852,
 1950 ["oplus"] = 8853,
 1951 ["CirclePlus"] = 8853,
 1952 ["ominus"] = 8854,
 1953 ["CircleMinus"] = 8854,
 1954 ["otimes"] = 8855,
 1955 ["CircleTimes"] = 8855,
 1956 ["osol"] = 8856,

1957 ["odot"] = 8857,
 1958 ["CircleDot"] = 8857,
 1959 ["ocir"] = 8858,
 1960 ["circledcirc"] = 8858,
 1961 ["oast"] = 8859,
 1962 ["circledast"] = 8859,
 1963 ["odash"] = 8861,
 1964 ["circleddash"] = 8861,
 1965 ["plusb"] = 8862,
 1966 ["boxplus"] = 8862,
 1967 ["minusb"] = 8863,
 1968 ["boxminus"] = 8863,
 1969 ["timesb"] = 8864,
 1970 ["boxtimes"] = 8864,
 1971 ["sdotb"] = 8865,
 1972 ["dotsquare"] = 8865,
 1973 ["vdash"] = 8866,
 1974 ["RightTee"] = 8866,
 1975 ["dashv"] = 8867,
 1976 ["LeftTee"] = 8867,
 1977 ["top"] = 8868,
 1978 ["DownTee"] = 8868,
 1979 ["bottom"] = 8869,
 1980 ["bot"] = 8869,
 1981 ["perp"] = 8869,
 1982 ["UpTee"] = 8869,
 1983 ["models"] = 8871,
 1984 ["vDash"] = 8872,
 1985 ["DoubleRightTee"] = 8872,
 1986 ["Vdash"] = 8873,
 1987 ["Vvdash"] = 8874,
 1988 ["VDash"] = 8875,
 1989 ["nvdash"] = 8876,
 1990 ["nvDash"] = 8877,
 1991 ["nVdash"] = 8878,
 1992 ["nVDash"] = 8879,
 1993 ["prurel"] = 8880,
 1994 ["vltri"] = 8882,
 1995 ["vartriangleleft"] = 8882,
 1996 ["LeftTriangle"] = 8882,
 1997 ["vrtri"] = 8883,
 1998 ["vartriangleright"] = 8883,
 1999 ["RightTriangle"] = 8883,
 2000 ["ltrie"] = 8884,
 2001 ["trianglelefteq"] = 8884,
 2002 ["LeftTriangleEqual"] = 8884,
 2003 ["rtrie"] = 8885,

2004 ["trianglerighteq"] = 8885,
 2005 ["RightTriangleEqual"] = 8885,
 2006 ["origof"] = 8886,
 2007 ["imof"] = 8887,
 2008 ["mumap"] = 8888,
 2009 ["multimap"] = 8888,
 2010 ["hercon"] = 8889,
 2011 ["intcal"] = 8890,
 2012 ["intercal"] = 8890,
 2013 ["veebar"] = 8891,
 2014 ["barvee"] = 8893,
 2015 ["angrtvb"] = 8894,
 2016 ["lrtri"] = 8895,
 2017 ["xwedge"] = 8896,
 2018 ["Wedge"] = 8896,
 2019 ["bigwedge"] = 8896,
 2020 ["xvee"] = 8897,
 2021 ["Vee"] = 8897,
 2022 ["bigvee"] = 8897,
 2023 ["xcap"] = 8898,
 2024 ["Intersection"] = 8898,
 2025 ["bigcap"] = 8898,
 2026 ["xcup"] = 8899,
 2027 ["Union"] = 8899,
 2028 ["bigcup"] = 8899,
 2029 ["diam"] = 8900,
 2030 ["diamond"] = 8900,
 2031 ["Diamond"] = 8900,
 2032 ["sdot"] = 8901,
 2033 ["sstarf"] = 8902,
 2034 ["Star"] = 8902,
 2035 ["divonx"] = 8903,
 2036 ["divideontimes"] = 8903,
 2037 ["bowtie"] = 8904,
 2038 ["ltimes"] = 8905,
 2039 ["rtimes"] = 8906,
 2040 ["lthree"] = 8907,
 2041 ["leftthreetimes"] = 8907,
 2042 ["rthree"] = 8908,
 2043 ["rightthreetimes"] = 8908,
 2044 ["bsime"] = 8909,
 2045 ["backsimeq"] = 8909,
 2046 ["cuvee"] = 8910,
 2047 ["curlyvee"] = 8910,
 2048 ["cuwed"] = 8911,
 2049 ["curlywedge"] = 8911,
 2050 ["Sub"] = 8912,

2051 ["Subset"] = 8912,
 2052 ["Sup"] = 8913,
 2053 ["Supset"] = 8913,
 2054 ["Cap"] = 8914,
 2055 ["Cup"] = 8915,
 2056 ["fork"] = 8916,
 2057 ["pitchfork"] = 8916,
 2058 ["epar"] = 8917,
 2059 ["ltdot"] = 8918,
 2060 ["lessdot"] = 8918,
 2061 ["gtdot"] = 8919,
 2062 ["gtrdot"] = 8919,
 2063 ["Ll"] = 8920,
 2064 ["Gg"] = 8921,
 2065 ["ggg"] = 8921,
 2066 ["leg"] = 8922,
 2067 ["LessEqualGreater"] = 8922,
 2068 ["lesseqgtr"] = 8922,
 2069 ["gel"] = 8923,
 2070 ["gtreqless"] = 8923,
 2071 ["GreaterEqualLess"] = 8923,
 2072 ["cuepr"] = 8926,
 2073 ["curlyeqprec"] = 8926,
 2074 ["cuesc"] = 8927,
 2075 ["curlyeqsucc"] = 8927,
 2076 ["nprcue"] = 8928,
 2077 ["NotPrecedesSlantEqual"] = 8928,
 2078 ["nscue"] = 8929,
 2079 ["NotSucceedsSlantEqual"] = 8929,
 2080 ["nsqsube"] = 8930,
 2081 ["NotSquareSubsetEqual"] = 8930,
 2082 ["nsqsupe"] = 8931,
 2083 ["NotSquareSupersetEqual"] = 8931,
 2084 ["lnsim"] = 8934,
 2085 ["gnsim"] = 8935,
 2086 ["prnsim"] = 8936,
 2087 ["precnsim"] = 8936,
 2088 ["scnsim"] = 8937,
 2089 ["succnsim"] = 8937,
 2090 ["nltri"] = 8938,
 2091 ["ntriangleleft"] = 8938,
 2092 ["NotLeftTriangle"] = 8938,
 2093 ["nrtri"] = 8939,
 2094 ["ntriangleright"] = 8939,
 2095 ["NotRightTriangle"] = 8939,
 2096 ["nltrie"] = 8940,
 2097 ["ntrianglelefteq"] = 8940,

2098 ["NotLeftTriangleEqual"] = 8940,
 2099 ["nrtrie"] = 8941,
 2100 ["ntriananglerighteq"] = 8941,
 2101 ["NotRightTriangleEqual"] = 8941,
 2102 ["vellip"] = 8942,
 2103 ["ctdot"] = 8943,
 2104 ["utdot"] = 8944,
 2105 ["dtdot"] = 8945,
 2106 ["disin"] = 8946,
 2107 ["isinsv"] = 8947,
 2108 ["isins"] = 8948,
 2109 ["isindot"] = 8949,
 2110 ["notinvc"] = 8950,
 2111 ["notinvb"] = 8951,
 2112 ["isinE"] = 8953,
 2113 ["nisd"] = 8954,
 2114 ["xnis"] = 8955,
 2115 ["nis"] = 8956,
 2116 ["notnivc"] = 8957,
 2117 ["notnivb"] = 8958,
 2118 ["barwed"] = 8965,
 2119 ["barwedge"] = 8965,
 2120 ["Barwed"] = 8966,
 2121 ["doublebarwedge"] = 8966,
 2122 ["lceil"] = 8968,
 2123 ["LeftCeiling"] = 8968,
 2124 ["rceil"] = 8969,
 2125 ["RightCeiling"] = 8969,
 2126 ["lfloor"] = 8970,
 2127 ["LeftFloor"] = 8970,
 2128 ["rfloor"] = 8971,
 2129 ["RightFloor"] = 8971,
 2130 ["drcrop"] = 8972,
 2131 ["dlcrop"] = 8973,
 2132 ["urcrop"] = 8974,
 2133 ["ulcrop"] = 8975,
 2134 ["bnot"] = 8976,
 2135 ["proflin"] = 8978,
 2136 ["profsurf"] = 8979,
 2137 ["telrec"] = 8981,
 2138 ["target"] = 8982,
 2139 ["ulcorn"] = 8988,
 2140 ["ulcorner"] = 8988,
 2141 ["urcorn"] = 8989,
 2142 ["urcorner"] = 8989,
 2143 ["dlcorn"] = 8990,
 2144 ["llcorner"] = 8990,

2145 ["drcorn"] = 8991,
2146 ["lrcorner"] = 8991,
2147 ["frown"] = 8994,
2148 ["sfrown"] = 8994,
2149 ["smile"] = 8995,
2150 ["ssmile"] = 8995,
2151 ["cylcty"] = 9005,
2152 ["profalar"] = 9006,
2153 ["topbot"] = 9014,
2154 ["ovbar"] = 9021,
2155 ["solbar"] = 9023,
2156 ["angzarr"] = 9084,
2157 ["lmoust"] = 9136,
2158 ["lmoustache"] = 9136,
2159 ["rmoust"] = 9137,
2160 ["rmoustache"] = 9137,
2161 ["tbrk"] = 9140,
2162 ["OverBracket"] = 9140,
2163 ["bbrk"] = 9141,
2164 ["UnderBracket"] = 9141,
2165 ["bbrktbrk"] = 9142,
2166 ["OverParenthesis"] = 9180,
2167 ["UnderParenthesis"] = 9181,
2168 ["OverBrace"] = 9182,
2169 ["UnderBrace"] = 9183,
2170 ["trpezium"] = 9186,
2171 ["elinters"] = 9191,
2172 ["blank"] = 9251,
2173 ["oS"] = 9416,
2174 ["circledS"] = 9416,
2175 ["boxh"] = 9472,
2176 ["HorizontalLine"] = 9472,
2177 ["boxv"] = 9474,
2178 ["boxdr"] = 9484,
2179 ["boxdl"] = 9488,
2180 ["boxur"] = 9492,
2181 ["boxul"] = 9496,
2182 ["boxvr"] = 9500,
2183 ["boxvl"] = 9508,
2184 ["boxhd"] = 9516,
2185 ["boxhu"] = 9524,
2186 ["boxvh"] = 9532,
2187 ["boxH"] = 9552,
2188 ["boxV"] = 9553,
2189 ["boxdR"] = 9554,
2190 ["boxDr"] = 9555,
2191 ["boxDR"] = 9556,

2192 ["boxdL"] = 9557,
 2193 ["boxDl"] = 9558,
 2194 ["boxDL"] = 9559,
 2195 ["boxuR"] = 9560,
 2196 ["boxUr"] = 9561,
 2197 ["boxUR"] = 9562,
 2198 ["boxuL"] = 9563,
 2199 ["boxUl"] = 9564,
 2200 ["boxUL"] = 9565,
 2201 ["boxvR"] = 9566,
 2202 ["boxVr"] = 9567,
 2203 ["boxVR"] = 9568,
 2204 ["boxvL"] = 9569,
 2205 ["boxVl"] = 9570,
 2206 ["boxVL"] = 9571,
 2207 ["boxHd"] = 9572,
 2208 ["boxhD"] = 9573,
 2209 ["boxHD"] = 9574,
 2210 ["boxHu"] = 9575,
 2211 ["boxhU"] = 9576,
 2212 ["boxHU"] = 9577,
 2213 ["boxvH"] = 9578,
 2214 ["boxVh"] = 9579,
 2215 ["boxVH"] = 9580,
 2216 ["uhblk"] = 9600,
 2217 ["lhblk"] = 9604,
 2218 ["block"] = 9608,
 2219 ["blk14"] = 9617,
 2220 ["blk12"] = 9618,
 2221 ["blk34"] = 9619,
 2222 ["squ"] = 9633,
 2223 ["square"] = 9633,
 2224 ["Square"] = 9633,
 2225 ["squf"] = 9642,
 2226 ["squarf"] = 9642,
 2227 ["blacksquare"] = 9642,
 2228 ["FilledVerySmallSquare"] = 9642,
 2229 ["EmptyVerySmallSquare"] = 9643,
 2230 ["rect"] = 9645,
 2231 ["marker"] = 9646,
 2232 ["fltns"] = 9649,
 2233 ["xutri"] = 9651,
 2234 ["bigtriangleup"] = 9651,
 2235 ["utrif"] = 9652,
 2236 ["blacktriangle"] = 9652,
 2237 ["utri"] = 9653,
 2238 ["triangle"] = 9653,

2239 ["rtrif"] = 9656,
2240 ["blacktriangleright"] = 9656,
2241 ["rtri"] = 9657,
2242 ["triangleright"] = 9657,
2243 ["xdtri"] = 9661,
2244 ["bigtriangledown"] = 9661,
2245 ["dtrif"] = 9662,
2246 ["blacktriangledown"] = 9662,
2247 ["dtri"] = 9663,
2248 ["triangledown"] = 9663,
2249 ["ltrif"] = 9666,
2250 ["blacktriangleleft"] = 9666,
2251 ["ltrif"] = 9667,
2252 ["triangleleft"] = 9667,
2253 ["loz"] = 9674,
2254 ["lozenge"] = 9674,
2255 ["cir"] = 9675,
2256 ["tridot"] = 9708,
2257 ["xcirc"] = 9711,
2258 ["bigcirc"] = 9711,
2259 ["ultri"] = 9720,
2260 ["urtri"] = 9721,
2261 ["lltri"] = 9722,
2262 ["EmptySmallSquare"] = 9723,
2263 ["FilledSmallSquare"] = 9724,
2264 ["starf"] = 9733,
2265 ["bigstar"] = 9733,
2266 ["star"] = 9734,
2267 ["phone"] = 9742,
2268 ["female"] = 9792,
2269 ["male"] = 9794,
2270 ["spades"] = 9824,
2271 ["spadesuit"] = 9824,
2272 ["clubs"] = 9827,
2273 ["clubsuit"] = 9827,
2274 ["hearts"] = 9829,
2275 ["heartsuit"] = 9829,
2276 ["diams"] = 9830,
2277 ["diamondsuit"] = 9830,
2278 ["sung"] = 9834,
2279 ["flat"] = 9837,
2280 ["natur"] = 9838,
2281 ["natural"] = 9838,
2282 ["sharp"] = 9839,
2283 ["check"] = 10003,
2284 ["checkmark"] = 10003,
2285 ["cross"] = 10007,

2286 ["malt"] = 10016,
 2287 ["maltese"] = 10016,
 2288 ["sext"] = 10038,
 2289 ["VerticalSeparator"] = 10072,
 2290 ["lbrk"] = 10098,
 2291 ["rbrk"] = 10099,
 2292 ["lobrk"] = 10214,
 2293 ["LeftDoubleBracket"] = 10214,
 2294 ["robrk"] = 10215,
 2295 ["RightDoubleBracket"] = 10215,
 2296 ["lang"] = 10216,
 2297 ["LeftAngleBracket"] = 10216,
 2298 ["langle"] = 10216,
 2299 ["rang"] = 10217,
 2300 ["RightAngleBracket"] = 10217,
 2301 ["rangle"] = 10217,
 2302 ["Lang"] = 10218,
 2303 ["Rang"] = 10219,
 2304 ["loang"] = 10220,
 2305 ["roang"] = 10221,
 2306 ["xlarr"] = 10229,
 2307 ["longleftarrow"] = 10229,
 2308 ["LongLeftArrow"] = 10229,
 2309 ["xrarr"] = 10230,
 2310 ["longrightarrow"] = 10230,
 2311 ["LongRightArrow"] = 10230,
 2312 ["xharr"] = 10231,
 2313 ["longleftrightarrow"] = 10231,
 2314 ["LongLeftRightArrow"] = 10231,
 2315 ["xlArr"] = 10232,
 2316 ["Longleftarrow"] = 10232,
 2317 ["DoubleLongLeftArrow"] = 10232,
 2318 ["xrArr"] = 10233,
 2319 ["Longrightarrow"] = 10233,
 2320 ["DoubleLongRightArrow"] = 10233,
 2321 ["xhArr"] = 10234,
 2322 ["Longleftrightarrow"] = 10234,
 2323 ["DoubleLongLeftRightArrow"] = 10234,
 2324 ["xmap"] = 10236,
 2325 ["longmapsto"] = 10236,
 2326 ["dzigrarr"] = 10239,
 2327 ["nvlArr"] = 10498,
 2328 ["nvrArr"] = 10499,
 2329 ["nvHarr"] = 10500,
 2330 ["Map"] = 10501,
 2331 ["lbarr"] = 10508,
 2332 ["rbarr"] = 10509,

2333 ["bkarow"] = 10509,
 2334 ["lBarr"] = 10510,
 2335 ["rBarr"] = 10511,
 2336 ["dbkarow"] = 10511,
 2337 ["RBarr"] = 10512,
 2338 ["drbkarow"] = 10512,
 2339 ["DDotrahd"] = 10513,
 2340 ["UpArrowBar"] = 10514,
 2341 ["DownArrowBar"] = 10515,
 2342 ["Rarrtl"] = 10518,
 2343 ["latail"] = 10521,
 2344 ["ratail"] = 10522,
 2345 ["lAtail"] = 10523,
 2346 ["rAtail"] = 10524,
 2347 ["larrfs"] = 10525,
 2348 ["rarrfs"] = 10526,
 2349 ["larrbfs"] = 10527,
 2350 ["rarrbfs"] = 10528,
 2351 ["nwarhk"] = 10531,
 2352 ["nearhk"] = 10532,
 2353 ["searhk"] = 10533,
 2354 ["hksearow"] = 10533,
 2355 ["swarhk"] = 10534,
 2356 ["hkswarow"] = 10534,
 2357 ["nwnear"] = 10535,
 2358 ["nesear"] = 10536,
 2359 ["toea"] = 10536,
 2360 ["seswar"] = 10537,
 2361 ["tosa"] = 10537,
 2362 ["swnwar"] = 10538,
 2363 ["rarrc"] = 10547,
 2364 ["cudarr"] = 10549,
 2365 ["ldca"] = 10550,
 2366 ["rdca"] = 10551,
 2367 ["cudarrl"] = 10552,
 2368 ["larrpl"] = 10553,
 2369 ["curarrm"] = 10556,
 2370 ["cularrp"] = 10557,
 2371 ["rarrpl"] = 10565,
 2372 ["harrcir"] = 10568,
 2373 ["Uarrocir"] = 10569,
 2374 ["lurdshar"] = 10570,
 2375 ["ldrushar"] = 10571,
 2376 ["LeftRightVector"] = 10574,
 2377 ["RightUpDownVector"] = 10575,
 2378 ["DownLeftRightVector"] = 10576,
 2379 ["LeftUpDownVector"] = 10577,

2380 ["LeftVectorBar"] = 10578,
 2381 ["RightVectorBar"] = 10579,
 2382 ["RightUpVectorBar"] = 10580,
 2383 ["RightDownVectorBar"] = 10581,
 2384 ["DownLeftVectorBar"] = 10582,
 2385 ["DownRightVectorBar"] = 10583,
 2386 ["LeftUpVectorBar"] = 10584,
 2387 ["LeftDownVectorBar"] = 10585,
 2388 ["LeftTeeVector"] = 10586,
 2389 ["RightTeeVector"] = 10587,
 2390 ["RightUpTeeVector"] = 10588,
 2391 ["RightDownTeeVector"] = 10589,
 2392 ["DownLeftTeeVector"] = 10590,
 2393 ["DownRightTeeVector"] = 10591,
 2394 ["LeftUpTeeVector"] = 10592,
 2395 ["LeftDownTeeVector"] = 10593,
 2396 ["lHar"] = 10594,
 2397 ["uHar"] = 10595,
 2398 ["rHar"] = 10596,
 2399 ["dHar"] = 10597,
 2400 ["luruhar"] = 10598,
 2401 ["ldrdhar"] = 10599,
 2402 ["ruluhar"] = 10600,
 2403 ["rdldhar"] = 10601,
 2404 ["lharul"] = 10602,
 2405 ["llhard"] = 10603,
 2406 ["rharul"] = 10604,
 2407 ["lrhard"] = 10605,
 2408 ["udhar"] = 10606,
 2409 ["UpEquilibrium"] = 10606,
 2410 ["duhar"] = 10607,
 2411 ["ReverseUpEquilibrium"] = 10607,
 2412 ["RoundImplies"] = 10608,
 2413 ["erarr"] = 10609,
 2414 ["simrarr"] = 10610,
 2415 ["larrsim"] = 10611,
 2416 ["rarrsim"] = 10612,
 2417 ["rarrap"] = 10613,
 2418 ["ltlarr"] = 10614,
 2419 ["gtrarr"] = 10616,
 2420 ["subrarr"] = 10617,
 2421 ["suplarr"] = 10619,
 2422 ["lfisht"] = 10620,
 2423 ["rfisht"] = 10621,
 2424 ["ufisht"] = 10622,
 2425 ["dfisht"] = 10623,
 2426 ["lopar"] = 10629,

2427 ["ropar"] = 10630,
2428 ["lbrke"] = 10635,
2429 ["rbrke"] = 10636,
2430 ["lbrkslu"] = 10637,
2431 ["rbrksld"] = 10638,
2432 ["lbrksld"] = 10639,
2433 ["rbrkslu"] = 10640,
2434 ["langd"] = 10641,
2435 ["rangd"] = 10642,
2436 ["lparlt"] = 10643,
2437 ["rpargt"] = 10644,
2438 ["gtlPar"] = 10645,
2439 ["ltrPar"] = 10646,
2440 ["vzigzag"] = 10650,
2441 ["vangrt"] = 10652,
2442 ["angrtvbd"] = 10653,
2443 ["ange"] = 10660,
2444 ["range"] = 10661,
2445 ["dwangle"] = 10662,
2446 ["uwangle"] = 10663,
2447 ["angmsdaa"] = 10664,
2448 ["angmsdab"] = 10665,
2449 ["angmsdac"] = 10666,
2450 ["angmsdad"] = 10667,
2451 ["angmsdae"] = 10668,
2452 ["angmsdaf"] = 10669,
2453 ["angmsdag"] = 10670,
2454 ["angmsdah"] = 10671,
2455 ["bemptyv"] = 10672,
2456 ["demptyv"] = 10673,
2457 ["cemptyv"] = 10674,
2458 ["raemptyv"] = 10675,
2459 ["laemptyv"] = 10676,
2460 ["ohbar"] = 10677,
2461 ["omid"] = 10678,
2462 ["opar"] = 10679,
2463 ["operp"] = 10681,
2464 ["olcross"] = 10683,
2465 ["odsold"] = 10684,
2466 ["olcir"] = 10686,
2467 ["ofcir"] = 10687,
2468 ["olt"] = 10688,
2469 ["ogt"] = 10689,
2470 ["cirscir"] = 10690,
2471 ["cirE"] = 10691,
2472 ["solb"] = 10692,
2473 ["bsolb"] = 10693,

2474 ["boxbox"] = 10697,
 2475 ["trishb"] = 10701,
 2476 ["rtriltri"] = 10702,
 2477 ["LeftTriangleBar"] = 10703,
 2478 ["RightTriangleBar"] = 10704,
 2479 ["race"] = 10714,
 2480 ["iinfin"] = 10716,
 2481 ["infintie"] = 10717,
 2482 ["nvinfin"] = 10718,
 2483 ["eparsl"] = 10723,
 2484 ["smeparsl"] = 10724,
 2485 ["eqvparsl"] = 10725,
 2486 ["lozf"] = 10731,
 2487 ["blacklozenge"] = 10731,
 2488 ["RuleDelayed"] = 10740,
 2489 ["dsol"] = 10742,
 2490 ["xodot"] = 10752,
 2491 ["bigodot"] = 10752,
 2492 ["xoplus"] = 10753,
 2493 ["bigoplus"] = 10753,
 2494 ["xotime"] = 10754,
 2495 ["bigotimes"] = 10754,
 2496 ["xuplus"] = 10756,
 2497 ["biguplus"] = 10756,
 2498 ["xsqcup"] = 10758,
 2499 ["bigsqcup"] = 10758,
 2500 ["qint"] = 10764,
 2501 ["iiiint"] = 10764,
 2502 ["fpartint"] = 10765,
 2503 ["cirfnint"] = 10768,
 2504 ["awint"] = 10769,
 2505 ["rppolint"] = 10770,
 2506 ["scpolint"] = 10771,
 2507 ["npolint"] = 10772,
 2508 ["pointint"] = 10773,
 2509 ["quatint"] = 10774,
 2510 ["intlarhk"] = 10775,
 2511 ["pluscir"] = 10786,
 2512 ["plusacir"] = 10787,
 2513 ["simplus"] = 10788,
 2514 ["plusdu"] = 10789,
 2515 ["plussim"] = 10790,
 2516 ["plustwo"] = 10791,
 2517 ["mcomma"] = 10793,
 2518 ["minusdu"] = 10794,
 2519 ["loplus"] = 10797,
 2520 ["roplus"] = 10798,

2521 ["Cross"] = 10799,
2522 ["timesd"] = 10800,
2523 ["timesbar"] = 10801,
2524 ["smashp"] = 10803,
2525 ["lotimes"] = 10804,
2526 ["rotimes"] = 10805,
2527 ["otimesas"] = 10806,
2528 ["Otimes"] = 10807,
2529 ["odiv"] = 10808,
2530 ["triplus"] = 10809,
2531 ["triminus"] = 10810,
2532 ["tritime"] = 10811,
2533 ["iproduct"] = 10812,
2534 ["intprod"] = 10812,
2535 ["amalg"] = 10815,
2536 ["capdot"] = 10816,
2537 ["ncup"] = 10818,
2538 ["ncap"] = 10819,
2539 ["capand"] = 10820,
2540 ["cupor"] = 10821,
2541 ["cupcap"] = 10822,
2542 ["capcup"] = 10823,
2543 ["cupbrcap"] = 10824,
2544 ["capbrcup"] = 10825,
2545 ["cupcup"] = 10826,
2546 ["capcap"] = 10827,
2547 ["ccups"] = 10828,
2548 ["ccaps"] = 10829,
2549 ["ccupssm"] = 10832,
2550 ["And"] = 10835,
2551 ["Or"] = 10836,
2552 ["andand"] = 10837,
2553 ["oror"] = 10838,
2554 ["orslope"] = 10839,
2555 ["andslope"] = 10840,
2556 ["andv"] = 10842,
2557 ["orv"] = 10843,
2558 ["andd"] = 10844,
2559 ["ord"] = 10845,
2560 ["wedbar"] = 10847,
2561 ["sdote"] = 10854,
2562 ["simdot"] = 10858,
2563 ["congdote"] = 10861,
2564 ["easter"] = 10862,
2565 ["apacir"] = 10863,
2566 ["apE"] = 10864,
2567 ["eplus"] = 10865,

2568 ["pluse"] = 10866,
 2569 ["Esim"] = 10867,
 2570 ["Colone"] = 10868,
 2571 ["Equal"] = 10869,
 2572 ["eDDot"] = 10871,
 2573 ["ddotseq"] = 10871,
 2574 ["equivDD"] = 10872,
 2575 ["ltcir"] = 10873,
 2576 ["gtcir"] = 10874,
 2577 ["ltquest"] = 10875,
 2578 ["gtquest"] = 10876,
 2579 ["les"] = 10877,
 2580 ["LessSlantEqual"] = 10877,
 2581 ["leqslant"] = 10877,
 2582 ["ges"] = 10878,
 2583 ["GreaterSlantEqual"] = 10878,
 2584 ["geqslant"] = 10878,
 2585 ["lesdot"] = 10879,
 2586 ["gesdot"] = 10880,
 2587 ["lesdoto"] = 10881,
 2588 ["gesdoto"] = 10882,
 2589 ["lesdotor"] = 10883,
 2590 ["gesdoto1"] = 10884,
 2591 ["lap"] = 10885,
 2592 ["lessapprox"] = 10885,
 2593 ["gap"] = 10886,
 2594 ["gtrapprox"] = 10886,
 2595 ["lne"] = 10887,
 2596 ["lneq"] = 10887,
 2597 ["gne"] = 10888,
 2598 ["gneq"] = 10888,
 2599 ["lnap"] = 10889,
 2600 ["lnapprox"] = 10889,
 2601 ["gnap"] = 10890,
 2602 ["gnapprox"] = 10890,
 2603 ["lEg"] = 10891,
 2604 ["lesseqqgtr"] = 10891,
 2605 ["gEl"] = 10892,
 2606 ["gtreqqless"] = 10892,
 2607 ["lsime"] = 10893,
 2608 ["gsime"] = 10894,
 2609 ["lsimg"] = 10895,
 2610 ["gsiml"] = 10896,
 2611 ["lgE"] = 10897,
 2612 ["glE"] = 10898,
 2613 ["lesges"] = 10899,
 2614 ["gesles"] = 10900,

2615 ["els"] = 10901,
 2616 ["eqslantless"] = 10901,
 2617 ["egs"] = 10902,
 2618 ["eqslantgtr"] = 10902,
 2619 ["elsdot"] = 10903,
 2620 ["egsdot"] = 10904,
 2621 ["el"] = 10905,
 2622 ["eg"] = 10906,
 2623 ["siml"] = 10909,
 2624 ["simg"] = 10910,
 2625 ["simLE"] = 10911,
 2626 ["simGE"] = 10912,
 2627 ["LessLess"] = 10913,
 2628 ["GreaterGreater"] = 10914,
 2629 ["glj"] = 10916,
 2630 ["gla"] = 10917,
 2631 ["ltcc"] = 10918,
 2632 ["gtcc"] = 10919,
 2633 ["lescc"] = 10920,
 2634 ["gescc"] = 10921,
 2635 ["smt"] = 10922,
 2636 ["lat"] = 10923,
 2637 ["smte"] = 10924,
 2638 ["late"] = 10925,
 2639 ["bumpE"] = 10926,
 2640 ["pre"] = 10927,
 2641 ["preceq"] = 10927,
 2642 ["PrecedesEqual"] = 10927,
 2643 ["sce"] = 10928,
 2644 ["succeq"] = 10928,
 2645 ["SucceedsEqual"] = 10928,
 2646 ["prE"] = 10931,
 2647 ["scE"] = 10932,
 2648 ["prnE"] = 10933,
 2649 ["precneqq"] = 10933,
 2650 ["scnE"] = 10934,
 2651 ["succneqq"] = 10934,
 2652 ["prap"] = 10935,
 2653 ["precapprox"] = 10935,
 2654 ["scap"] = 10936,
 2655 ["succapprox"] = 10936,
 2656 ["prnap"] = 10937,
 2657 ["precnapprox"] = 10937,
 2658 ["scnap"] = 10938,
 2659 ["succnapprox"] = 10938,
 2660 ["Pr"] = 10939,
 2661 ["Sc"] = 10940,

2662 ["subdot"] = 10941,
 2663 ["supdot"] = 10942,
 2664 ["subplus"] = 10943,
 2665 ["supplus"] = 10944,
 2666 ["submult"] = 10945,
 2667 ["supmult"] = 10946,
 2668 ["subedot"] = 10947,
 2669 ["supedot"] = 10948,
 2670 ["subE"] = 10949,
 2671 ["subseteqq"] = 10949,
 2672 ["supE"] = 10950,
 2673 ["supseteqq"] = 10950,
 2674 ["subsim"] = 10951,
 2675 ["supsim"] = 10952,
 2676 ["subnE"] = 10955,
 2677 ["subsetneqq"] = 10955,
 2678 ["supnE"] = 10956,
 2679 ["supsetneqq"] = 10956,
 2680 ["csub"] = 10959,
 2681 ["csup"] = 10960,
 2682 ["csube"] = 10961,
 2683 ["csupe"] = 10962,
 2684 ["subsup"] = 10963,
 2685 ["supsup"] = 10964,
 2686 ["subsub"] = 10965,
 2687 ["supsup"] = 10966,
 2688 ["suphsub"] = 10967,
 2689 ["supdsub"] = 10968,
 2690 ["forkv"] = 10969,
 2691 ["topfork"] = 10970,
 2692 ["mlcp"] = 10971,
 2693 ["Dashv"] = 10980,
 2694 ["DoubleLeftTee"] = 10980,
 2695 ["Vdashl"] = 10982,
 2696 ["Barv"] = 10983,
 2697 ["vBar"] = 10984,
 2698 ["vBarv"] = 10985,
 2699 ["Vbar"] = 10987,
 2700 ["Not"] = 10988,
 2701 ["bNot"] = 10989,
 2702 ["rnmid"] = 10990,
 2703 ["cirmid"] = 10991,
 2704 ["midcir"] = 10992,
 2705 ["topcir"] = 10993,
 2706 ["nhpar"] = 10994,
 2707 ["parsim"] = 10995,
 2708 ["parssl"] = 11005,

2709 ["fflig"] = 64256,
2710 ["filig"] = 64257,
2711 ["fllig"] = 64258,
2712 ["ffilig"] = 64259,
2713 ["ffllig"] = 64260,
2714 ["Ascr"] = 119964,
2715 ["Cscr"] = 119966,
2716 ["Dscr"] = 119967,
2717 ["Gscr"] = 119970,
2718 ["Jscr"] = 119973,
2719 ["Kscr"] = 119974,
2720 ["Nscr"] = 119977,
2721 ["Oscr"] = 119978,
2722 ["Pscr"] = 119979,
2723 ["Qscr"] = 119980,
2724 ["Sscr"] = 119982,
2725 ["Tscr"] = 119983,
2726 ["Uscr"] = 119984,
2727 ["Vscr"] = 119985,
2728 ["Wscr"] = 119986,
2729 ["Xscr"] = 119987,
2730 ["Yscr"] = 119988,
2731 ["Zscr"] = 119989,
2732 ["ascr"] = 119990,
2733 ["bscr"] = 119991,
2734 ["cscr"] = 119992,
2735 ["dscr"] = 119993,
2736 ["fscr"] = 119995,
2737 ["hscr"] = 119997,
2738 ["iscr"] = 119998,
2739 ["jscr"] = 119999,
2740 ["kscr"] = 120000,
2741 ["lscr"] = 120001,
2742 ["mscr"] = 120002,
2743 ["nscr"] = 120003,
2744 ["pscr"] = 120005,
2745 ["qscr"] = 120006,
2746 ["rscr"] = 120007,
2747 ["sscr"] = 120008,
2748 ["tscr"] = 120009,
2749 ["uscr"] = 120010,
2750 ["vscr"] = 120011,
2751 ["wscr"] = 120012,
2752 ["xscr"] = 120013,
2753 ["yscr"] = 120014,
2754 ["zscr"] = 120015,
2755 ["Afr"] = 120068,

2756 ["Bfr"] = 120069,
2757 ["Dfr"] = 120071,
2758 ["Efr"] = 120072,
2759 ["Ffr"] = 120073,
2760 ["Gfr"] = 120074,
2761 ["Jfr"] = 120077,
2762 ["Kfr"] = 120078,
2763 ["Lfr"] = 120079,
2764 ["Mfr"] = 120080,
2765 ["Nfr"] = 120081,
2766 ["Ofr"] = 120082,
2767 ["Pfr"] = 120083,
2768 ["Qfr"] = 120084,
2769 ["Sfr"] = 120086,
2770 ["Tfr"] = 120087,
2771 ["Ufr"] = 120088,
2772 ["Vfr"] = 120089,
2773 ["Wfr"] = 120090,
2774 ["Xfr"] = 120091,
2775 ["Yfr"] = 120092,
2776 ["afr"] = 120094,
2777 ["bfr"] = 120095,
2778 ["cfr"] = 120096,
2779 ["dfr"] = 120097,
2780 ["efr"] = 120098,
2781 ["ffr"] = 120099,
2782 ["gfr"] = 120100,
2783 ["hfr"] = 120101,
2784 ["ifr"] = 120102,
2785 ["jfr"] = 120103,
2786 ["kfr"] = 120104,
2787 ["lfr"] = 120105,
2788 ["mfr"] = 120106,
2789 ["nfr"] = 120107,
2790 ["ofr"] = 120108,
2791 ["pfr"] = 120109,
2792 ["qfr"] = 120110,
2793 ["rfr"] = 120111,
2794 ["sfr"] = 120112,
2795 ["tfr"] = 120113,
2796 ["ufr"] = 120114,
2797 ["vfr"] = 120115,
2798 ["wfr"] = 120116,
2799 ["xfr"] = 120117,
2800 ["yfr"] = 120118,
2801 ["zfr"] = 120119,
2802 ["Aopf"] = 120120,

```
2803 ["Bopf"] = 120121,  
2804 ["Dopf"] = 120123,  
2805 ["Eopf"] = 120124,  
2806 ["Fopf"] = 120125,  
2807 ["Gopf"] = 120126,  
2808 ["Iopf"] = 120128,  
2809 ["Jopf"] = 120129,  
2810 ["Kopf"] = 120130,  
2811 ["Lopf"] = 120131,  
2812 ["Mopf"] = 120132,  
2813 ["Oopf"] = 120134,  
2814 ["Sopf"] = 120138,  
2815 ["Topf"] = 120139,  
2816 ["Uopf"] = 120140,  
2817 ["Vopf"] = 120141,  
2818 ["Wopf"] = 120142,  
2819 ["Xopf"] = 120143,  
2820 ["Yopf"] = 120144,  
2821 ["aopf"] = 120146,  
2822 ["bopf"] = 120147,  
2823 ["copf"] = 120148,  
2824 ["dopf"] = 120149,  
2825 ["eopf"] = 120150,  
2826 ["fopf"] = 120151,  
2827 ["gopf"] = 120152,  
2828 ["hopf"] = 120153,  
2829 ["iopf"] = 120154,  
2830 ["jopf"] = 120155,  
2831 ["kopf"] = 120156,  
2832 ["lopf"] = 120157,  
2833 ["mopf"] = 120158,  
2834 ["nopf"] = 120159,  
2835 ["oopf"] = 120160,  
2836 ["popf"] = 120161,  
2837 ["qopf"] = 120162,  
2838 ["ropf"] = 120163,  
2839 ["sopf"] = 120164,  
2840 ["topf"] = 120165,  
2841 ["uopf"] = 120166,  
2842 ["vopf"] = 120167,  
2843 ["wopf"] = 120168,  
2844 ["xopf"] = 120169,  
2845 ["yopf"] = 120170,  
2846 ["zopf"] = 120171,  
2847 }
```

Given a string `s` of decimal digits, the `entities.dec_entity` returns the corresponding UTF8-encoded Unicode codepoint.

```
2848 function entities.dec_entity(s)
2849   return unicode.utf8.char(tonumber(s))
2850 end
```

Given a string `s` of hexadecimal digits, the `entities.hex_entity` returns the corresponding UTF8-encoded Unicode codepoint.

```
2851 function entities.hex_entity(s)
2852   return unicode.utf8.char(tonumber("0x"..s))
2853 end
```

Given a character entity name `s` (like `ouml`), the `entities.char_entity` returns the corresponding UTF8-encoded Unicode codepoint.

```
2854 function entities.char_entity(s)
2855   local n = character_entities[s]
2856   if n == nil then
2857     return "&" .. s .. ";"
2858   end
2859   return unicode.utf8.char(n)
2860 end
```

3.1.3 Plain T_EX Writer

This section documents the `writer` object, which implements the routines for producing the T_EX output. The object is an amalgamate of the generic, T_EX, L^AT_EX writer objects that were located in the `lunamark/writer/generic.lua`, `lunamark/writer/tex.lua`, and `lunamark/writer/latex.lua` files in the Luna-mark Lua module.

Although not specified in the Lua interface (see Section 2.1), the `writer` object is exported, so that the curious user could easily tinker with the methods of the objects produced by the `writer.new` method described below. The user should be aware, however, that the implementation may change in a future revision.

```
2861 M.writer = {}
```

The `writer.new` method creates and returns a new T_EX writer object associated with the Lua interface options (see Section 2.1.2) `options`. When `options` are unspecified, it is assumed that an empty table was passed to the method.

The objects produced by the `writer.new` method expose instance methods and variables of their own. As a convention, I will refer to these *member*s as `writer->member`.

```
2862 function M.writer.new(options)
2863   local self = {}
2864   options = options or {}
```

Make the `options` table inherit from the `defaultOptions` table.

```
2865 setmetatable(options, { __index = function (_, key)
2866     return defaultOptions[key] end })
```

Parse the `slice` option and define `writer->slice_begin` `writer->slice_end`, and `writer->is_writing`.

```
2867 local slice_specifiers = {}
2868 for specifier in options.slice:gmatch("[^%s]+") do
2869     table.insert(slice_specifiers, specifier)
2870 end
2871
2872 if #slice_specifiers == 2 then
2873     self.slice_begin, self.slice_end = table.unpack(slice_specifiers)
2874     local slice_begin_type = self.slice_begin:sub(1, 1)
2875     if slice_begin_type ~= "^" and slice_begin_type ~= "$" then
2876         self.slice_begin = "^" .. self.slice_begin
2877     end
2878     local slice_end_type = self.slice_end:sub(1, 1)
2879     if slice_end_type ~= "^" and slice_end_type ~= "$" then
2880         self.slice_end = "$" .. self.slice_end
2881     end
2882 elseif #slice_specifiers == 1 then
2883     self.slice_begin = "^" .. slice_specifiers[1]
2884     self.slice_end = "$" .. slice_specifiers[1]
2885 end
2886
2887 if self.slice_begin == "^" and self.slice_end ~= "^" then
2888     self.is_writing = true
2889 else
2890     self.is_writing = false
2891 end
```

Define `writer->suffix` as the suffix of the produced cache files.

```
2892 self.suffix = ".tex"
```

Define `writer->space` as the output format of a space character.

```
2893 self.space = " "
```

Define `writer->nbsp` as the output format of a non-breaking space character.

```
2894 self.nbsp = "\\markdownRendererNbsp{}"
```

Define `writer->plain` as a function that will transform an input plain text block `s` to the output format.

```
2895 function self.plain(s)
2896     return s
2897 end
```

Define `writer->paragraph` as a function that will transform an input paragraph `s` to the output format.

```

2898 function self.paragraph(s)
2899     if not self.is_writing then return "" end
2900     return s
2901 end

```

Define `writer->pack` as a function that will take the filename `name` of the output file prepared by the reader and transform it to the output format.

```

2902 function self.pack(name)
2903     return [[\input ]] .. name .. [[\relax{}]]
2904 end

```

Define `writer->interblocksep` as the output format of a block element separator.

```

2905 function self.interblocksep()
2906     if not self.is_writing then return "" end
2907     return "\\markdownRendererInterblockSeparator\n{}"
2908 end

```

Define `writer->eof` as the end of file marker in the output format.

```

2909 self.eof = [[\relax]]

```

Define `writer->linebreak` as the output format of a forced line break.

```

2910 self.linebreak = "\\markdownRendererLineBreak\n{}"

```

Define `writer->ellipsis` as the output format of an ellipsis.

```

2911 self.ellipsis = "\\markdownRendererEllipsis{}"

```

Define `writer->hrule` as the output format of a horizontal rule.

```

2912 function self.hrule()
2913     if not self.is_writing then return "" end
2914     return "\\markdownRendererHorizontalRule{}"
2915 end

```

Define a table `escaped_chars` containing the mapping from special plain T_EX characters (including the active pipe character (|) of ConT_EXt) to their escaped variants. Define tables `escaped_minimal_chars` and `escaped_minimal_strings` containing the mapping from special plain characters and character strings that need to be escaped even in content that will not be typeset.

```

2916 local escaped_chars = {
2917     [{"{"}] = "\\markdownRendererLeftBrace{}",
2918     [{"}"}] = "\\markdownRendererRightBrace{}",
2919     [{"$"}] = "\\markdownRendererDollarSign{}",
2920     [{"%"}] = "\\markdownRendererPercentSign{}",
2921     [{"&"}] = "\\markdownRendererAmpersand{}",
2922     [{"_"}] = "\\markdownRendererUnderscore{}",
2923     [{"#"}] = "\\markdownRendererHash{}",
2924     [{"^"}] = "\\markdownRendererCircumflex{}",
2925     [{"\"}] = "\\markdownRendererBackslash{}",
2926     [{"~"}] = "\\markdownRendererTilde{}",
2927     [{"|"}] = "\\markdownRendererPipe{}",

```

```

2928   }
2929   local escaped_uri_chars = {
2930     [{""] = "\\markdownRendererLeftBrace{}"},
2931     ["}"] = "\\markdownRendererRightBrace{}"},
2932     [%"] = "\\markdownRendererPercentSign{}"},
2933     [ "\\"] = "\\markdownRendererBackslash{}"},
2934   }
2935   local escaped_citation_chars = {
2936     [{""] = "\\markdownRendererLeftBrace{}"},
2937     ["}"] = "\\markdownRendererRightBrace{}"},
2938     [%"] = "\\markdownRendererPercentSign{}"},
2939     [#"] = "\\markdownRendererHash{}"},
2940     [ "\\"] = "\\markdownRendererBackslash{}"},
2941   }
2942   local escaped_minimal_strings = {
2943     [{"^"}] = "\\markdownRendererCircumflex\\markdownRendererCircumflex ",
2944   }

```

Use the `escaped_chars` table to create an escaper function `escape` and the `escaped_minimal_chars` and `escaped_minimal_strings` tables to create an escaper function `escape_minimal`.

```

2945   local escape = util.escaper(escaped_chars)
2946   local escape_citation = util.escaper(escaped_citation_chars,
2947     escaped_minimal_strings)
2948   local escape_uri = util.escaper(escaped_uri_chars, escaped_minimal_strings)

```

Define `writer->string` as a function that will transform an input plain text span `s` to the output format and `writer->uri` as a function that will transform an input URI `u` to the output format. If the `hybrid` option is `true`, use identity functions. Otherwise, use the `escape` and `escape_minimal` functions.

```

2949   if options.hybrid then
2950     self.string = function(s) return s end
2951     self.citation = function(c) return c end
2952     self.uri = function(u) return u end
2953   else
2954     self.string = escape
2955     self.citation = escape_citation
2956     self.uri = escape_uri
2957   end

```

Define `writer->code` as a function that will transform an input inlined code span `s` to the output format.

```

2958   function self.code(s)
2959     return {"\\markdownRendererCodeSpan{", escape(s), "}"}
2960   end

```

Define `writer->link` as a function that will transform an input hyperlink to the output format, where `lab` corresponds to the label, `src` to URI, and `tit` to the title of the link.

```

2961 function self.link(lab,src,tit)
2962     return {"\\markdownRendererLink{" ,lab,"} ",
2963             "{" ,self.string(src),"} ",
2964             "{" ,self.uri(src),"} ",
2965             "{" ,self.string(tit or ""),"} "}
2966 end

```

Define `writer->table` as a function that will transform an input table to the output format, where `rows` is a sequence of columns and a column is a sequence of cell texts.

```

2967 function self.table(rows, caption)
2968     local buffer = {"\\markdownRendererTable{" ,
2969                     caption or "" , "}" , #rows - 1 , "}" , #rows[1] , "}" }
2970     local temp = rows[2] -- put alignments on the first row
2971     rows[2] = rows[1]
2972     rows[1] = temp
2973     for i, row in ipairs(rows) do
2974         table.insert(buffer, "{" )
2975         for _, column in ipairs(row) do
2976             if i > 1 then -- do not use braces for alignments
2977                 table.insert(buffer, "{" )
2978             end
2979             table.insert(buffer, column)
2980             if i > 1 then
2981                 table.insert(buffer, "%\n" )
2982             end
2983         end
2984         table.insert(buffer, "%\n" )
2985     end
2986     return buffer
2987 end

```

Define `writer->image` as a function that will transform an input image to the output format, where `lab` corresponds to the label, `src` to the URL, and `tit` to the title of the image.

```

2988 function self.image(lab,src,tit)
2989     return {"\\markdownRendererImage{" ,lab,"} ",
2990             "{" ,self.string(src),"} ",
2991             "{" ,self.uri(src),"} ",
2992             "{" ,self.string(tit or ""),"} "}
2993 end

```

The `languages_json` table maps programming language filename extensions to fence infostrings. All `options.contentBlocksLanguageMap` files located by `kpathsea`

are loaded into a chain of tables. `languages_json` corresponds to the first table and is chained with the rest via Lua metatables.

```

2994 local languages_json = (function()
2995     local kpse = require("kpse")
2996     kpse.set_program_name("luatex")
2997     local base, prev, curr
2998     for _, file in ipairs{kpse.lookup(options.contentBlocksLanguageMap,
2999                               { all=true })} do
3000         json = io.open(file, "r"):read("*all")
3001                :gsub('"[^\\n]-"', '%1=')
3002         curr = (function()
3003             local _ENV={ json=json, load=load } -- run in sandbox
3004             return load("return ..json")()
3005         end)()
3006         if type(curr) == "table" then
3007             if base == nil then
3008                 base = curr
3009             else
3010                 setmetatable(prev, { __index = curr })
3011             end
3012             prev = curr
3013         end
3014     end
3015     return base or {}
3016 end)()

```

Define `writer->contentblock` as a function that will transform an input iA Writer content block to the output format, where `src` corresponds to the URI prefix, `suf` to the URI extension, `type` to the type of the content block (`localfile` or `onlineimage`), and `tit` to the title of the content block.

```

3017 function self.contentblock(src,suf,type,tit)
3018     if not self.is_writing then return "" end
3019     src = src..".."..suf
3020     suf = suf:lower()
3021     if type == "onlineimage" then
3022         return {"\\markdownRendererContentBlockOnlineImage{"..suf.."},"",
3023                "{"..self.string(src).."},"",
3024                "{"..self.uri(src).."},"",
3025                "{"..self.string(tit or "").."},""}
3026     elseif languages_json[suf] then
3027         return {"\\markdownRendererContentBlockCode{"..suf.."},"",
3028                "{"..self.string(languages_json[suf]).."},"",
3029                "{"..self.string(src).."},"",
3030                "{"..self.uri(src).."},"",
3031                "{"..self.string(tit or "").."},""}
3032     else
3033         return {"\\markdownRendererContentBlock{"..suf.."},"",

```



```

3034         "{,self.string(src),"}",
3035         "{,self.uri(src),}",
3036         "{,self.string(tit or ""),}"
3037     end
3038 end

```

Define `writer->bulletlist` as a function that will transform an input bulleted list to the output format, where `items` is an array of the list items and `tight` specifies, whether the list is tight or not.

```

3039 local function ulitem(s)
3040     return {"\markdownRenderUlItem ",s,
3041           "\markdownRenderUlItemEnd "}
3042 end
3043
3044 function self.bulletlist(items,tight)
3045     if not self.is_writing then return "" end
3046     local buffer = {}
3047     for _,item in ipairs(items) do
3048         buffer[#buffer + 1] = ulitem(item)
3049     end
3050     local contents = util.intersperse(buffer,"\n")
3051     if tight and options.tightLists then
3052         return {"\markdownRenderUlBeginTight\n",contents,
3053               "\n\markdownRenderUlEndTight "}
3054     else
3055         return {"\markdownRenderUlBegin\n",contents,
3056               "\n\markdownRenderUlEnd "}
3057     end
3058 end

```

Define `writer->ollist` as a function that will transform an input ordered list to the output format, where `items` is an array of the list items and `tight` specifies, whether the list is tight or not. If the optional parameter `startnum` is present, it should be used as the number of the first list item.

```

3059 local function olitem(s,num)
3060     if num ~= nil then
3061         return {"\markdownRenderOlItemWithNumber{" ,num,"}",s,
3062               "\markdownRenderOlItemEnd "}
3063     else
3064         return {"\markdownRenderOlItem ",s,
3065               "\markdownRenderOlItemEnd "}
3066     end
3067 end
3068
3069 function self.orderedlist(items,tight,startnum)
3070     if not self.is_writing then return "" end
3071     local buffer = {}

```

```

3072     local num = startnum
3073     for _,item in ipairs(items) do
3074         buffer[#buffer + 1] = olitem(item,num)
3075         if num ~= nil then
3076             num = num + 1
3077         end
3078     end
3079     local contents = util.intersperse(buffer,"\n")
3080     if tight and options.tightLists then
3081         return {"\\markdownRendererOlBeginTight\n",contents,
3082             "\n\\markdownRendererOlEndTight "}
3083     else
3084         return {"\\markdownRendererOlBegin\n",contents,
3085             "\n\\markdownRendererOlEnd "}
3086     end
3087 end

```

Define `writer->inline_html` and `writer->display_html` as functions that will transform an inline or block HTML element respectively to the output format, where `html` is the HTML input.

```

3088     function self.inline_html(html) return "" end
3089     function self.display_html(html) return "" end

```

Define `writer->definitionlist` as a function that will transform an input definition list to the output format, where `items` is an array of tables, each of the form `{ term = t, definitions = defs }`, where `t` is a term and `defs` is an array of definitions. `tight` specifies, whether the list is tight or not.

```

3090     local function dlitem(term, defs)
3091         local retVal = {"\\markdownRendererDlItem{",term,"}"}
3092         for _, def in ipairs(defs) do
3093             retVal[#retVal+1] = {"\\markdownRendererDlDefinitionBegin ",def,
3094                 "\\markdownRendererDlDefinitionEnd "}
3095         end
3096         retVal[#retVal+1] = "\\markdownRendererDlItemEnd "
3097         return retVal
3098     end
3099
3100     function self.definitionlist(items,tight)
3101         if not self.is_writing then return "" end
3102         local buffer = {}
3103         for _,item in ipairs(items) do
3104             buffer[#buffer + 1] = dlitem(item.term, item.definitions)
3105         end
3106         if tight and options.tightLists then
3107             return {"\\markdownRendererDlBeginTight\n", buffer,
3108                 "\n\\markdownRendererDlEndTight"}
3109         else

```

```

3110     return {"\\markdownRendererDlBegin\n", buffer,
3111           "\n\\markdownRendererDlEnd"}
3112   end
3113 end

```

Define `writer->emphasis` as a function that will transform an emphasized span `s` of input text to the output format.

```

3114 function self.emphasis(s)
3115   return {"\\markdownRendererEmphasis{",s,""}
3116 end

```

Define `writer->strong` as a function that will transform a strongly emphasized span `s` of input text to the output format.

```

3117 function self.strong(s)
3118   return {"\\markdownRendererStrongEmphasis{",s,""}
3119 end

```

Define `writer->blockquote` as a function that will transform an input block quote `s` to the output format.

```

3120 function self.blockquote(s)
3121   if #util.ropetostring(s) == 0 then return "" end
3122   return {"\\markdownRendererBlockQuoteBegin\n",s,
3123         "\n\\markdownRendererBlockQuoteEnd "}
3124 end

```

Define `writer->verbatim` as a function that will transform an input code block `s` to the output format.

```

3125 function self.verbatim(s)
3126   if not self.is_writing then return "" end
3127   s = string.gsub(s, '[\r\n%$]*$', '')
3128   local name = util.cache(options.cacheDir, s, nil, nil, ".verbatim")
3129   return {"\\markdownRendererInputVerbatim{",name,""}
3130 end

```

Define `writer->codeFence` as a function that will transform an input fenced code block `s` with the infostring `i` to the output format.

```

3131 function self.fencedCode(i, s)
3132   if not self.is_writing then return "" end
3133   s = string.gsub(s, '[\r\n%$]*$', '')
3134   local name = util.cache(options.cacheDir, s, nil, nil, ".verbatim")
3135   return {"\\markdownRendererInputFencedCode{",name,""}{"",i,""}
3136 end

```

Define `writer->active_headings` as a stack of identifiers of the headings that are currently active.

```

3137 self.active_headings = {}

```

Define `writer->heading` as a function that will transform an input heading `s` at level `level` with identifiers `identifiers` to the output format.

```

3138 function self.heading(s,level,attributes)
3139     local active_headings = self.active_headings
3140     local slice_begin_type = self.slice_begin:sub(1, 1)
3141     local slice_begin_identifier = self.slice_begin:sub(2) or ""
3142     local slice_end_type = self.slice_end:sub(1, 1)
3143     local slice_end_identifier = self.slice_end:sub(2) or ""
3144
3145     while #active_headings < level do
3146         -- push empty identifiers for implied sections
3147         table.insert(active_headings, {})
3148     end
3149
3150     while #active_headings >= level do
3151         -- pop identifiers for sections that have ended
3152         local active_identifiers = active_headings[#active_headings]
3153         if active_identifiers[slice_begin_identifier] ~= nil
3154             and slice_begin_type == "$" then
3155             self.is_writing = true
3156         end
3157         if active_identifiers[slice_end_identifier] ~= nil
3158             and slice_end_type == "$" then
3159             self.is_writing = false
3160         end
3161         table.remove(active_headings, #active_headings)
3162     end
3163
3164     -- push identifiers for the new section
3165     attributes = attributes or {}
3166     local identifiers = {}
3167     for index = 1, #attributes do
3168         attribute = attributes[index]
3169         identifiers[attribute:sub(2)] = true
3170     end
3171     if identifiers[slice_begin_identifier] ~= nil
3172         and slice_begin_type == "^" then
3173         self.is_writing = true
3174     end
3175     if identifiers[slice_end_identifier] ~= nil
3176         and slice_end_type == "^" then
3177         self.is_writing = false
3178     end
3179     table.insert(active_headings, identifiers)
3180
3181     if not self.is_writing then return "" end
3182
3183     local cmd
3184     level = level + options.shiftHeadings

```

```

3185     if level <= 1 then
3186         cmd = "\\markdownRendererHeadingOne"
3187     elseif level == 2 then
3188         cmd = "\\markdownRendererHeadingTwo"
3189     elseif level == 3 then
3190         cmd = "\\markdownRendererHeadingThree"
3191     elseif level == 4 then
3192         cmd = "\\markdownRendererHeadingFour"
3193     elseif level == 5 then
3194         cmd = "\\markdownRendererHeadingFive"
3195     elseif level >= 6 then
3196         cmd = "\\markdownRendererHeadingSix"
3197     else
3198         cmd = ""
3199     end
3200     return {cmd, "{" ,s, "}"}
3201 end

```

Define `writer->note` as a function that will transform an input footnote `s` to the output format.

```

3202 function self.note(s)
3203     return {"\\markdownRendererFootnote{" ,s, "}"}
3204 end

```

Define `writer->citations` as a function that will transform an input array of citations `cites` to the output format. If `text_cites` is `true`, the citations should be rendered in-text, when applicable. The `cites` array contains tables with the following keys and values:

- `suppress_author` – If the value of the key is true, then the author of the work should be omitted in the citation, when applicable.
- `prenote` – The value of the key is either `nil` or a rope that should be inserted before the citation.
- `postnote` – The value of the key is either `nil` or a rope that should be inserted after the citation.
- `name` – The value of this key is the citation name.

```

3205 function self.citations(text_cites, cites)
3206     local buffer = {"\\markdownRenderer", text_cites and "TextCite" or "Cite",
3207         "{" , #cites, "}"}
3208     for _,cite in ipairs(cites) do
3209         buffer[#buffer+1] = {cite.suppress_author and "-" or "+", "{" ,
3210             cite.prenote or "", "}{" , cite.postnote or "", "}{" , cite.name, "}"}
3211     end
3212     return buffer

```

```
3213 end
3214
3215 return self
3216 end
```

3.1.4 Parsers

The `parsers` hash table stores PEG patterns that are static and can be reused between different `reader` objects.

```
3217 local parsers = {}
```

3.1.4.1 Basic Parsers

```
3218 parsers.percent = P("%")
3219 parsers.at = P("@")
3220 parsers.comma = P(",")
3221 parsers.asterisk = P("*")
3222 parsers.dash = P("-")
3223 parsers.plus = P("+")
3224 parsers.underscore = P("_")
3225 parsers.period = P(".")
3226 parsers.hash = P("#")
3227 parsers.ampersand = P("&")
3228 parsers.backtick = P("`")
3229 parsers.less = P("<")
3230 parsers.more = P(">")
3231 parsers.space = P(" ")
3232 parsers.squote = P("'")
3233 parsers.dquote = P('"')
3234 parsers.lparent = P("(")
3235 parsers.rparent = P(")")
3236 parsers.lbracket = P("[")
3237 parsers.rbracket = P("]")
3238 parsers.lbrace = P("{")
3239 parsers.rbrace = P("}")
3240 parsers.circumflex = P("^")
3241 parsers.slash = P("/")
3242 parsers.equal = P("=")
3243 parsers.colon = P(":")
3244 parsers.semicolon = P(";")
3245 parsers.exclamation = P("!")
3246 parsers.pipe = P("|")
3247 parsers.tilde = P("~")
3248 parsers.tab = P("\t")
3249 parsers.newline = P("\n")
3250 parsers.tightblocksep = P("\001")
3251
```

```

3252 parsers.digit           = R("09")
3253 parsers.hexdigit       = R("09", "af", "AF")
3254 parsers.letter         = R("AZ", "az")
3255 parsers.alphanumeric    = R("AZ", "az", "09")
3256 parsers.keyword        = parsers.letter
3257                          * parsers.alphanumeric^0
3258 parsers.citation_chars  = parsers.alphanumeric
3259                          + S("#$%&-+<>~/_")
3260 parsers.internal_punctuation = S(":,;.?" )
3261
3262 parsers.doubleasterisks  = P("**")
3263 parsers.doubleunderscores = P("__")
3264 parsers.fourspace       = P("    ")
3265
3266 parsers.any              = P(1)
3267 parsers.fail             = parsers.any - 1
3268
3269 parsers.escapable       = S("\\`*_{}[]()+_ .!<>#-~:~^@;")
3270 parsers.anyescaped      = P("\\") / "" * parsers.escapable
3271                          + parsers.any
3272
3273 parsers.spacechar       = S("\t ")
3274 parsers.spacing        = S(" \n\r\t")
3275 parsers.nonspacechar    = parsers.any - parsers.spacing
3276 parsers.optionalspace  = parsers.spacechar^0
3277
3278 parsers.specialchar     = S("*_`&[]<!\\.\@-~")
3279
3280 parsers.normalchar     = parsers.any - (parsers.specialchar
3281                                         + parsers.spacing
3282                                         + parsers.tightblocksep)
3283 parsers.eof             = -parsers.any
3284 parsers.nonindentpace   = parsers.space^-3 * - parsers.spacechar
3285 parsers.indent         = parsers.space^-3 * parsers.tab
3286                          + parsers.fourspace / ""
3287 parsers.linechar        = P(1 - parsers.newline)
3288
3289 parsers.blankline      = parsers.optionalspace
3290                          * parsers.newline / "\n"
3291 parsers.blanklines     = parsers.blankline^0
3292 parsers.skipblanklines = (parsers.optionalspace * parsers.newline)^0
3293 parsers.indentedline   = parsers.indent / ""
3294                          * C(parsers.linechar^1 * parsers.newline^-
3295                              1)
3296 parsers.optionallyindentedline = parsers.indent^-1 / ""
3297                          * C(parsers.linechar^1 * parsers.newline^-
3298                              1)

```

```

3297 parsers.sp = parsers.spacing^0
3298 parsers.spnl = parsers.optionalspace
3299 * (parsers.newline * parsers.optionalspace)^-
3300 1
3301 parsers.line = parsers.linechar^0 * parsers.newline
3302 parsers.nonemptyline = parsers.line - parsers.blankline
3303 parsers.chunk = parsers.line * (parsers.optionallyindentedline
3304 - parsers.blankline)^0
3305
3306 parsers.css_identifier_char = R("AZ", "az", "09") + S("-_")
3307 parsers.css_identifier = (parsers.hash + parsers.period)
3308 * (((parsers.css_identifier_char
3309 - parsers.dash - parsers.digit)
3310 * parsers.css_identifier_char^1)
3311 + (parsers.dash
3312 * (parsers.css_identifier_char
3313 - parsers.digit)
3314 * parsers.css_identifier_char^0))
3315 parsers.attribute_name_char = parsers.any - parsers.space
3316 - parsers.squote - parsers.dquote
3317 - parsers.more - parsers.slash
3318 - parsers.equal
3319 parsers.attribute_value_char = parsers.any - parsers.dquote
3320 - parsers.more
3321
3322 -- block followed by 0 or more optionally
3323 -- indented blocks with first line indented.
3324 parsers.indented_blocks = function(bl)
3325   return Cs( bl
3326     * (parsers.blankline^1 * parsers.indent * -parsers.blankline * bl)^0
3327     * (parsers.blankline^1 + parsers.eof) )
3328 end

```

3.1.4.2 Parsers Used for Markdown Lists

```

3329 parsers.bulletchar = C(parsers.plus + parsers.asterisk + parsers.dash)
3330
3331 parsers.bullet = ( parsers.bulletchar * #parsers.spacing
3332                   * (parsers.tab + parsers.space^-3)
3333                   + parsers.space * parsers.bulletchar * #parsers.spacing
3334                   * (parsers.tab + parsers.space^-2)
3335                   + parsers.space * parsers.space * parsers.bulletchar
3336                   * #parsers.spacing
3337                   * (parsers.tab + parsers.space^-1)
3338                   + parsers.space * parsers.space * parsers.space
3339                   * parsers.bulletchar * #parsers.spacing

```



```
3340 )
```

3.1.4.3 Parsers Used for Markdown Code Spans

```
3341 parsers.openticks = Cg(parsers.backtick^1, "ticks")
3342
3343 local function captures_equal_length(s,i,a,b)
3344   return #a == #b and i
3345 end
3346
3347 parsers.closeticks = parsers.space^-1
3348                   * Cmt(C(parsers.backtick^1)
3349                       * Cb("ticks"), captures_equal_length)
3350
3351 parsers.intickschar = (parsers.any - S("\n\r`"))
3352                   + (parsers.newline * -parsers.blankline)
3353                   + (parsers.space - parsers.closeticks)
3354                   + (parsers.backtick^1 - parsers.closeticks)
3355
3356 parsers.inticks = parsers.openticks * parsers.space^-1
3357                 * C(parsers.intickschar^0) * parsers.closeticks
```

3.1.4.4 Parsers Used for Fenced Code Blocks

```
3358 local function captures_geq_length(s,i,a,b)
3359   return #a >= #b and i
3360 end
3361
3362 parsers.infostring = (parsers.linechar - (parsers.backtick
3363     + parsers.space^1 * (parsers.newline + parsers.eof)))^0
3364
3365 local fenceindent
3366 parsers.fencehead = function(char)
3367   return C(parsers.nonindentospace) / function(s) fenceindent = #s end
3368   * Cg(char^3, "fencelength")
3369   * parsers.optionalspace * C(parsers.infostring)
3370   * parsers.optionalspace * (parsers.newline + parsers.eof)
3371 end
3372
3373 parsers.fencetail = function(char)
3374   return parsers.nonindentospace
3375   * Cmt(C(char^3) * Cb("fencelength"), captures_geq_length)
3376   * parsers.optionalspace * (parsers.newline + parsers.eof)
3377   + parsers.eof
3378 end
3379
3380 parsers.fencedline = function(char)
3381   return C(parsers.line - parsers.fencetail(char))
```

```

3382         / function(s)
3383         i = 1
3384         remaining = fenceindent
3385         while true do
3386             c = s:sub(i, i)
3387             if c == " " and remaining > 0 then
3388                 remaining = remaining - 1
3389                 i = i + 1
3390             elseif c == "\t" and remaining > 3 then
3391                 remaining = remaining - 4
3392                 i = i + 1
3393             else
3394                 break
3395             end
3396         end
3397         return s:sub(i)
3398     end
3399 end

```

3.1.4.5 Parsers Used for Markdown Tags and Links

```

3400 parsers.leader      = parsers.space^-3
3401
3402 -- content in balanced brackets, parentheses, or quotes:
3403 parsers.bracketed  = P{ parsers.lbracket
3404     * ((parsers.anyescaped - (parsers.lbracket
3405         + parsers.rbracket
3406         + parsers.blankline^2)
3407     ) + V(1))^0
3408     * parsers.rbracket }
3409
3410 parsers.inparens   = P{ parsers.lparent
3411     * ((parsers.anyescaped - (parsers.lparent
3412         + parsers.rparent
3413         + parsers.blankline^2)
3414     ) + V(1))^0
3415     * parsers.rparent }
3416
3417 parsers.squoted    = P{ parsers.squote * parsers.alphanumeric
3418     * ((parsers.anyescaped - (parsers.squote
3419         + parsers.blankline^2)
3420     ) + V(1))^0
3421     * parsers.squote }
3422
3423 parsers.dquoted    = P{ parsers.dquote * parsers.alphanumeric
3424     * ((parsers.anyescaped - (parsers.dquote
3425         + parsers.blankline^2)

```

```

3426         ) + V(1))^0
3427     * parsers.dquote }
3428
3429 -- bracketed tag for markdown links, allowing nested brackets:
3430 parsers.tag      = parsers.lbracket
3431                 * Cs((parsers.alphanumeric^1
3432                     + parsers.bracketed
3433                     + parsers.inticks
3434                     + (parsers.anyescaped
3435                       - (parsers.rbracket + parsers.blankline^2)))^0)
3436                 * parsers.rbracket
3437
3438 -- url for markdown links, allowing nested brackets:
3439 parsers.url      = parsers.less * Cs((parsers.anyescaped
3440                                     - parsers.more)^0)
3441                 * parsers.more
3442                 + Cs((parsers.inparens + (parsers.anyescaped
3443                     - parsers.spacing
3444                     - parsers.rparent))^1)
3445
3446 -- quoted text, possibly with nested quotes:
3447 parsers.title_s  = parsers.squote * Cs(((parsers.anyescaped-parsers.squote)
3448                                     + parsers.squoted)^0)
3449                 * parsers.squote
3450
3451 parsers.title_d  = parsers.dquote * Cs(((parsers.anyescaped-parsers.dquote)
3452                                     + parsers.dquoted)^0)
3453                 * parsers.dquote
3454
3455 parsers.title_p  = parsers.lparent
3456                 * Cs((parsers.inparens + (parsers.anyescaped-parsers.rparent))^0)
3457                 * parsers.rparent
3458
3459 parsers.title    = parsers.title_d + parsers.title_s + parsers.title_p
3460
3461 parsers.optionaltitle
3462                 = parsers.spnl * parsers.title * parsers.spacechar^0
3463                 + Cc("")

```

3.1.4.6 Parsers Used for iA Writer Content Blocks

```

3464 parsers.contentblock_tail
3465                 = parsers.optionaltitle
3466                 * (parsers.newline + parsers.eof)
3467
3468 -- case insensitive online image suffix:
3469 parsers.onlineimagesuffix

```

```

3470         = (function(...)
3471             local parser = nil
3472             for _,suffix in ipairs({...}) do
3473                 local pattern=nil
3474                 for i=1,#suffix do
3475                     local char=suffix:sub(i,i)
3476                     char = S(char:lower()..char:upper())
3477                     if pattern == nil then
3478                         pattern = char
3479                     else
3480                         pattern = pattern * char
3481                     end
3482                 end
3483                 if parser == nil then
3484                     parser = pattern
3485                 else
3486                     parser = parser + pattern
3487                 end
3488             end
3489             return parser
3490         end)("png", "jpg", "jpeg", "gif", "tif", "tiff")
3491
3492 -- online image url for iA Writer content blocks with mandatory suffix,
3493 -- allowing nested brackets:
3494 parsers.onlineimageurl
3495         = (parsers.less
3496             * Cs((parsers.anyescaped
3497                 - parsers.more
3498                 - #(parsers.period
3499                     * parsers.onlineimagesuffix
3500                     * parsers.more
3501                     * parsers.contentblock_tail))^0)
3502             * parsers.period
3503             * Cs(parsers.onlineimagesuffix)
3504             * parsers.more
3505             + (Cs((parsers.inparens
3506                 + (parsers.anyescaped
3507                     - parsers.spacing
3508                     - parsers.rparent
3509                     - #(parsers.period
3510                         * parsers.onlineimagesuffix
3511                         * parsers.contentblock_tail))))^0)
3512             * parsers.period
3513             * Cs(parsers.onlineimagesuffix))
3514             ) * Cc("onlineimage")
3515
3516 -- filename for iA Writer content blocks with mandatory suffix:

```

```

3517 parsers.localfilepath
3518         = parsers.slash
3519         * Cs((parsers.anyescaped
3520             - parsers.tab
3521             - parsers.newline
3522             - #(parsers.period
3523                 * parsers.alphanumeric^1
3524                 * parsers.contentblock_tail))^1)
3525         * parsers.period
3526         * Cs(parsers.alphanumeric^1)
3527         * Cc("localfile")

```

3.1.4.7 Parsers Used for Citations

```

3528 parsers.citation_name = Cs(parsers.dash^-1) * parsers.at
3529         * Cs(parsers.citation_chars
3530             * (((parsers.citation_chars + parsers.internal_punctuation
3531                 - parsers.comma - parsers.semicolon)
3532                 * -#((parsers.internal_punctuation - parsers.comma
3533                     - parsers.semicolon)^0
3534                     * -(parsers.citation_chars + parsers.internal_punctuat.
3535                         - parsers.comma - parsers.semicolon)))^0
3536                 * parsers.citation_chars)^-1)
3537
3538 parsers.citation_body_prenote
3539         = Cs((parsers.alphanumeric^1
3540             + parsers.bracketed
3541             + parsers.inticks
3542             + (parsers.anyescaped
3543                 - (parsers.rbracket + parsers.blankline^2))
3544             - (parsers.spnl * parsers.dash^-1 * parsers.at))^0)
3545
3546 parsers.citation_body_postnote
3547         = Cs((parsers.alphanumeric^1
3548             + parsers.bracketed
3549             + parsers.inticks
3550             + (parsers.anyescaped
3551                 - (parsers.rbracket + parsers.semicolon
3552                     + parsers.blankline^2))
3553             - (parsers.spnl * parsers.rbracket))^0)
3554
3555 parsers.citation_body_chunk
3556         = parsers.citation_body_prenote
3557         * parsers.spnl * parsers.citation_name
3558         * (parsers.internal_punctuation - parsers.semicolon)^-
1
3559         * parsers.spnl * parsers.citation_body_postnote

```

```

3560
3561 parsers.citation_body
3562     = parsers.citation_body_chunk
3563     * (parsers.semicolon * parsers.spnl
3564     * parsers.citation_body_chunk)^0
3565
3566 parsers.citation_headless_body_postnote
3567     = Cs((parsers.alphanumeric^1
3568     + parsers.bracketed
3569     + parsers.inticks
3570     + (parsers.anyescaped
3571     - (parsers.rbracket + parsers.at
3572     + parsers.semicolon + parsers.blankline^2))
3573     - (parsers.spnl * parsers.rbracket))^0
3574
3575 parsers.citation_headless_body
3576     = parsers.citation_headless_body_postnote
3577     * (parsers.sp * parsers.semicolon * parsers.spnl
3578     * parsers.citation_body_chunk)^0

```

3.1.4.8 Parsers Used for Footnotes

```

3579 local function strip_first_char(s)
3580     return s:sub(2)
3581 end
3582
3583 parsers.RawNoteRef = #(parsers.lbracket * parsers.circumflex)
3584     * parsers.tag / strip_first_char

```

3.1.4.9 Parsers Used for Tables

```

3585 local function make_pipe_table_rectangular(rows)
3586     local num_columns = #rows[2]
3587     local rectangular_rows = {}
3588     for i = 1, #rows do
3589         local row = rows[i]
3590         local rectangular_row = {}
3591         for j = 1, num_columns do
3592             rectangular_row[j] = row[j] or ""
3593         end
3594         table.insert(rectangular_rows, rectangular_row)
3595     end
3596     return rectangular_rows
3597 end
3598
3599 local function pipe_table_row(allow_empty_first_column
3600     , nonempty_column
3601     , column_separator

```

```

3602         , column)
3603 local row_beginning
3604 if allow_empty_first_column then
3605     row_beginning = -- empty first column
3606         #(parsers.spacechar^4
3607         * column_separator)
3608         * parsers.optionalspace
3609         * column
3610         * parsers.optionalspace
3611     -- non-empty first column
3612     + parsers.nonindentspace
3613     * nonempty_column^-1
3614     * parsers.optionalspace
3615 else
3616     row_beginning = parsers.nonindentspace
3617         * nonempty_column^-1
3618         * parsers.optionalspace
3619 end
3620
3621 return Ct(row_beginning
3622     * (-- single column with no leading pipes
3623     #(column_separator
3624     * parsers.optionalspace
3625     * parsers.newline)
3626     * column_separator
3627     * parsers.optionalspace
3628     -- single column with leading pipes or
3629     -- more than a single column
3630     + (column_separator
3631     * parsers.optionalspace
3632     * column
3633     * parsers.optionalspace)^1
3634     * (column_separator
3635     * parsers.optionalspace)^-1))
3636 end
3637
3638 parsers.table_hline_separator = parsers.pipe + parsers.plus
3639 parsers.table_hline_column = (parsers.dash
3640     - #(parsers.dash
3641     * (parsers.spacechar
3642     + parsers.table_hline_separator
3643     + parsers.newline)))^1
3644     * (parsers.colon * Cc("r")
3645     + parsers.dash * Cc("d"))
3646     + parsers.colon
3647     * (parsers.dash
3648     - #(parsers.dash

```

```

3649             * (parsers.spacechar
3650               + parsers.table_hline_separator
3651               + parsers.newline)))^1
3652         * (parsers.colon * Cc("c")
3653           + parsers.dash * Cc("l"))
3654 parsers.table_hline = pipe_table_row(false
3655                                     , parsers.table_hline_column
3656                                     , parsers.table_hline_separator
3657                                     , parsers.table_hline_column)
3658 parsers.table_caption_beginning = parsers.skipblanklines
3659                                 * parsers.nonindentSPACE
3660                                 * (P("Table")^-1 * parsers.colon)
3661                                 * parsers.optionalSPACE

```

3.1.4.10 Parsers Used for HTML

```

3662 -- case-insensitive match (we assume s is lowercase). must be single byte encoding
3663 parsers.keyword_exact = function(s)
3664   local parser = P(0)
3665   for i=1,#s do
3666     local c = s:sub(i,i)
3667     local m = c .. upper(c)
3668     parser = parser * S(m)
3669   end
3670   return parser
3671 end
3672
3673 parsers.block_keyword =
3674   parsers.keyword_exact("address") + parsers.keyword_exact("blockquote") +
3675   parsers.keyword_exact("center") + parsers.keyword_exact("del") +
3676   parsers.keyword_exact("dir") + parsers.keyword_exact("div") +
3677   parsers.keyword_exact("p") + parsers.keyword_exact("pre") +
3678   parsers.keyword_exact("li") + parsers.keyword_exact("ol") +
3679   parsers.keyword_exact("ul") + parsers.keyword_exact("dl") +
3680   parsers.keyword_exact("dd") + parsers.keyword_exact("form") +
3681   parsers.keyword_exact("fieldset") + parsers.keyword_exact("isindex") +
3682   parsers.keyword_exact("ins") + parsers.keyword_exact("menu") +
3683   parsers.keyword_exact("noframes") + parsers.keyword_exact("frameset") +
3684   parsers.keyword_exact("h1") + parsers.keyword_exact("h2") +
3685   parsers.keyword_exact("h3") + parsers.keyword_exact("h4") +
3686   parsers.keyword_exact("h5") + parsers.keyword_exact("h6") +
3687   parsers.keyword_exact("hr") + parsers.keyword_exact("script") +
3688   parsers.keyword_exact("noscript") + parsers.keyword_exact("table") +
3689   parsers.keyword_exact("tbody") + parsers.keyword_exact("tfoot") +
3690   parsers.keyword_exact("thead") + parsers.keyword_exact("th") +
3691   parsers.keyword_exact("td") + parsers.keyword_exact("tr")
3692

```



```

3693 -- There is no reason to support bad html, so we expect quoted attributes
3694 parsers.htmlattributevalue
3695         = parsers.squote * (parsers.any - (parsers.blankline
3696                                           + parsers.squote))^0
3697           * parsers.squote
3698         + parsers.dquote * (parsers.any - (parsers.blankline
3699                                           + parsers.dquote))^0
3700           * parsers.dquote
3701
3702 parsers.htmlattribute = parsers.spacing^1
3703                       * (parsers.alphanumeric + S("_-"))^1
3704                       * parsers.sp * parsers.equal * parsers.sp
3705                       * parsers.htmlattributevalue
3706
3707 parsers.htmlcomment   = P("<!--") * (parsers.any - P("-->"))^0 * P("-->")
3708
3709 parsers.htmlinstruction = P("<?") * (parsers.any - P("?>"))^0 * P("?>")
3710
3711 parsers.openelt_any = parsers.less * parsers.keyword * parsers.htmlattribute^0
3712                   * parsers.sp * parsers.more
3713
3714 parsers.openelt_exact = function(s)
3715   return parsers.less * parsers.sp * parsers.keyword_exact(s)
3716         * parsers.htmlattribute^0 * parsers.sp * parsers.more
3717 end
3718
3719 parsers.openelt_block = parsers.sp * parsers.block_keyword
3720                   * parsers.htmlattribute^0 * parsers.sp * parsers.more
3721
3722 parsers.closeelt_any = parsers.less * parsers.sp * parsers.slash
3723                   * parsers.keyword * parsers.sp * parsers.more
3724
3725 parsers.closeelt_exact = function(s)
3726   return parsers.less * parsers.sp * parsers.slash * parsers.keyword_exact(s)
3727         * parsers.sp * parsers.more
3728 end
3729
3730 parsers.emptyelt_any = parsers.less * parsers.sp * parsers.keyword
3731                   * parsers.htmlattribute^0 * parsers.sp * parsers.slash
3732                   * parsers.more
3733
3734 parsers.emptyelt_block = parsers.less * parsers.sp * parsers.block_keyword
3735                   * parsers.htmlattribute^0 * parsers.sp * parsers.slash
3736                   * parsers.more
3737
3738 parsers.displaytext = (parsers.any - parsers.less)^1
3739

```

```

3740 -- return content between two matched HTML tags
3741 parsers.in_matched = function(s)
3742   return { parsers.openelt_exact(s)
3743           * (V(1) + parsers.displaytext
3744             + (parsers.less - parsers.closeelt_exact(s)))^0
3745           * parsers.closeelt_exact(s) }
3746 end
3747
3748 local function parse_matched_tags(s,pos)
3749   local t = string.lower(lpeg.match(C(parsers.keyword),s,pos))
3750   return lpeg.match(parsers.in_matched(t),s,pos-1)
3751 end
3752
3753 parsers.in_matched_block_tags = parsers.less
3754                               * Cmt(#parsers.openelt_block, parse_matched_tags)
3755
3756 parsers.displayhtml = parsers.htmlcomment
3757                    + parsers.emptyelt_block
3758                    + parsers.openelt_exact("hr")
3759                    + parsers.in_matched_block_tags
3760                    + parsers.htmlinstruction
3761
3762 parsers.inlinehtml  = parsers.emptyelt_any
3763                    + parsers.htmlcomment
3764                    + parsers.htmlinstruction
3765                    + parsers.openelt_any
3766                    + parsers.closeelt_any

```

3.1.4.11 Parsers Used for HTML Entities

```

3767 parsers.hexentity = parsers.ampersand * parsers.hash * S("Xx")
3768                    * C(parsers.hexdigit^1) * parsers.semicolon
3769 parsers.decentity = parsers.ampersand * parsers.hash
3770                    * C(parsers.digit^1) * parsers.semicolon
3771 parsers.tagentity = parsers.ampersand * C(parsers.alphanumeric^1)
3772                    * parsers.semicolon

```

3.1.4.12 Helpers for References

```

3773 -- parse a reference definition: [foo]: /bar "title"
3774 parsers.define_reference_parser = parsers.leader * parsers.tag * parsers.colon
3775                               * parsers.spacechar^0 * parsers.url
3776                               * parsers.optionaltitle * parsers.blankline^1

```

3.1.4.13 Inline Elements

```

3777 parsers.Inline      = V("Inline")
3778 parsers.IndentedInline = V("IndentedInline")

```

```

3779
3780 -- parse many p between starter and ender
3781 parsers.between = function(p, starter, ender)
3782   local ender2 = B(parsers.nonspacechar) * ender
3783   return (starter * #parsers.nonspacechar * Ct(p * (p - ender2)^0) * ender2)
3784 end
3785
3786 parsers.urlchar      = parsers.anyescaped - parsers.newline - parsers.more

```

3.1.4.14 Block Elements

```

3787 parsers.Block      = V("Block")
3788
3789 parsers.OnlineImageURL
3790     = parsers.leader
3791     * parsers.onlineimageurl
3792     * parsers.optionaltitle
3793
3794 parsers.LocalFilePath
3795     = parsers.leader
3796     * parsers.localfilepath
3797     * parsers.optionaltitle
3798
3799 parsers.TildeFencedCode
3800     = parsers.fencehead(parsers.tilde)
3801     * Cs(parsers.fencedline(parsers.tilde)^0)
3802     * parsers.fencetail(parsers.tilde)
3803
3804 parsers.BacktickFencedCode
3805     = parsers.fencehead(parsers.backtick)
3806     * Cs(parsers.fencedline(parsers.backtick)^0)
3807     * parsers.fencetail(parsers.backtick)
3808
3809 parsers.lineof = function(c)
3810   return (parsers.leader * (P(c) * parsers.optionalspace)^3
3811         * (parsers.newline * parsers.blankline^1
3812         + parsers.newline^-1 * parsers.eof))
3813 end

```

3.1.4.15 Lists

```

3814 parsers.defstartchar = S("~:")
3815 parsers.defstart     = ( parsers.defstartchar * #parsers.spacing
3816                       * (parsers.tab + parsers.space^-
3817
3818
3817                       + parsers.space * parsers.defstartchar * #parsers.spacing
3818                       * (parsers.tab + parsers.space^-2)
3819
3819                       + parsers.space * parsers.space * parsers.defstartchar

```

```

3820             * #parsers.spacing
3821             * (parsers.tab + parsers.space^-1)
3822         + parsers.space * parsers.space * parsers.space
3823             * parsers.defstartchar * #parsers.spacing
3824     )
3825
3826 parsers.dlchunk = Cs(parsers.line * (parsers.indentedline - parsers.blankline)^0)

```

3.1.4.16 Headings

```

3827 parsers.heading_attribute = C(parsers.css_identifier)
3828     + C((parsers.attribute_name_char
3829         - parsers.rbrace)^1
3830         * parsers.equal
3831         * (parsers.attribute_value_char
3832         - parsers.rbrace)^1)
3833 parsers.HeadingAttributes = parsers.lbrace
3834     * parsers.heading_attribute
3835     * (parsers.spacechar^1
3836     * parsers.heading_attribute)^0
3837     * parsers.rbrace
3838
3839 -- parse Atx heading start and return level
3840 parsers.HeadingStart = #parsers.hash * C(parsers.hash^-6)
3841     * -parsers.hash / length
3842
3843 -- parse setext header ending and return level
3844 parsers.HeadingLevel = parsers.equal^1 * Cc(1) + parsers.dash^1 * Cc(2)
3845
3846 local function strip_atx_end(s)
3847     return s:gsub("#%s*\n$", "")
3848 end

```

3.1.5 Markdown Reader

This section documents the `reader` object, which implements the routines for parsing the markdown input. The object corresponds to the markdown reader object that was located in the `lunamark/reader/markdown.lua` file in the Lunamark Lua module.

Although not specified in the Lua interface (see Section 2.1), the `reader` object is exported, so that the curious user could easily tinker with the methods of the objects produced by the `reader.new` method described below. The user should be aware, however, that the implementation may change in a future revision.

The `reader.new` method creates and returns a new TeX reader object associated with the Lua interface options (see Section 2.1.2) `options` and with a writer object `writer`. When `options` are unspecified, it is assumed that an empty table was passed to the method.

The objects produced by the `reader.new` method expose instance methods and variables of their own. As a convention, I will refer to these *member*s as `reader->member`.

```
3849 M.reader = {}
3850 function M.reader.new(writer, options)
3851   local self = {}
3852   options = options or {}
```

Make the `options` table inherit from the `defaultOptions` table.

```
3853   setmetatable(options, { __index = function (_, key)
3854     return defaultOptions[key] end })
```

3.1.5.1 Top-Level Helper Functions Define `normalize_tag` as a function that normalizes a markdown reference tag by lowercasing it, and by collapsing any adjacent whitespace characters.

```
3855   local function normalize_tag(tag)
3856     return unicode.utf8.lower(
3857       gsub(util.rope_to_string(tag), "[\n\r\t]+", " "))
3858   end
```

Define `expandtabs` either as an identity function, when the `preserveTabs` Lua inrface option is `true`, or to a function that expands tabs into spaces otherwise.

```
3859   local expandtabs
3860   if options.preserveTabs then
3861     expandtabs = function(s) return s end
3862   else
3863     expandtabs = function(s)
3864       if s:find("\t") then
3865         return s:gsub("[^\n]*", util.expand_tabs_in_line)
3866       else
3867         return s
3868       end
3869     end
3870   end
```

The `larsers` (as in `local \luam{parsers}'`) hash table stores `\acro{peg}` patterns `tions`, which impedes their reuse between different `reader` objects.

```
3871   local larsers = {}
```

3.1.5.2 Top-Level Parser Functions

```
3872   local function create_parser(name, grammar)
3873     return function(str)
3874       local res = lpeg.match(grammar(), str)
3875       if res == nil then
3876         error(format("%s failed on:\n%s", name, str:sub(1,20)))
3877       else
```

```

3878     return res
3879   end
3880 end
3881 end
3882
3883 local parse_blocks
3884   = create_parser("parse_blocks",
3885     function()
3886       return larsers.blocks
3887     end)
3888
3889 local parse_blocks_toplevel
3890   = create_parser("parse_blocks_toplevel",
3891     function()
3892       return larsers.blocks_toplevel
3893     end)
3894
3895 local parse_inlines
3896   = create_parser("parse_inlines",
3897     function()
3898       return larsers.inlines
3899     end)
3900
3901 local parse_inlines_no_link
3902   = create_parser("parse_inlines_no_link",
3903     function()
3904       return larsers.inlines_no_link
3905     end)
3906
3907 local parse_inlines_no_inline_note
3908   = create_parser("parse_inlines_no_inline_note",
3909     function()
3910       return larsers.inlines_no_inline_note
3911     end)
3912
3913 local parse_inlines_nbsp
3914   = create_parser("parse_inlines_nbsp",
3915     function()
3916       return larsers.inlines_nbsp
3917     end)

```

3.1.5.3 Parsers Used for Markdown Lists (local)

```

3918 if options.hashEnumerators then
3919   larsers.dig = parsers.digit + parsers.hash
3920 else
3921   larsers.dig = parsers.digit

```

```

3922 end
3923
3924 larsers.enumerator = C(larsers.dig^3 * parsers.period) * #parsers.spacing
3925 + C(larsers.dig^2 * parsers.period) * #parsers.spacing
3926 + (parsers.tab + parsers.space^1)
3927 + C(larsers.dig * parsers.period) * #parsers.spacing
3928 + (parsers.tab + parsers.space^-2)
3929 + parsers.space * C(larsers.dig^2 * parsers.period)
3930 + #parsers.spacing
3931 + parsers.space * C(larsers.dig * parsers.period)
3932 + #parsers.spacing
3933 + (parsers.tab + parsers.space^-1)
3934 + parsers.space * parsers.space * C(larsers.dig^1
3935 + parsers.period) * #parsers.spacing

```

3.1.5.4 Parsers Used for Blockquotes (local)

```

3936 -- strip off leading > and indents, and run through blocks
3937 larsers.blockquote_body = ((parsers.lead * parsers.more * parsers.space^-
3938 1)/""
3939 + parsers.linechar^0 * parsers.newline)^1
3940 + (-(parsers.lead * parsers.more
3941 + parsers.blankline) * parsers.linechar^1
3942 + parsers.newline)^0
3943 if not options.breakableBlockquotes then
3944   larsers.blockquote_body = larsers.blockquote_body
3945   + (parsers.blankline^0 / "")
3946 end

```

3.1.5.5 Parsers Used for Citations (local)

```

3947 larsers.citations = function(text_cites, raw_cites)
3948   local function normalize(str)
3949     if str == "" then
3950       str = nil
3951     else
3952       str = (options.citationNbsps and parse_inlines_nbsp or
3953         parse_inlines)(str)
3954     end
3955     return str
3956   end
3957
3958   local cites = {}
3959   for i = 1,#raw_cites,4 do
3960     cites[#cites+1] = {
3961       prenote = normalize(raw_cites[i]),
3962       suppress_author = raw_cites[i+1] == "-",

```

```

3963         name = writer.citation(raw_cites[i+2]),
3964         postnote = normalize(raw_cites[i+3]),
3965     }
3966 end
3967 return writer.citations(text_cites, cites)
3968 end

```

3.1.5.6 Parsers Used for Footnotes (local)

```

3969 local rawnotes = {}
3970
3971 -- like indirect_link
3972 local function lookup_note(ref)
3973     return function()
3974         local found = rawnotes[normalize_tag(ref)]
3975         if found then
3976             return writer.note(parse_blocks_toplevel(found))
3977         else
3978             return {"[", parse_inlines("^" .. ref), "]" }
3979         end
3980     end
3981 end
3982
3983 local function register_note(ref,rawnote)
3984     rawnotes[normalize_tag(ref)] = rawnote
3985     return ""
3986 end
3987
3988 larsers.NoteRef      = parsers.RawNoteRef / lookup_note
3989
3990
3991 larsers.NoteBlock   = parsers.leader * parsers.RawNoteRef * parsers.colon
3992                    * parsers.spnl * parsers.indented_blocks(parsers.chunk)
3993                    / register_note
3994
3995 larsers.InlineNote  = parsers.circumflex
3996                    * (parsers.tag / parse_inlines_no_inline_note) -- no notes inside
3997                    / writer.note

```

3.1.5.7 Parsers Used for Tables (local)

```

3998 larsers.table_row = pipe_table_row(true
3999                                , (C((parsers.linechar - parsers.pipe)^1)
4000                                / parse_inlines)
4001                                , parsers.pipe
4002                                , (C((parsers.linechar - parsers.pipe)^0)
4003                                / parse_inlines))
4004

```



```

4005 if options.tableCaptions then
4006   larsers.table_caption = #parsers.table_caption_beginning
4007                         * parsers.table_caption_beginning
4008                         * Ct(parsers.IndentedInline^1)
4009                         * parsers.newline
4010 else
4011   larsers.table_caption = parsers.fail
4012 end
4013
4014 larsers.PipeTable = Ct(larsers.table_row * parsers.newline
4015                       * parsers.table_hline
4016                       * (parsers.newline * larsers.table_row)^0)
4017                       / make_pipe_table_rectangular
4018                       * larsers.table_caption^-1
4019                       / writer.table

```

3.1.5.8 Helpers for Links and References (local)

```

4020 -- List of references defined in the document
4021 local references
4022
4023 -- add a reference to the list
4024 local function register_link(tag,url,title)
4025   references[normalize_tag(tag)] = { url = url, title = title }
4026   return ""
4027 end
4028
4029 -- lookup link reference and return either
4030 -- the link or nil and fallback text.
4031 local function lookup_reference(label,sps,tag)
4032   local tagpart
4033   if not tag then
4034     tag = label
4035     tagpart = ""
4036   elseif tag == "" then
4037     tag = label
4038     tagpart = "[]"
4039   else
4040     tagpart = {"[", parse_inlines(tag), "]" }
4041   end
4042   if sps then
4043     tagpart = {sps, tagpart}
4044   end
4045   local r = references[normalize_tag(tag)]
4046   if r then
4047     return r
4048   else

```

```

4049         return nil, {"[", parse_inlines(label), "]", tagpart}
4050     end
4051 end
4052
4053 -- lookup link reference and return a link, if the reference is found,
4054 -- or a bracketed label otherwise.
4055 local function indirect_link(label,sps,tag)
4056     return function()
4057         local r, fallback = lookup_reference(label,sps,tag)
4058         if r then
4059             return writer.link(parse_inlines_no_link(label), r.url, r.title)
4060         else
4061             return fallback
4062         end
4063     end
4064 end
4065
4066 -- lookup image reference and return an image, if the reference is found,
4067 -- or a bracketed label otherwise.
4068 local function indirect_image(label,sps,tag)
4069     return function()
4070         local r, fallback = lookup_reference(label,sps,tag)
4071         if r then
4072             return writer.image(writer.string(label), r.url, r.title)
4073         else
4074             return {"!", fallback}
4075         end
4076     end
4077 end

```

3.1.5.9 Inline Elements (local)

```

4078 larsers.Str      = (parsers.normalchar * (parsers.normalchar + parsers.at)^0)
4079                  / writer.string
4080
4081 larsers.Symbol   = (parsers.specialchar - parsers.tightblocksep)
4082                  / writer.string
4083
4084 larsers.Ellipsis = P("...") / writer.ellipsis
4085
4086 larsers.Smart    = larsers.Ellipsis
4087
4088 larsers.Code     = parsers.inticks / writer.code
4089
4090 if options.blankBeforeBlockquote then
4091     larsers.bqstart = parsers.fail
4092 else

```

```

4093     larsers.bqstart = parsers.more
4094 end
4095
4096 if options.blankBeforeHeading then
4097     larsers.headerstart = parsers.fail
4098 else
4099     larsers.headerstart = parsers.hash
4100                        + (parsers.line * (parsers.equal^1 + parsers.dash^1)
4101                        * parsers.optionalspace * parsers.newline)
4102 end
4103
4104 if not options.fencedCode or options.blankBeforeCodeFence then
4105     larsers.fencestart = parsers.fail
4106 else
4107     larsers.fencestart = parsers.fencehead(parsers.backtick)
4108                        + parsers.fencehead(parsers.tilde)
4109 end
4110
4111 larsers.Endline   = parsers.newline * -( -- newline, but not before...
4112                    parsers.blankline -- paragraph break
4113                    + parsers.tightblocksep -- nested list
4114                    + parsers.eof         -- end of document
4115                    + larsers.bqstart
4116                    + larsers.headerstart
4117                    + larsers.fencestart
4118                    ) * parsers.spacechar^0 / writer.space
4119
4120 larsers.OptionalIndent
4121     = parsers.spacechar^1 / writer.space
4122
4123 larsers.Space     = parsers.spacechar^2 * larsers.Endline / writer.linebreak
4124                    + parsers.spacechar^1 * larsers.Endline^-1 * parsers.eof / ""
4125                    + parsers.spacechar^1 * larsers.Endline^-1
4126                    * parsers.optionalspace / writer.space
4127
4128 larsers.NonbreakingEndline
4129     = parsers.newline * -( -- newline, but not before...
4130                    parsers.blankline -- paragraph break
4131                    + parsers.tightblocksep -- nested list
4132                    + parsers.eof         -- end of document
4133                    + larsers.bqstart
4134                    + larsers.headerstart
4135                    + larsers.fencestart
4136                    ) * parsers.spacechar^0 / writer.nbsp
4137
4138 larsers.NonbreakingSpace
4139     = parsers.spacechar^2 * larsers.Endline / writer.linebreak

```

```

4140         + parsers.spacechar^1 * larsers.Endline^-1 * parsers.eof / ""
4141         + parsers.spacechar^1 * larsers.Endline^-1
4142             * parsers.optionalspace / writer.nbsp
4143
4144     if options.underscores then
4145         larsers.Strong = ( parsers.between(parsers.Inline, parsers.doubleasterisks,
4146             parsers.doubleasterisks)
4147             + parsers.between(parsers.Inline, parsers.doubleunderscores,
4148             parsers.doubleunderscores)
4149             ) / writer.strong
4150
4151         larsers.Emph  = ( parsers.between(parsers.Inline, parsers.asterisk,
4152             parsers.asterisk)
4153             + parsers.between(parsers.Inline, parsers.underscore,
4154             parsers.underscore)
4155             ) / writer.emphasis
4156     else
4157         larsers.Strong = ( parsers.between(parsers.Inline, parsers.doubleasterisks,
4158             parsers.doubleasterisks)
4159             ) / writer.strong
4160
4161         larsers.Emph  = ( parsers.between(parsers.Inline, parsers.asterisk,
4162             parsers.asterisk)
4163             ) / writer.emphasis
4164     end
4165
4166     larsers.AutoLinkUrl  = parsers.less
4167         * C(parsers.alphanumeric^1 * P("://") * parsers.urlchar^1)
4168         * parsers.more
4169         / function(url)
4170             return writer.link(writer.string(url), url)
4171         end
4172
4173     larsers.AutoLinkEmail = parsers.less
4174         * C((parsers.alphanumeric + S("-._+"))^1
4175         * P("@") * parsers.urlchar^1)
4176         * parsers.more
4177         / function(email)
4178             return writer.link(writer.string(email),
4179                 "mailto:.."email)
4180         end
4181
4182     larsers.DirectLink  = (parsers.tag / parse_inlines_no_link) -- no links inside link
4183         * parsers.spnl
4184         * parsers.lparent
4185         * (parsers.url + Cc("")) -- link can be empty [foo]()
4186         * parsers.optionaltitle

```

```

4187             * parsers.rparent
4188             / writer.link
4189
4190 larsers.IndirectLink = parsers.tag * (C(parsers.spnl) * parsers.tag)^-
1
4191             / indirect_link
4192
4193 -- parse a link or image (direct or indirect)
4194 larsers.Link         = larsers.DirectLink + larsers.IndirectLink
4195
4196 larsers.DirectImage = parsers.exclamation
4197                     * (parsers.tag / parse_inlines)
4198                     * parsers.spnl
4199                     * parsers.lparent
4200                     * (parsers.url + Cc("")) -- link can be empty [foo]()
4201                     * parsers.optionaltitle
4202                     * parsers.rparent
4203                     / writer.image
4204
4205 larsers.IndirectImage = parsers.exclamation * parsers.tag
4206                     * (C(parsers.spnl) * parsers.tag)^-1 / indirect_image
4207
4208 larsers.Image        = larsers.DirectImage + larsers.IndirectImage
4209
4210 larsers.TextCitations = Ct(Cc(""))
4211                     * parsers.citation_name
4212                     * ((parsers.spnl
4213                       * parsers.lbracket
4214                       * parsers.citation_headless_body
4215                       * parsers.rbracket) + Cc(""))))
4216                     / function(raw_cites)
4217                       return larsers.citations(true, raw_cites)
4218                     end
4219
4220 larsers.ParenthesizedCitations
4221         = Ct(parsers.lbracket
4222           * parsers.citation_body
4223           * parsers.rbracket)
4224         / function(raw_cites)
4225           return larsers.citations(false, raw_cites)
4226         end
4227
4228 larsers.Citations    = larsers.TextCitations + larsers.ParenthesizedCitations
4229
4230 -- avoid parsing long strings of * or _ as emph/strong
4231 larsers.UlOrStarLine = parsers.asterisk^4 + parsers.underscore^4
4232                     / writer.string

```

```

4233
4234 larsers.EscapedChar = S("\\") * C(parsers.escapable) / writer.string
4235
4236 larsers.InlineHtml = C(parsers.inlinehtml) / writer.inline_html
4237
4238 larsers.HtmlEntity = parsers.hexentity / entities.hex_entity / writer.string
4239 + parsers.decentity / entities.dec_entity / writer.string
4240 + parsers.tagentity / entities.char_entity / writer.string

```

3.1.5.10 Block Elements (local)

```

4241 larsers.ContentBlock = parsers.leader
4242 * (parsers.localfilepath + parsers.onlineimageurl)
4243 * parsers.contentblock_tail
4244 / writer.contentblock
4245
4246 larsers.DisplayHtml = C(parsers.displayhtml)
4247 / expandtabs / writer.display_html
4248
4249 larsers.Verbatim = Cs( (parsers.blanklines
4250 * ((parsers.indentedline - parsers.blankline))^1)^1
4251 ) / expandtabs / writer.verbatim
4252
4253 larsers.FencedCode = (parsers.TildeFencedCode
4254 + parsers.BacktickFencedCode)
4255 / function(infostring, code)
4256 return writer.fencedCode(writer.string(infostring),
4257 expandtabs(code))
4258 end
4259
4260 larsers.Blockquote = Cs(larsers.blockquote_body^1)
4261 / parse_blocks_toplevel / writer.blockquote
4262
4263 larsers.HorizontalRule = ( parsers.lineof(parsers.asterisk)
4264 + parsers.lineof(parsers.dash)
4265 + parsers.lineof(parsers.underscore)
4266 ) / writer.hrule
4267
4268 larsers.Reference = parsers.define_reference_parser / register_link
4269
4270 larsers.Paragraph = parsers.nonindentspace * Ct(parsers.Inline^1)
4271 * parsers.newline
4272 * ( parsers.blankline^1
4273 + #parsers.hash
4274 + #(parsers.leader * parsers.more * parsers.space^-
1)
4275 )

```

```

4276             / writer.paragraph
4277
4278 larsers.ToplevelParagraph
4279         = parsers.nonindentspace * Ct(parsers.Inline^1)
4280         * ( parsers.newline
4281         * ( parsers.blankline^1
4282         + #parsers.hash
4283         + #(parsers.leader * parsers.more * parsers.space^-
1)
4284         + parsers.eof
4285         )
4286         + parsers.eof )
4287         / writer.paragraph
4288
4289 larsers.Plain         = parsers.nonindentspace * Ct(parsers.Inline^1)
4290         / writer.plain

```

3.1.5.11 Lists (local)

```

4291 larsers.starter = parsers.bullet + larsers.enumerator
4292
4293 -- we use \001 as a separator between a tight list item and a
4294 -- nested list under it.
4295 larsers.NestedList         = Cs((parsers.optionallyindentedline
4296         - larsers.starter)^1)
4297         / function(a) return "\001"..a end
4298
4299 larsers.ListBlockLine     = parsers.optionallyindentedline
4300         - parsers.blankline - (parsers.indent^-1
4301         * larsers.starter)
4302
4303 larsers.ListBlock        = parsers.line * larsers.ListBlockLine^0
4304
4305 larsers.ListContinuationBlock = parsers.blanklines * (parsers.indent / "")
4306         * larsers.ListBlock
4307
4308 larsers.TightListItem = function(starter)
4309     return -larsers.HorizontalRule
4310         * (Cs(starter / "" * larsers.ListBlock * larsers.NestedList^-
1)
4311         / parse_blocks)
4312         * -(parsers.blanklines * parsers.indent)
4313 end
4314
4315 larsers.LooseListItem = function(starter)
4316     return -larsers.HorizontalRule
4317         * Cs( starter / "" * larsers.ListBlock * Cc("\n")

```

```

4318         * (larsers.NestedList + larsers.ListContinuationBlock^0)
4319         * (parsers.blanklines / "\n\n")
4320         ) / parse_blocks
4321     end
4322
4323     larsers.BulletList = ( Ct(larsers.TightListItem(parsers.bullet)^1) * Cc(true)
4324         * parsers.skipblanklines * -parsers.bullet
4325         + Ct(larsers.LooseListItem(parsers.bullet)^1) * Cc(false)
4326         * parsers.skipblanklines )
4327         / writer.bulletlist
4328
4329     local function ordered_list(items,tight,startNumber)
4330         if options.startNumber then
4331             startNumber = tonumber(startNumber) or 1 -- fallback for '#'
4332             if startNumber ~= nil then
4333                 startNumber = math.floor(startNumber)
4334             end
4335         else
4336             startNumber = nil
4337         end
4338         return writer.orderedlist(items,tight,startNumber)
4339     end
4340
4341     larsers.OrderedList = Cg(larsers.enumerator, "listtype") *
4342         ( Ct(larsers.TightListItem(Cb("listtype")))
4343         * larsers.TightListItem(larsers.enumerator)^0)
4344         * Cc(true) * parsers.skipblanklines * -larsers.enumerator
4345         + Ct(larsers.LooseListItem(Cb("listtype")))
4346         * larsers.LooseListItem(larsers.enumerator)^0)
4347         * Cc(false) * parsers.skipblanklines
4348         ) * Cb("listtype") / ordered_list
4349
4350     local function definition_list_item(term, defs, tight)
4351         return { term = parse_inlines(term), definitions = defs }
4352     end
4353
4354     larsers.DefinitionListItemLoose = C(parsers.line) * parsers.skipblanklines
4355         * Ct((parsers.defstart
4356         * parsers.indented_blocks(parsers.dlchunk)
4357         / parse_blocks_toplevel)^1)
4358         * Cc(false) / definition_list_item
4359
4360     larsers.DefinitionListItemTight = C(parsers.line)
4361         * Ct((parsers.defstart * parsers.dlchunk
4362         / parse_blocks)^1)
4363         * Cc(true) / definition_list_item
4364

```



```

4365 larsers.DefinitionList = ( Ct(larsers.DefinitionListItemLoose^1) * Cc(false)
4366 + Ct(larsers.DefinitionListItemTight^1)
4367 * (parsers.skipblanklines
4368 * -larsers.DefinitionListItemLoose * Cc(true))
4369 ) / writer.definitionlist

```

3.1.5.12 Blank (local)

```

4370 larsers.Blank = parsers.blankline / ""
4371 + larsers.NoteBlock
4372 + larsers.Reference
4373 + (parsers.tightblocksep / "\n")

```

3.1.5.13 Headings (local)

```

4374 -- parse atx header
4375 if options.headerAttributes then
4376   larsers.AtHeading = Cg(parsers.HeadingStart,"level")
4377   * parsers.optionalspace
4378   * (C(((parsers.linechar
4379     - ((parsers.hash^1
4380       * parsers.optionalspace
4381       * parsers.HeadingAttributes^-1
4382       + parsers.HeadingAttributes)
4383       * parsers.optionalspace
4384       * parsers.newline)))
4385     * (parsers.linechar
4386       - parsers.hash
4387       - parsers.lbrace)^0)^1)
4388   / parse_inlines)
4389   * Cg(Ct(parsers.newline
4390     + (parsers.hash^1
4391       * parsers.optionalspace
4392       * parsers.HeadingAttributes^-1
4393       + parsers.HeadingAttributes)
4394       * parsers.optionalspace
4395       * parsers.newline), "attributes")
4396   * Cb("level")
4397   * Cb("attributes")
4398   / writer.heading
4399
4400 larsers.SetextHeading = #(parsers.line * S("-"))
4401 * (C(((parsers.linechar
4402   - (parsers.HeadingAttributes
4403     * parsers.optionalspace
4404     * parsers.newline)))
4405   * (parsers.linechar
4406     - parsers.lbrace)^0)^1)

```

```

4407         / parse_inlines)
4408     * Cg(Ct(parsers.newline
4409         + (parsers.HeadingAttributes
4410           * parsers.optionalspace
4411           * parsers.newline)), "attributes")
4412     * parsers.HeadingLevel
4413     * Cb("attributes")
4414     * parsers.optionalspace
4415     * parsers.newline
4416     / writer.heading
4417 else
4418     larsers.AtxHeading = Cg(parsers.HeadingStart,"level")
4419     * parsers.optionalspace
4420     * (C(parsers.line) / strip_atx_end / parse_inlines)
4421     * Cb("level")
4422     / writer.heading
4423
4424     larsers.SettextHeading = #(parsers.line * S("-"))
4425     * Ct(parsers.linechar^1 / parse_inlines)
4426     * parsers.newline
4427     * parsers.HeadingLevel
4428     * parsers.optionalspace
4429     * parsers.newline
4430     / writer.heading
4431 end
4432
4433 larsers.Heading = larsers.AtxHeading + larsers.SettextHeading

```

3.1.5.14 Syntax Specification

```

4434 local syntax =
4435     { "Blocks",
4436
4437       Blocks = larsers.Blank^0 * parsers.Block^-1
4438               * (larsers.Blank^0 / writer.interblocksep
4439                 * parsers.Block)^0
4440               * larsers.Blank^0 * parsers.eof,
4441
4442       Blank = larsers.Blank,
4443
4444       Block = V("ContentBlock")
4445              + V("Blockquote")
4446              + V("PipeTable")
4447              + V("Verbatim")
4448              + V("FencedCode")
4449              + V("HorizontalRule")
4450              + V("BulletList")

```

```

4451         + V("OrderedList")
4452         + V("Heading")
4453         + V("DefinitionList")
4454         + V("DisplayHtml")
4455         + V("Paragraph")
4456         + V("Plain"),
4457
4458     ContentBlock      = larsers.ContentBlock,
4459     Blockquote       = larsers.Blockquote,
4460     Verbatim         = larsers.Verbatim,
4461     FencedCode       = larsers.FencedCode,
4462     HorizontalRule   = larsers.HorizontalRule,
4463     BulletList       = larsers.BulletList,
4464     OrderedList      = larsers.OrderedList,
4465     Heading          = larsers.Heading,
4466     DefinitionList   = larsers.DefinitionList,
4467     DisplayHtml      = larsers.DisplayHtml,
4468     Paragraph        = larsers.Paragraph,
4469     PipeTable        = larsers.PipeTable,
4470     Plain            = larsers.Plain,
4471
4472     Inline            = V("Str")
4473         + V("Space")
4474         + V("Endline")
4475         + V("U1OrStarLine")
4476         + V("Strong")
4477         + V("Emph")
4478         + V("InlineNote")
4479         + V("NoteRef")
4480         + V("Citations")
4481         + V("Link")
4482         + V("Image")
4483         + V("Code")
4484         + V("AutoLinkUrl")
4485         + V("AutoLinkEmail")
4486         + V("InlineHtml")
4487         + V("HtmlEntity")
4488         + V("EscapedChar")
4489         + V("Smart")
4490         + V("Symbol"),
4491
4492     IndentedInline    = V("Str")
4493         + V("OptionalIndent")
4494         + V("Endline")
4495         + V("U1OrStarLine")
4496         + V("Strong")
4497         + V("Emph")

```

```

4498         + V("InlineNote")
4499         + V("NoteRef")
4500         + V("Citations")
4501         + V("Link")
4502         + V("Image")
4503         + V("Code")
4504         + V("AutoLinkUrl")
4505         + V("AutoLinkEmail")
4506         + V("InlineHtml")
4507         + V("HtmlEntity")
4508         + V("EscapedChar")
4509         + V("Smart")
4510         + V("Symbol"),
4511
4512     Str           = larsers.Str,
4513     Space        = larsers.Space,
4514     OptionalIndent = larsers.OptionalIndent,
4515     Endline      = larsers.Endline,
4516     U1OrStarLine = larsers.U1OrStarLine,
4517     Strong       = larsers.Strong,
4518     Emph        = larsers.Emph,
4519     InlineNote   = larsers.InlineNote,
4520     NoteRef      = larsers.NoteRef,
4521     Citations    = larsers.Citations,
4522     Link         = larsers.Link,
4523     Image        = larsers.Image,
4524     Code         = larsers.Code,
4525     AutoLinkUrl  = larsers.AutoLinkUrl,
4526     AutoLinkEmail = larsers.AutoLinkEmail,
4527     InlineHtml   = larsers.InlineHtml,
4528     HtmlEntity   = larsers.HtmlEntity,
4529     EscapedChar  = larsers.EscapedChar,
4530     Smart        = larsers.Smart,
4531     Symbol       = larsers.Symbol,
4532 }
4533
4534 if not options.citations then
4535     syntax.Citations = parsers.fail
4536 end
4537
4538 if not options.contentBlocks then
4539     syntax.ContentBlock = parsers.fail
4540 end
4541
4542 if not options.codeSpans then
4543     syntax.Code = parsers.fail
4544 end

```

```

4545
4546 if not options.definitionLists then
4547     syntax.DefinitionList = parsers.fail
4548 end
4549
4550 if not options.fencedCode then
4551     syntax.FencedCode = parsers.fail
4552 end
4553
4554 if not options.footnotes then
4555     syntax.NoteRef = parsers.fail
4556 end
4557
4558 if not options.html then
4559     syntax.DisplayHtml = parsers.fail
4560     syntax.InlineHtml = parsers.fail
4561     syntax.HtmlEntity = parsers.fail
4562 end
4563
4564 if not options.inlineFootnotes then
4565     syntax.InlineNote = parsers.fail
4566 end
4567
4568 if not options.smartEllipses then
4569     syntax.Smart = parsers.fail
4570 end
4571
4572 if not options.pipeTables then
4573     syntax.PipeTable = parsers.fail
4574 end
4575
4576 local blocks_toplevel_t = util.table_copy(syntax)
4577 blocks_toplevel_t.Paragraph = larsers.ToplevelParagraph
4578 larsers.blocks_toplevel = Ct(blocks_toplevel_t)
4579
4580 larsers.blocks = Ct(syntax)
4581
4582 local inlines_t = util.table_copy(syntax)
4583 inlines_t[1] = "Inlines"
4584 inlines_t.Inlines = parsers.Inline^0 * (parsers.spacing^0 * parsers.eof / "")
4585 larsers.inlines = Ct(inlines_t)
4586
4587 local inlines_no_link_t = util.table_copy(inlines_t)
4588 inlines_no_link_t.Link = parsers.fail
4589 larsers.inlines_no_link = Ct(inlines_no_link_t)
4590
4591 local inlines_no_inline_note_t = util.table_copy(inlines_t)

```

```

4592 inlines_no_inline_note_t.InlineNote = parsers.fail
4593 larsers.inlines_no_inline_note = Ct(inlines_no_inline_note_t)
4594
4595 local inlines_nbsp_t = util.table_copy(inlines_t)
4596 inlines_nbsp_t.Endline = larsers.NonbreakingEndline
4597 inlines_nbsp_t.Space = larsers.NonbreakingSpace
4598 larsers.inlines_nbsp = Ct(inlines_nbsp_t)

```

3.1.5.15 Exported Conversion Function Define `reader->convert` as a function that converts markdown string `input` into a plain T_EX output and returns it. Note that the converter assumes that the input has UNIX line endings.

```

4599 function self.convert(input)
4600     references = {}

```

When determining the name of the cache file, create salt for the hashing function out of the package version and the passed options recognized by the Lua interface (see Section 2.1.2). The `cacheDir` option is disregarded.

```

4601     local opt_string = {}
4602     for k,_ in pairs(defaultOptions) do
4603         local v = options[k]
4604         if k ~= "cacheDir" then
4605             opt_string[#opt_string+1] = k .. "=" .. tostring(v)
4606         end
4607     end
4608     table.sort(opt_string)
4609     local salt = table.concat(opt_string, ",") .. "," .. metadata.version

```

Produce the cache file, transform its filename via the `writer->pack` method, and return the result.

```

4610     local name = util.cache(options.cacheDir, input, salt, function(input)
4611         return util.rope_to_string(parse_blocks_toplevel(input)) .. writer.eof
4612     end, ".md" .. writer.suffix)
4613     return writer.pack(name)
4614 end
4615 return self
4616 end

```

3.1.6 Conversion from Markdown to Plain T_EX

The `new` method returns the `reader->convert` function of a reader object associated with the Lua interface options (see Section 2.1.2) `options` and with a writer object associated with `options`.

```

4617 function M.new(options)
4618     local writer = M.writer.new(options)
4619     local reader = M.reader.new(writer, options)
4620     return reader.convert

```

```

4621 end
4622
4623 return M

```

3.1.7 Command-Line Implementation

The command-line implementation provides the actual conversion routine for the command-line interface described in Section 2.1.5.

```

4624
4625 local input
4626 if input_filename then
4627   local input_file = io.open(input_filename, "r")
4628   input = assert(input_file:read("*a"))
4629   input_file:close()
4630 else
4631   input = assert(io.read("*a"))
4632 end
4633

```

First, ensure that the `options.cacheDir` directory exists.

```

4634 local lfs = require("lfs")
4635 if options.cacheDir and not lfs.isdir(options.cacheDir) then
4636   assert(lfs.mkdir(options["cacheDir"]))
4637 end
4638
4639 local kpse = require("kpse")
4640 kpse.set_program_name("luatex")
4641 local md = require("markdown")

```

Since we are loading the rest of the Lua implementation dynamically, check that both the `markdown` module and the command line implementation are the same version.

```

4642 if metadata.version ~= md.metadata.version then
4643   warn("markdown-cli.lua " .. metadata.version .. " used with " ..
4644     "markdown.lua " .. md.metadata.version .. ".")
4645 end
4646 local convert = md.new(options)

```

Since the Lua converter expects UNIX line endings, normalize the input. Also add a line ending at the end of the file in case the input file has none.

```

4647 local output = convert(input:gsub("\r\n?", "\n") .. "\n")
4648
4649 if output_filename then
4650   local output_file = io.open(output_filename, "w")
4651   assert(output_file:write(output))
4652   assert(output_file:close())
4653 else
4654   assert(io.write(output))
4655 end

```

3.2 Plain T_EX Implementation

The plain T_EX implementation provides macros for the interfacing between T_EX and Lua and for the buffering of input text. These macros are then used to implement the macros for the conversion from markdown to plain T_EX exposed by the plain T_EX interface (see Section 2.2).

3.2.1 Logging Facilities

```
4656 \def\markdownInfo#1{%
4657   \immediate\write-1{(1.\the\inputlineno) markdown.tex info: #1.}}%
4658 \def\markdownWarning#1{%
4659   \immediate\write16{(1.\the\inputlineno) markdown.tex warning: #1}}%
4660 \def\markdownError#1#2{%
4661   \errhelp{#2.}}%
4662   \errmessage{(1.\the\inputlineno) markdown.tex error: #1}}%
```

3.2.2 Token Renderer Prototypes

The following definitions should be considered placeholder.

```
4663 \def\markdownRendererInterblockSeparatorPrototype{\par}%
4664 \def\markdownRendererLineBreakPrototype{\hfil\break}%
4665 \let\markdownRendererEllipsisPrototype\dots
4666 \def\markdownRendererNbspPrototype{~}%
4667 \def\markdownRendererLeftBracePrototype{\char`\{}%
4668 \def\markdownRendererRightBracePrototype{\char`\}}%
4669 \def\markdownRendererDollarSignPrototype{\char`\$}%
4670 \def\markdownRendererPercentSignPrototype{\char`\}%
4671 \def\markdownRendererAmpersandPrototype{\char`\&%
4672 \def\markdownRendererUnderscorePrototype{\char`\_}%
4673 \def\markdownRendererHashPrototype{\char`\#}%
4674 \def\markdownRendererCircumflexPrototype{\char`\^}%
4675 \def\markdownRendererBackslashPrototype{\char`\}%
4676 \def\markdownRendererTildePrototype{\char`\~}%
4677 \def\markdownRendererPipePrototype{|}%
4678 \def\markdownRendererCodeSpanPrototype#1{{\tt#1}}%
4679 \def\markdownRendererLinkPrototype#1#2#3#4{#2}%
4680 \def\markdownRendererContentBlockPrototype#1#2#3#4{%
4681   \markdownInput{#3}}%
4682 \def\markdownRendererContentBlockOnlineImagePrototype{%
4683   \markdownRendererImage}%
4684 \def\markdownRendererContentBlockCodePrototype#1#2#3#4#5{%
4685   \markdownRendererInputFencedCode{#3}{#2}}%
4686 \def\markdownRendererImagePrototype#1#2#3#4{#2}%
4687 \def\markdownRendererULBeginPrototype{}%
4688 \def\markdownRendererULBeginTightPrototype{}%
4689 \def\markdownRendererULItemPrototype{}%
```



```

4690 \def\markdownRendererUListItemEndPrototype{}%
4691 \def\markdownRendererUListEndPrototype{}%
4692 \def\markdownRendererUListEndTightPrototype{}%
4693 \def\markdownRendererOListBeginPrototype{}%
4694 \def\markdownRendererOListBeginTightPrototype{}%
4695 \def\markdownRendererOListItemPrototype{}%
4696 \def\markdownRendererOListItemWithNumberPrototype#1{}%
4697 \def\markdownRendererOListItemEndPrototype{}%
4698 \def\markdownRendererOListEndPrototype{}%
4699 \def\markdownRendererOListEndTightPrototype{}%
4700 \def\markdownRendererDListBeginPrototype{}%
4701 \def\markdownRendererDListBeginTightPrototype{}%
4702 \def\markdownRendererDListItemPrototype#1{#1}%
4703 \def\markdownRendererDListItemEndPrototype{}%
4704 \def\markdownRendererDListDefinitionBeginPrototype{}%
4705 \def\markdownRendererDListDefinitionEndPrototype{\par}%
4706 \def\markdownRendererDListEndPrototype{}%
4707 \def\markdownRendererDListEndTightPrototype{}%
4708 \def\markdownRendererEmphasisPrototype#1{{\it#1}}%
4709 \def\markdownRendererStrongEmphasisPrototype#1{{\bf#1}}%
4710 \def\markdownRendererBlockQuoteBeginPrototype{\par\begingroup\it}%
4711 \def\markdownRendererBlockQuoteEndPrototype{\endgroup\par}%
4712 \def\markdownRendererInputVerbatimPrototype#1{%
4713   \par{\tt\input#1\relax{}}\par}%
4714 \def\markdownRendererInputFencedCodePrototype#1#2{%
4715   \markdownRendererInputVerbatimPrototype{#1}}%
4716 \def\markdownRendererHeadingOnePrototype#1{#1}%
4717 \def\markdownRendererHeadingTwoPrototype#1{#1}%
4718 \def\markdownRendererHeadingThreePrototype#1{#1}%
4719 \def\markdownRendererHeadingFourPrototype#1{#1}%
4720 \def\markdownRendererHeadingFivePrototype#1{#1}%
4721 \def\markdownRendererHeadingSixPrototype#1{#1}%
4722 \def\markdownRendererHorizontalRulePrototype{}%
4723 \def\markdownRendererFootnotePrototype#1{#1}%
4724 \def\markdownRendererCitePrototype#1{}%
4725 \def\markdownRendererTextCitePrototype#1{}%

```

3.2.3 Lua Snippets

The `\markdownLuaOptions` macro expands to a Lua table that contains the plain TeX options (see Section 2.2.2) in a format recognized by Lua (see Section 2.1.2).

```

4726 \def\markdownLuaOptions{%
4727   \ifx\markdownOptionBlankBeforeBlockquote\undefined\else
4728     blankBeforeBlockquote = \markdownOptionBlankBeforeBlockquote,
4729   \fi
4730   \ifx\markdownOptionBlankBeforeCodeFence\undefined\else
4731     blankBeforeCodeFence = \markdownOptionBlankBeforeCodeFence,

```

```

4732 \fi
4733 \ifx\markdownOptionBlankBeforeHeading\undefined\else
4734   blankBeforeHeading = \markdownOptionBlankBeforeHeading,
4735 \fi
4736 \ifx\markdownOptionBreakableBlockquotes\undefined\else
4737   breakableBlockquotes = \markdownOptionBreakableBlockquotes,
4738 \fi
4739   cacheDir = "\markdownOptionCacheDir",
4740 \ifx\markdownOptionCitations\undefined\else
4741   citations = \markdownOptionCitations,
4742 \fi
4743 \ifx\markdownOptionCitationNbsps\undefined\else
4744   citationNbsps = \markdownOptionCitationNbsps,
4745 \fi
4746 \ifx\markdownOptionCodeSpans\undefined\else
4747   codeSpans = \markdownOptionCodeSpans,
4748 \fi
4749 \ifx\markdownOptionContentBlocks\undefined\else
4750   contentBlocks = \markdownOptionContentBlocks,
4751 \fi
4752 \ifx\markdownOptionContentBlocksLanguageMap\undefined\else
4753   contentBlocksLanguageMap =
4754     "\markdownOptionContentBlocksLanguageMap",
4755 \fi
4756 \ifx\markdownOptionDefinitionLists\undefined\else
4757   definitionLists = \markdownOptionDefinitionLists,
4758 \fi
4759 \ifx\markdownOptionFootnotes\undefined\else
4760   footnotes = \markdownOptionFootnotes,
4761 \fi
4762 \ifx\markdownOptionFencedCode\undefined\else
4763   fencedCode = \markdownOptionFencedCode,
4764 \fi
4765 \ifx\markdownOptionHashEnumerators\undefined\else
4766   hashEnumerators = \markdownOptionHashEnumerators,
4767 \fi
4768 \ifx\markdownOptionHeaderAttributes\undefined\else
4769   headerAttributes = \markdownOptionHeaderAttributes,
4770 \fi
4771 \ifx\markdownOptionHtml\undefined\else
4772   html = \markdownOptionHtml,
4773 \fi
4774 \ifx\markdownOptionHybrid\undefined\else
4775   hybrid = \markdownOptionHybrid,
4776 \fi
4777 \ifx\markdownOptionInlineFootnotes\undefined\else
4778   inlineFootnotes = \markdownOptionInlineFootnotes,

```

```

4779 \fi
4780 \ifx\markdownOptionPipeTables\undefined\else
4781   pipeTables = \markdownOptionPipeTables,
4782 \fi
4783 \ifx\markdownOptionPreserveTabs\undefined\else
4784   preserveTabs = \markdownOptionPreserveTabs,
4785 \fi
4786 \ifx\markdownOptionShiftHeadings\undefined\else
4787   shiftHeadings = "\markdownOptionShiftHeadings",
4788 \fi
4789 \ifx\markdownOptionSlice\undefined\else
4790   slice = "\markdownOptionSlice",
4791 \fi
4792 \ifx\markdownOptionSmartEllipses\undefined\else
4793   smartEllipses = \markdownOptionSmartEllipses,
4794 \fi
4795 \ifx\markdownOptionStartNumber\undefined\else
4796   startNumber = \markdownOptionStartNumber,
4797 \fi
4798 \ifx\markdownOptionTableCaptions\undefined\else
4799   tableCaptions = \markdownOptionTableCaptions,
4800 \fi
4801 \ifx\markdownOptionTightLists\undefined\else
4802   tightLists = \markdownOptionTightLists,
4803 \fi
4804 \ifx\markdownOptionUnderscores\undefined\else
4805   underscores = \markdownOptionUnderscores,
4806 \fi}
4807 }%

```

The `\markdownPrepare` macro contains the Lua code that is executed prior to any conversion from markdown to plain T_EX. It exposes the `convert` function for the use by any further Lua code.

```
4808 \def\markdownPrepare{%
```

First, ensure that the `\markdownOptionCacheDir` directory exists.

```

4809 local lfs = require("lfs")
4810 local cacheDir = "\markdownOptionCacheDir"
4811 if not lfs.isdir(cacheDir) then
4812   assert(lfs.mkdir(cacheDir))
4813 end

```

Next, load the `markdown` module and create a converter function using the plain T_EX options, which were serialized to a Lua table via the `\markdownLuaOptions` macro.

```

4814 local md = require("markdown")
4815 local convert = md.new(\markdownLuaOptions)
4816 }%

```

3.2.4 Buffering Markdown Input

The macros `\markdownInputFileStream` and `\markdownOutputFileStream` contain the number of the input and output file streams that will be used for the IO operations of the package.

```
4817 \csname newread\endcsname\markdownInputFileStream
4818 \csname newwrite\endcsname\markdownOutputFileStream
```

The `\markdownReadAndConvertTab` macro contains the tab character literal.

```
4819 \begingroup
4820 \catcode`\^^I=12%
4821 \gdef\markdownReadAndConvertTab{^^I}%
4822 \endgroup
```

The `\markdownReadAndConvert` macro is largely a rewrite of the $\text{\LaTeX} 2_{\epsilon}$ `\filecontents` macro to plain \TeX .

```
4823 \begingroup
```

Make the newline and tab characters active and swap the character codes of the backslash symbol (`\`) and the pipe symbol (`|`), so that we can use the backslash as an ordinary character inside the macro definition. Likewise, swap the character codes of the percent sign (`%`) and the ampersand (`@`), so that we can remove percent signs from the beginning of lines when `\markdownOptionStripPercentSigns` is `true`.

```
4824 \catcode`\^^M=13%
4825 \catcode`\^^I=13%
4826 \catcode`|=0%
4827 \catcode`\=12%
4828 |catcode`%=14%
4829 |catcode`|=12@
4830 |gdef|markdownReadAndConvert#1#2{@
4831   |begingroup@
```

Open the `\markdownOptionInputTempFileName` file for writing.

```
4832   |immediate|openout|markdownOutputFileStream@
4833   |markdownOptionInputTempFileName|relax@
4834   |markdownInfo{Buffering markdown input into the temporary @
4835   input file "|markdownOptionInputTempFileName" and scanning @
4836   for the closing token sequence "#1"}@
```

Locally change the category of the special plain \TeX characters to *other* in order to prevent unwanted interpretation of the input. Change also the category of the space character, so that we can retrieve it unaltered.

```
4837   |def|do##1{|catcode`##1=12}|dospecials@
4838   |catcode`|=12@
4839   |markdownMakeOther@
```

The `\markdownReadAndConvertStripPercentSigns` macro will process the individual lines of output, stripping away leading percent signs (`%`) when

`\markdownOptionStripPercentSigns` is `true`. Notice the use of the comments (`@`) to ensure that the entire macro is at a single line and therefore no (active) newline symbols (`^^M`) are produced.

```

4840     |def|markdownReadAndConvertStripPercentSign##1{@
4841         |markdownIfOption{StripPercentSigns}@
4842         |if##1%{@
4843             |expandafter|expandafter|expandafter@
4844             |markdownReadAndConvertProcessLine@
4845         |else@
4846             |expandafter|expandafter|expandafter@
4847             |markdownReadAndConvertProcessLine@
4848             |expandafter|expandafter|expandafter##1@
4849         |fi@
4850     |else@
4851         |expandafter@
4852         |markdownReadAndConvertProcessLine@
4853         |expandafter##1@
4854     |fi}@

```

The `\markdownReadAndConvertProcessLine` macro will process the individual lines of output. Notice the use of the comments (`@`) to ensure that the entire macro is at a single line and therefore no (active) newline symbols (`^^M`) are produced.

```

4855     |def|markdownReadAndConvertProcessLine##1#1##2#1##3|relax{@

```

When the ending token sequence does not appear in the line, store the line in the `\markdownOptionInputTempFileName` file.

```

4856         |ifx|relax##3|relax@
4857         |immediate|write|markdownOutputFileStream{##1}@
4858     |else@

```

When the ending token sequence appears in the line, make the next newline character close the `\markdownOptionInputTempFileName` file, return the character categories back to the former state, convert the `\markdownOptionInputTempFileName` file from markdown to plain TeX, `\input` the result of the conversion, and expand the ending control sequence.

```

4859     |def^^M{@
4860         |markdownInfo{The ending token sequence was found}@
4861         |immediate|closeout|markdownOutputFileStream@
4862         |endgroup@
4863         |markdownInput{@
4864             |markdownOptionOutputDir@
4865             /|markdownOptionInputTempFileName@
4866         }@
4867         #2}@
4868     |fi@

```

Repeat with the next line.

```

4869     ^^M}@

```

Make the tab character active at expansion time and make it expand to a literal tab character.

```
4870 |catcode`|^I=13@
4871 |def^^I{|markdownReadAndConvertTab}@
```

Make the newline character active at expansion time and make it consume the rest of the line on expansion. Throw away the rest of the first line and pass the second line to the `\markdownReadAndConvertProcessLine` macro.

```
4872 |catcode`|^M=13@
4873 |def^^M##1^^M{@
4874 |def^^M####1^^M{@
4875 |markdownReadAndConvertStripPercentSign####1#1#1|relax}@
4876 ^^M}@
4877 ^^M}@
```

Reset the character categories back to the former state.

```
4878 |endgroup
```

3.2.5 Lua Shell Escape Bridge

The following \TeX code is intended for \TeX engines that do not provide direct access to Lua, but expose the shell of the operating system. This corresponds to the `\markdownMode` values of 0 and 1.

The `\markdownLuaExecute` macro defined here and in Section 3.2.6 are meant to be indistinguishable to the remaining code.

The package assumes that although the user is not using the Lua \TeX engine, their \TeX distribution contains it, and uses shell access to produce and execute Lua scripts using the \TeX Lua interpreter [2, Section 3.1.1].

```
4879 \ifnum\markdownMode<2\relax
4880 \ifnum\markdownMode=0\relax
4881 \markdownInfo{Using mode 0: Shell escape via write18}%
4882 \else
4883 \markdownInfo{Using mode 1: Shell escape via os.execute}%
4884 \fi
```

The `\markdownExecuteShellEscape` macro contains the numeric value indicating whether the shell access is enabled (1), disabled (0), or restricted (2).

Inherit the value of the the `\pdfshellescape` (Lua \TeX , Pdf \TeX) or the `\shellescape` (X \TeX) commands. If neither of these commands is defined and Lua is available, attempt to access the `status.shell_escape` configuration item.

If you cannot detect, whether the shell access is enabled, act as if it were.

```
4885 \ifx\pdfshellescape\undefined
4886 \ifx\shellescape\undefined
4887 \ifnum\markdownMode=0\relax
4888 \def\markdownExecuteShellEscape{1}%
4889 \else
```

```

4890     \def\markdownExecuteShellEscape{%
4891         \directlua{tex.sprint(status.shell_escape or "1")}}%
4892     \fi
4893     \else
4894         \let\markdownExecuteShellEscape\shellescape
4895     \fi
4896 \else
4897     \let\markdownExecuteShellEscape\pdfshellescape
4898 \fi

```

The `\markdownExecuteDirect` macro executes the code it has received as its first argument by writing it to the output file stream 18, if Lua is unavailable, or by using the Lua `os.execute` method otherwise.

```

4899 \ifnum\markdownMode=0\relax
4900     \def\markdownExecuteDirect#1{\immediate\write18{#1}}%
4901 \else
4902     \def\markdownExecuteDirect#1{%
4903         \directlua{os.execute("\luaescapestring{#1}")}}%
4904 \fi

```

The `\markdownExecute` macro is a wrapper on top of `\markdownExecuteDirect` that checks the value of `\markdownExecuteShellEscape` and prints an error message if the shell is inaccessible.

```

4905 \def\markdownExecute#1{%
4906     \ifnum\markdownExecuteShellEscape=1\relax
4907         \markdownExecuteDirect{#1}%
4908     \else
4909         \markdownError{I can not access the shell}{Either run the TeX
4910             compiler with the --shell-escape or the --enable-write18 flag,
4911             or set shell_escape=t in the texmf.cnf file}%
4912     \fi}%

```

The `\markdownLuaExecute` macro executes the Lua code it has received as its first argument. The Lua code may not directly interact with the TeX engine, but it can use the `print` function in the same manner it would use the `tex.print` method.

```

4913 \begingroup

```

Swap the category code of the backslash symbol and the pipe symbol, so that we may use the backslash symbol freely inside the Lua code.

```

4914     \catcode`\|=0%
4915     \catcode`\|=12%
4916     \gdef\markdownLuaExecute#1{%

```

Create the file `\markdownOptionHelperScriptFileName` and fill it with the input Lua code prepended with `kpathsea` initialization, so that Lua modules from the TeX distribution are available.

```

4917         \immediate\openout\markdownOutputFileStream=%
4918         \markdownOptionHelperScriptFileName

```

```

4919 |markdownInfo{Writing a helper Lua script to the file
4920   "|markdownOptionHelperScriptFileName"}%
4921 |immediate|write|markdownOutputFileStream{%
4922   local ran_ok, error = pcall(function()
4923     local kpse = require("kpse")
4924     kpse.set_program_name("luatex")
4925     #1
4926     end)

```

If there was an error, use the file `\markdownOptionErrorTempFileName` to store the error message.

```

4927   if not ran_ok then
4928     local file = io.open("%
4929     |markdownOptionOutputDir
4930     /|markdownOptionErrorTempFileName", "w")
4931     if file then
4932       file:write(error .. "\n")
4933       file:close()
4934     end
4935     print('\|markdownError{An error was encountered while executing
4936     Lua code}{For further clues, examine the file
4937     "|markdownOptionOutputDir
4938     /|markdownOptionErrorTempFileName"}')
4939   end}%
4940 |immediate|closeout|markdownOutputFileStream

```

Execute the generated `\markdownOptionHelperScriptFileName` Lua script using the `TeXLua` binary and store the output in the `\markdownOptionOutputTempFileName` file.

```

4941 |markdownInfo{Executing a helper Lua script from the file
4942   "|markdownOptionHelperScriptFileName" and storing the result in the
4943   file "|markdownOptionOutputTempFileName"}%
4944 |markdownExecute{texlua "|markdownOptionOutputDir
4945   /|markdownOptionHelperScriptFileName" > %
4946   "|markdownOptionOutputDir
4947   /|markdownOptionOutputTempFileName"}%

```

`\input` the generated `\markdownOptionOutputTempFileName` file.

```

4948   |input|markdownOptionOutputTempFileName|relax}%
4949 |endgroup

```

3.2.6 Direct Lua Access

The following `TeX` code is intended for `TeX` engines that provide direct access to Lua (Lua`TeX`). The macro `\markdownLuaExecute` defined here and in Section 3.2.5 are meant to be indistinguishable to the remaining code. This corresponds to the `\markdownMode` value of 2.


```

4950 \else
4951 \markdownInfo{Using mode 2: Direct Lua access}%

```

The direct Lua access version of the `\markdownLuaExecute` macro is defined in terms of the `\directlua` primitive. The `print` function is set as an alias to the `\tex.print` method in order to mimic the behaviour of the `\markdownLuaExecute` definition from Section 3.2.5,

```

4952 \def\markdownLuaExecute#1{\directlua{local print = tex.print #1}}%
4953 \fi

```

3.2.7 Typesetting Markdown

The `\markdownInput` macro uses an implementation of the `\markdownLuaExecute` macro to convert the contents of the file whose filename it has received as its single argument from markdown to plain \TeX .

```

4954 \begingroup

```

Swap the category code of the backslash symbol and the pipe symbol, so that we may use the backslash symbol freely inside the Lua code.

```

4955 \catcode`\|=0%
4956 \catcode`\|=12%
4957 |gdef|markdownInput#1{%
4958   |markdownInfo{Including markdown document "#1"}%

```

Attempt to open the markdown document to record it in the `.log` and `.fls` files. This allows external programs such as \LaTeX Mk to track changes to the markdown document.

```

4959   |openin|markdownInputFileStream#1
4960   |closein|markdownInputFileStream
4961   |markdownLuaExecute{%
4962     |markdownPrepare
4963     local input = assert(io.open("#1", "r"):read("*a"))

```

Since the Lua converter expects UNIX line endings, normalize the input. Also add a line ending at the end of the file in case the input file has none.

```

4964     print(convert(input:gsub("\r\n?", "\n") .. "\n"))}%
4965 |endgroup

```

3.3 \LaTeX Implementation

The \LaTeX implementation makes use of the fact that, apart from some subtle differences, \LaTeX implements the majority of the plain \TeX format [7, Section 9]. As a consequence, we can directly reuse the existing plain \TeX implementation.

```

4966 \input markdown
4967 \def\markdownVersionSpace{ }%
4968 \ProvidesPackage{markdown}[\markdownLastModified\markdownVersionSpace v%
4969 \markdownVersion\markdownVersionSpace markdown renderer]%

```

3.3.1 Logging Facilities

The \LaTeX implementation redefines the plain \TeX logging macros (see Section 3.2.1) to use the \LaTeX `\PackageInfo`, `\PackageWarning`, and `\PackageError` macros.

```
4970 \renewcommand\markdownInfo[1]{\PackageInfo{markdown}{#1}}%
4971 \renewcommand\markdownWarning[1]{\PackageWarning{markdown}{#1}}%
4972 \renewcommand\markdownError[2]{\PackageError{markdown}{#1}{#2.}}%
```

3.3.2 Typesetting Markdown

The `\markdownInputPlainTeX` macro is used to store the original plain \TeX implementation of the `\markdownInput` macro. The `\markdownInput` is then redefined to accept an optional argument with options recognized by the \LaTeX interface (see Section 2.3.2).

```
4973 \let\markdownInputPlainTeX\markdownInput
4974 \renewcommand\markdownInput[2] [] {%
4975   \begingroup
4976     \markdownSetup{#1}%
4977     \markdownInputPlainTeX{#2}%
4978   \endgroup}%
```

The `markdown`, and `markdown*` \LaTeX environments are implemented using the `\markdownReadAndConvert` macro.

```
4979 \renewenvironment{markdown}{%
4980   \markdownReadAndConvert@markdown{}}\relax
4981 \renewenvironment{markdown*}[1]{%
4982   \markdownSetup{#1}%
4983   \markdownReadAndConvert@markdown*}\relax
4984 \begingroup
```

Locally swap the category code of the backslash symbol with the pipe symbol, and of the left (`{`) and right brace (`}`) with the less-than (`<`) and greater-than (`>`) signs. This is required in order that all the special symbols that appear in the first argument of the `markdownReadAndConvert` macro have the category code *other*.

```
4985 \catcode`\|=0\catcode`\<=1\catcode`\>=2%
4986 \catcode`\={12\catcode`\|={12\catcode`\}=12%
4987 |gdef|markdownReadAndConvert@markdown#1<%
4988   |markdownReadAndConvert<\end{markdown#1}>%
4989                               <|end<markdown#1>>>%
4990 |endgroup
```

3.3.3 Options

The supplied package options are processed using the `\markdownSetup` macro.

```
4991 \DeclareOption*{%
4992   \expandafter\markdownSetup\expandafter{\CurrentOption}}%
```

```
4993 \ProcessOptions\relax
```

After processing the options, activate the `renderers` and `rendererPrototypes` keys.

```
4994 \define@key{markdownOptions}{renderers}{%
4995   \setkeys{markdownRenderers}{#1}%
4996   \def\KV@prefix{KV@markdownOptions@}}%
4997 \define@key{markdownOptions}{rendererPrototypes}{%
4998   \setkeys{markdownRendererPrototypes}{#1}%
4999   \def\KV@prefix{KV@markdownOptions@}}%
```

3.3.4 Token Renderer Prototypes

The following configuration should be considered placeholder.

If the `\markdownOptionTightLists` macro expands to `false`, do not load the `paralist` package. This is necessary for $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$ document classes that do not play nice with `paralist`, such as `beamer`. If the `\markdownOptionTightLists` is undefined and the `beamer` document class is in use, then do not load the `paralist` package either.

```
5000 \ifx\markdownOptionTightLists\undefined
5001   \@ifclassloaded{beamer}{}{
5002     \RequirePackage{paralist}}
5003 \else
5004   \ifthenelse{equal{\markdownOptionTightLists}{false}}{}{
5005     \RequirePackage{paralist}}
5006 \fi
```

If we loaded the `paralist` package, define the respective renderer prototypes to make use of the capabilities of the package. Otherwise, define the renderer prototypes to fall back on the corresponding renderers for the non-tight lists.

```
5007 \@ifpackageloaded{paralist}{
5008   \markdownSetup{rendererPrototypes={
5009     ulBeginTight = {\begin{compactitem}},
5010     ulEndTight = {\end{compactitem}},
5011     olBeginTight = {\begin{compactenum}},
5012     olEndTight = {\end{compactenum}},
5013     dlBeginTight = {\begin{compactdesc}},
5014     dlEndTight = {\end{compactdesc}}}}
5015 }{
5016   \markdownSetup{rendererPrototypes={
5017     ulBeginTight = {\markdownRendererUlBegin},
5018     ulEndTight = {\markdownRendererUlEnd},
5019     olBeginTight = {\markdownRendererOlBegin},
5020     olEndTight = {\markdownRendererOlEnd},
5021     dlBeginTight = {\markdownRendererDlBegin},
5022     dlEndTight = {\markdownRendererDlEnd}}}}
```

```

5023 \markdownSetup{rendererPrototypes={
5024   lineBreak = {\},
5025   leftBrace = {\textbraceleft},
5026   rightBrace = {\textbraceright},
5027   dollarSign = {\textdollar},
5028   underscore = {\textunderscore},
5029   circumflex = {\textasciicircum},
5030   backslash = {\textbackslash},
5031   tilde = {\textasciitilde},
5032   pipe = {\textbar},
5033   codeSpan = {\texttt{#1}},
5034   contentBlock = {%
5035     \ifthenelse{\equal{#1}{csv}}{%
5036       \begin{table}%
5037         \begin{center}%
5038           \csvautotabular{#3}%
5039         \end{center}
5040       \ifx\empty#4\empty\else
5041         \caption{#4}%
5042       \fi
5043       \label{tab:#1}%
5044     \end{table}}{%
5045     \markdownInput{#3}}},
5046   image = {%
5047     \begin{figure}%
5048       \begin{center}%
5049         \includegraphics{#3}%
5050       \end{center}%
5051       \ifx\empty#4\empty\else
5052         \caption{#4}%
5053       \fi
5054       \label{fig:#1}%
5055     \end{figure}},
5056   ulBegin = {\begin{itemize}},
5057   ulItem = {\item},
5058   ulEnd = {\end{itemize}},
5059   olBegin = {\begin{enumerate}},
5060   olItem = {\item},
5061   olItemWithNumber = {\item[#1.]},
5062   olEnd = {\end{enumerate}},
5063   dlBegin = {\begin{description}},
5064   dlItem = {\item[#1]},
5065   dlEnd = {\end{description}},
5066   emphasis = {\emph{#1}},
5067   blockQuoteBegin = {\begin{quotation}},
5068   blockQuoteEnd = {\end{quotation}},
5069   inputVerbatim = {\VerbatimInput{#1}},

```

```

5070 inputFencedCode = {%
5071   \ifx\relax#2\relax
5072     \VerbatimInput{#1}%
5073   \else
5074     \ifx\minted@code\undefined
5075     \ifx\lst@version\undefined
5076       \markdownRendererInputFencedCode{#1}{-%

```

When the listings package is loaded, use it for syntax highlighting.

```

5077     \else
5078       \lstinputlisting[language=#2]{#1}%
5079     \fi

```

When the minted package is loaded, use it for syntax highlighting. The minted package is preferred over listings.

```

5080     \else
5081       \inputminted{#2}{#1}%
5082     \fi
5083   \fi},
5084   horizontalRule = {\noindent\rule[0.5ex]{\linewidth}{1pt}},
5085   footnote = {\footnote{#1}}}}

```

Support the nesting of strong emphasis.

```

5086 \newif\ifmarkdownLATEXStrongEmphasisNested
5087 \markdownLATEXStrongEmphasisNestedfalse
5088 \markdownSetup{rendererPrototypes={
5089   strongEmphasis = {%
5090     \ifmarkdownLATEXStrongEmphasisNested
5091     \markdownLATEXStrongEmphasisNestedfalse
5092     \textmd{#1}%
5093     \markdownLATEXStrongEmphasisNestedtrue
5094   \else
5095     \markdownLATEXStrongEmphasisNestedtrue
5096     \textbf{#1}%
5097     \markdownLATEXStrongEmphasisNestedfalse
5098   \fi}}}

```

Support L^AT_EX document classes that do not provide chapters.

```

5099 \ifx\chapter\undefined
5100   \markdownSetup{rendererPrototypes = {
5101     headingOne = {\section{#1}},
5102     headingTwo = {\subsection{#1}},
5103     headingThree = {\subsubsection{#1}},
5104     headingFour = {\paragraph{#1}\leavevmode},
5105     headingFive = {\subparagraph{#1}\leavevmode}}}
5106 \else
5107   \markdownSetup{rendererPrototypes = {
5108     headingOne = {\chapter{#1}},
5109     headingTwo = {\section{#1}},

```

```

5110 headingThree = {\subsection{#1}},
5111 headingFour = {\subsubsection{#1}},
5112 headingFive = {\paragraph{#1}\leavevmode},
5113 headingSix = {\subparagraph{#1}\leavevmode}}
5114 \fi

```

There is a basic implementation for citations that uses the L^AT_EX `\cite` macro. There are also implementations that use the natbib `\citep`, and `\citet` macros, and the BibL^AT_EX `\autocites` and `\textcites` macros. These implementations will be used, when the respective packages are loaded.

```

5115 \newcount\markdownLaTeXCitationsCounter
5116
5117 % Basic implementation
5118 \def\markdownLaTeXBasicCitations#1#2#3#4#5#6{%
5119   \advance\markdownLaTeXCitationsCounter by 1\relax
5120   \ifx\relax#4\relax
5121     \ifx\relax#5\relax
5122       \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
5123         \cite{#1#2#6}% Without prenotes and postnotes, just accumulate cites
5124         \expandafter\expandafter\expandafter
5125         \expandafter\expandafter\expandafter\expandafter
5126         \@gobblethree
5127       \fi
5128     \else% Before a postnote (#5), dump the accumulator
5129       \ifx\relax#1\relax\else
5130         \cite{#1}%
5131       \fi
5132       \cite[#5]{#6}%
5133       \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
5134       \else
5135         \expandafter\expandafter\expandafter
5136         \expandafter\expandafter\expandafter\expandafter
5137         \expandafter\expandafter\expandafter
5138         \expandafter\expandafter\expandafter\expandafter
5139         \markdownLaTeXBasicCitations
5140       \fi
5141       \expandafter\expandafter\expandafter
5142       \expandafter\expandafter\expandafter\expandafter{%
5143       \expandafter\expandafter\expandafter
5144       \expandafter\expandafter\expandafter\expandafter}%
5145       \expandafter\expandafter\expandafter
5146       \expandafter\expandafter\expandafter\expandafter{%
5147       \expandafter\expandafter\expandafter
5148       \expandafter\expandafter\expandafter\expandafter}%
5149       \expandafter\expandafter\expandafter
5150       \@gobblethree
5151     \fi

```

```

5152 \else% Before a prenote (#4), dump the accumulator
5153 \ifx\relax#1\relax\else
5154 \cite{#1}%
5155 \fi
5156 \ifnum\markdownLaTeXCitationsCounter>1\relax
5157 \space % Insert a space before the prenote in later citations
5158 \fi
5159 #4-\expandafter\cite\ifx\relax#5\relax{#6}\else[#5]{#6}\fi
5160 \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
5161 \else
5162 \expandafter\expandafter\expandafter
5163 \expandafter\expandafter\expandafter\expandafter
5164 \markdownLaTeXBasicCitations
5165 \fi
5166 \expandafter\expandafter\expandafter{%
5167 \expandafter\expandafter\expandafter}%
5168 \expandafter\expandafter\expandafter{%
5169 \expandafter\expandafter\expandafter}%
5170 \expandafter
5171 \@gobblethree
5172 \fi\markdownLaTeXBasicCitations{#1#2#6},)
5173 \let\markdownLaTeXBasicTextCitations\markdownLaTeXBasicCitations
5174
5175 % Natbib implementation
5176 \def\markdownLaTeXNatbibCitations#1#2#3#4#5{%
5177 \advance\markdownLaTeXCitationsCounter by 1\relax
5178 \ifx\relax#3\relax
5179 \ifx\relax#4\relax
5180 \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
5181 \citep{#1,#5}% Without prenotes and postnotes, just accumulate cites
5182 \expandafter\expandafter\expandafter
5183 \expandafter\expandafter\expandafter\expandafter
5184 \@gobbletwo
5185 \fi
5186 \else% Before a postnote (#4), dump the accumulator
5187 \ifx\relax#1\relax\else
5188 \citep{#1}%
5189 \fi
5190 \citep[] [#4]{#5}%
5191 \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
5192 \else
5193 \expandafter\expandafter\expandafter
5194 \expandafter\expandafter\expandafter\expandafter
5195 \expandafter\expandafter\expandafter
5196 \expandafter\expandafter\expandafter\expandafter
5197 \markdownLaTeXNatbibCitations
5198 \fi

```

```

5199     \expandafter\expandafter\expandafter
5200     \expandafter\expandafter\expandafter\expandafter{%
5201     \expandafter\expandafter\expandafter
5202     \expandafter\expandafter\expandafter\expandafter}%
5203     \expandafter\expandafter\expandafter
5204     \@gobbletwo
5205     \fi
5206 \else% Before a prenote (#3), dump the accumulator
5207     \ifx\relax#1\relax\relax\else
5208         \citep{#1}%
5209     \fi
5210     \citep[#3][#4]{#5}%
5211     \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
5212     \else
5213         \expandafter\expandafter\expandafter
5214         \expandafter\expandafter\expandafter\expandafter
5215         \markdownLaTeXNatbibCitations
5216     \fi
5217     \expandafter\expandafter\expandafter{%
5218     \expandafter\expandafter\expandafter}%
5219     \expandafter
5220     \@gobbletwo
5221     \fi\markdownLaTeXNatbibCitations{#1,#5}}
5222 \def\markdownLaTeXNatbibTextCitations#1#2#3#4#5{%
5223     \advance\markdownLaTeXCitationsCounter by 1\relax
5224     \ifx\relax#3\relax
5225         \ifx\relax#4\relax
5226             \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
5227                 \citet{#1,#5}% Without prenotes and postnotes, just accumulate cites
5228                 \expandafter\expandafter\expandafter
5229                 \expandafter\expandafter\expandafter\expandafter
5230                 \@gobbletwo
5231             \fi
5232         \else% After a prenote or a postnote, dump the accumulator
5233             \ifx\relax#1\relax\else
5234                 \citet{#1}%
5235             \fi
5236             , \citet[#3][#4]{#5}%
5237             \ifnum\markdownLaTeXCitationsCounter<\markdownLaTeXCitationsTotal\relax
5238                 ,
5239             \else
5240                 \ifnum\markdownLaTeXCitationsCounter=\markdownLaTeXCitationsTotal\relax
5241                     ,
5242                 \fi
5243             \fi
5244             \expandafter\expandafter\expandafter
5245             \expandafter\expandafter\expandafter\expandafter

```



```

5246     \markdownLaTeXNatbibTextCitations
5247     \expandafter\expandafter\expandafter
5248     \expandafter\expandafter\expandafter\expandafter{%
5249     \expandafter\expandafter\expandafter
5250     \expandafter\expandafter\expandafter\expandafter}%
5251     \expandafter\expandafter\expandafter
5252     \@gobbletwo
5253     \fi
5254 \else% After a prenote or a postnote, dump the accumulator
5255     \ifx\relax#1\relax\relax\else
5256         \citet{#1}%
5257     \fi
5258     , \citet[#3][#4]{#5}%
5259     \ifnum\markdownLaTeXCitationsCounter<\markdownLaTeXCitationsTotal\relax
5260     ,
5261     \else
5262         \ifnum\markdownLaTeXCitationsCounter=\markdownLaTeXCitationsTotal\relax
5263         ,
5264         \fi
5265     \fi
5266     \expandafter\expandafter\expandafter
5267     \markdownLaTeXNatbibTextCitations
5268     \expandafter\expandafter\expandafter{%
5269     \expandafter\expandafter\expandafter}%
5270     \expandafter
5271     \@gobbletwo
5272 \fi\markdownLaTeXNatbibTextCitations{#1,#5}}
5273
5274 % BibLaTeX implementation
5275 \def\markdownLaTeXBibLaTeXCitations#1#2#3#4#5{%
5276     \advance\markdownLaTeXCitationsCounter by 1\relax
5277     \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
5278         \autocites#1[#3][#4]{#5}%
5279         \expandafter\@gobbletwo
5280     \fi\markdownLaTeXBibLaTeXCitations{#1[#3][#4]{#5}}
5281 \def\markdownLaTeXBibLaTeXTextCitations#1#2#3#4#5{%
5282     \advance\markdownLaTeXCitationsCounter by 1\relax
5283     \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
5284         \textcites#1[#3][#4]{#5}%
5285         \expandafter\@gobbletwo
5286     \fi\markdownLaTeXBibLaTeXTextCitations{#1[#3][#4]{#5}}
5287
5288 \markdownSetup{rendererPrototypes = {
5289     cite = {%
5290         \markdownLaTeXCitationsCounter=1%
5291         \def\markdownLaTeXCitationsTotal{#1}%
5292         \ifx\autocites\undefined

```

```

5293     \ifx\citep\undefined
5294     \expandafter\expandafter\expandafter
5295     \markdownLaTeXBasicCitations
5296     \expandafter\expandafter\expandafter{%
5297     \expandafter\expandafter\expandafter}%
5298     \expandafter\expandafter\expandafter{%
5299     \expandafter\expandafter\expandafter}%
5300     \else
5301     \expandafter\expandafter\expandafter
5302     \markdownLaTeXNatbibCitations
5303     \expandafter\expandafter\expandafter{%
5304     \expandafter\expandafter\expandafter}%
5305     \fi
5306     \else
5307     \expandafter\expandafter\expandafter
5308     \markdownLaTeXBibLaTeXCitations
5309     \expandafter{\expandafter}%
5310     \fi},
5311     textCite = {%
5312     \markdownLaTeXCitationsCounter=1%
5313     \def\markdownLaTeXCitationsTotal{#1}%
5314     \ifx\autocites\undefined
5315     \ifx\citep\undefined
5316     \expandafter\expandafter\expandafter
5317     \markdownLaTeXBasicTextCitations
5318     \expandafter\expandafter\expandafter{%
5319     \expandafter\expandafter\expandafter}%
5320     \expandafter\expandafter\expandafter{%
5321     \expandafter\expandafter\expandafter}%
5322     \else
5323     \expandafter\expandafter\expandafter
5324     \markdownLaTeXNatbibTextCitations
5325     \expandafter\expandafter\expandafter{%
5326     \expandafter\expandafter\expandafter}%
5327     \fi
5328     \else
5329     \expandafter\expandafter\expandafter
5330     \markdownLaTeXBibLaTeXTextCitations
5331     \expandafter{\expandafter}%
5332     \fi}}}}

```

Before consuming the parameters for the hyperlink renderer, we change the category code of the hash sign (#) to other, so that it cannot be mistaken for a parameter character. After the hyperlink has been typeset, we restore the original catcode.

```

5333 \def\markdownRendererLinkPrototype{%
5334   \begingroup
5335   \catcode\#=12

```

```

5336 \def\next##1##2##3##4{%
5337   ##1\footnote{%
5338     \ifx\empty##4\empty\else##4: \fi\texttt<\url{##3}\texttt>}%
5339   \endgroup}%
5340 \next}

```

There is a basic implementation of tables. If the booktabs package is loaded, then it is used to produce horizontal lines.

```

5341 \newcount\markdownLaTeXRowCounter
5342 \newcount\markdownLaTeXRowTotal
5343 \newcount\markdownLaTeXColumnCounter
5344 \newcount\markdownLaTeXColumnTotal
5345 \newtoks\markdownLaTeXTable
5346 \newtoks\markdownLaTeXTableAlignment
5347 \newtoks\markdownLaTeXTableEnd
5348 \@ifpackageloaded{booktabs}{
5349   \let\markdownLaTeXTopRule\toprule
5350   \let\markdownLaTeXMidRule\midrule
5351   \let\markdownLaTeXBottomRule\bottomrule
5352 }{
5353   \let\markdownLaTeXTopRule\hline
5354   \let\markdownLaTeXMidRule\hline
5355   \let\markdownLaTeXBottomRule\hline
5356 }
5357 \markdownSetup{rendererPrototypes={
5358   table = {%
5359     \markdownLaTeXTable={}%
5360     \markdownLaTeXTableAlignment={}%
5361     \markdownLaTeXTableEnd={%
5362       \markdownLaTeXBottomRule
5363     \end{tabular}}}%
5364     \ifx\empty#1\empty\else
5365       \addto@hook\markdownLaTeXTable{%
5366         \begin{table}
5367         \centering}%
5368       \addto@hook\markdownLaTeXTableEnd{%
5369         \caption{#1}
5370         \end{table}}}%
5371     \fi
5372     \addto@hook\markdownLaTeXTable{\begin{tabular}}%
5373     \markdownLaTeXRowCounter=0%
5374     \markdownLaTeXRowTotal=#2%
5375     \markdownLaTeXColumnTotal=#3%
5376     \markdownLaTeXRenderTableRow
5377   }
5378 }}
5379 \def\markdownLaTeXRenderTableRow#1{%

```

```

5380 \markdownLaTeXColumnCounter=0%
5381 \ifnum\markdownLaTeXRowCounter=0\relax
5382   \markdownLaTeXReadAlignments#1%
5383   \markdownLaTeXTable=\expandafter\expandafter\expandafter{%
5384     \expandafter\the\expandafter\markdownLaTeXTable\expandafter{%
5385       \the\markdownLaTeXTableAlignment}}%
5386   \addto@hook\markdownLaTeXTable{\markdownLaTeXTopRule}%
5387 \else
5388   \markdownLaTeXRenderTableCell#1%
5389 \fi
5390 \ifnum\markdownLaTeXRowCounter=1\relax
5391   \addto@hook\markdownLaTeXTable\markdownLaTeXMidRule
5392 \fi
5393 \advance\markdownLaTeXRowCounter by 1\relax
5394 \ifnum\markdownLaTeXRowCounter>\markdownLaTeXRowTotal\relax
5395   \markdownInfo{\the\markdownLaTeXTable}
5396   \markdownInfo{\the\markdownLaTeXTableEnd}
5397   \the\markdownLaTeXTable
5398   \the\markdownLaTeXTableEnd
5399   \expandafter\@gobble
5400 \fi\markdownLaTeXRenderTableRow}
5401 \def\markdownLaTeXReadAlignments#1{%
5402   \advance\markdownLaTeXColumnCounter by 1\relax
5403   \if#1d%
5404     \addto@hook\markdownLaTeXTableAlignment{1}%
5405   \else
5406     \addto@hook\markdownLaTeXTableAlignment{#1}%
5407   \fi
5408   \ifnum\markdownLaTeXColumnCounter<\markdownLaTeXColumnTotal\relax\else
5409     \expandafter\@gobble
5410   \fi\markdownLaTeXReadAlignments}
5411 \def\markdownLaTeXRenderTableCell#1{%
5412   \advance\markdownLaTeXColumnCounter by 1\relax
5413   \ifnum\markdownLaTeXColumnCounter<\markdownLaTeXColumnTotal\relax
5414     \addto@hook\markdownLaTeXTable{#1&}%
5415   \else
5416     \addto@hook\markdownLaTeXTable{#1\\}%
5417     \expandafter\@gobble
5418   \fi\markdownLaTeXRenderTableCell}

```

3.3.5 Miscellanea

When buffering user input, we should disable the bytes with the high bit set, since these are made active by the `inputenc` package. We will do this by redefining the `\markdownMakeOther` macro accordingly. The code is courtesy of Scott Pakin, the creator of the `filecontents` package.

```

5419 \newcommand\markdownMakeOther{%
5420   \count0=128\relax
5421   \loop
5422     \catcode\count0=11\relax
5423     \advance\count0 by 1\relax
5424   \ifnum\count0<256\repeat}%

```

3.4 ConT_EXt Implementation

The ConT_EXt implementation makes use of the fact that, apart from some subtle differences, the Mark II and Mark IV ConT_EXt formats *seem* to implement (the documentation is scarce) the majority of the plain T_EX format required by the plain T_EX implementation. As a consequence, we can directly reuse the existing plain T_EX implementation after supplying the missing plain T_EX macros.

```

5425 \def\dospecials{\do\ \do\\\do\{\do\}\do\$\do\&%
5426   \do\#\do\^\do\_ \do\% \do\~}%
5427 \input markdown

```

When buffering user input, we should disable the bytes with the high bit set, since these are made active by the `\enableregime` macro. We will do this by redefining the `\markdownMakeOther` macro accordingly. The code is courtesy of Scott Pakin, the creator of the filecontents L^AT_EX package.

```

5428 \def\markdownMakeOther{%
5429   \count0=128\relax
5430   \loop
5431     \catcode\count0=11\relax
5432     \advance\count0 by 1\relax
5433   \ifnum\count0<256\repeat

```

On top of that, make the pipe character (`|`) inactive during the scanning. This is necessary, since the character is active in ConT_EXt.

```

5434   \catcode`|=12}%

```

3.4.1 Logging Facilities

The ConT_EXt implementation redefines the plain T_EX logging macros (see Section 3.2.1) to use the ConT_EXt `\writestatus` macro.

```

5435 \def\markdownInfo#1{\writestatus{markdown}{#1.}}%
5436 \def\markdownWarning#1{\writestatus{markdown\space warn}{#1.}}%

```

3.4.2 Typesetting Markdown

The `\startmarkdown` and `\stopmarkdown` macros are implemented using the `\markdownReadAndConvert` macro.

```

5437 \begingroup

```

Locally swap the category code of the backslash symbol with the pipe symbol. This is required in order that all the special symbols that appear in the first argument of the `markdownReadAndConvert` macro have the category code *other*.

```
5438 \catcode`\|=0%
5439 \catcode`\|=12%
5440 |gdef|startmarkdown{%
5441     |markdownReadAndConvert{\stopmarkdown}%
5442     {|\stopmarkdown}}%
5443 |endgroup
```

3.4.3 Token Renderer Prototypes

The following configuration should be considered placeholder.

```
5444 \def\markdownRendererLineBreakPrototype{\blank}%
5445 \def\markdownRendererLeftBracePrototype{\textbraceleft}%
5446 \def\markdownRendererRightBracePrototype{\textbraceright}%
5447 \def\markdownRendererDollarSignPrototype{\textdollar}%
5448 \def\markdownRendererPercentSignPrototype{\percent}%
5449 \def\markdownRendererUnderscorePrototype{\textunderscore}%
5450 \def\markdownRendererCircumflexPrototype{\textcircumflex}%
5451 \def\markdownRendererBackslashPrototype{\textbackslash}%
5452 \def\markdownRendererTildePrototype{\textasciitilde}%
5453 \def\markdownRendererPipePrototype{\char`|}%
5454 \def\markdownRendererLinkPrototype#1#2#3#4{%
5455     \useURL[#1][#3][#4]#1\footnote[#1]{\ifx\empty#4\empty\else#4:
5456     \fi\texttt<\hyphenatedurl{#3}>}}%
5457 \usemodule[database]
5458 \defineseparatedlist
5459     [MarkdownConTeXtCSV]
5460     [separator={,},
5461     before=\bTABLE,after=\eTABLE,
5462     first=\bTR,last=\eTR,
5463     left=\bTD,right=\eTD]
5464 \def\markdownConTeXtCSV{csv}
5465 \def\markdownRendererContentBlockPrototype#1#2#3#4{%
5466     \def\markdownConTeXtCSV@arg{#1}%
5467     \ifx\markdownConTeXtCSV@arg\markdownConTeXtCSV
5468         \placetable[] [tab:#1]{#4}{%
5469             \processeparatedfile[MarkdownConTeXtCSV][#3]}%
5470     \else
5471     \markdownInput{#3}%
5472     \fi}%
5473 \def\markdownRendererImagePrototype#1#2#3#4{%
5474     \placefigure[] [fig:#1]{#4}{\externalfigure[#3]}%
5475 \def\markdownRendererULBeginPrototype{\startitemize}%
5476 \def\markdownRendererULBeginTightPrototype{\startitemize[packed]}%
```

```

5477 \def\markdownRendererUListItemPrototype{\item}%
5478 \def\markdownRendererUListEndPrototype{\stopitemize}%
5479 \def\markdownRendererUListEndTightPrototype{\stopitemize}%
5480 \def\markdownRendererOListBeginPrototype{\startitemize[n]}%
5481 \def\markdownRendererOListBeginTightPrototype{\startitemize[packed,n]}%
5482 \def\markdownRendererOListItemPrototype{\item}%
5483 \def\markdownRendererOListItemWithNumberPrototype#1{\sym{#1.}}%
5484 \def\markdownRendererOListEndPrototype{\stopitemize}%
5485 \def\markdownRendererOListEndTightPrototype{\stopitemize}%
5486 \definedescription
5487   [MarkdownConTeXtDListItemPrototype]
5488   [location=hanging,
5489    margin=standard,
5490    headstyle=bold]%
5491 \definestartstop
5492   [MarkdownConTeXtDListPrototype]
5493   [before=\blank,
5494    after=\blank]%
5495 \definestartstop
5496   [MarkdownConTeXtDListTightPrototype]
5497   [before=\blank\startpacked,
5498    after=\stoppacked\blank]%
5499 \def\markdownRendererDListBeginPrototype{%
5500   \startMarkdownConTeXtDListPrototype}%
5501 \def\markdownRendererDListBeginTightPrototype{%
5502   \startMarkdownConTeXtDListTightPrototype}%
5503 \def\markdownRendererDListItemPrototype#1{%
5504   \startMarkdownConTeXtDListItemPrototype{#1}}%
5505 \def\markdownRendererDListItemEndPrototype{%
5506   \stopMarkdownConTeXtDListItemPrototype}%
5507 \def\markdownRendererDListEndPrototype{%
5508   \stopMarkdownConTeXtDListPrototype}%
5509 \def\markdownRendererDListEndTightPrototype{%
5510   \stopMarkdownConTeXtDListTightPrototype}%
5511 \def\markdownRendererEmphasisPrototype#1{{\em#1}}%
5512 \def\markdownRendererStrongEmphasisPrototype#1{{\bf#1}}%
5513 \def\markdownRendererBlockQuoteBeginPrototype{\startquotation}%
5514 \def\markdownRendererBlockQuoteEndPrototype{\stopquotation}%
5515 \def\markdownRendererInputVerbatimPrototype#1{\typefile{#1}}%
5516 \def\markdownRendererInputFencedCodePrototype#1#2{%
5517   \ifx\relax#2\relax
5518     \typefile{#1}%
5519   \else

```

The code fence infostring is used as a name from the ConTeXt `\definetyping` macro. This allows the user to set up code highlighting mapping as follows:

```

% Map the `TEX` syntax highlighter to the `latex` infostring.

```

```

\definetying [latex]
\setuptyping [latex] [option=TEX]

\starttext
  \startmarkdown
  ~~~ latex
\documentclass{article}
\begin{document}
  Hello world!
\end{document}
  ~~~
  \stopmarkdown
\stoptext

```

```

5520   \typefile[#2] []{#1}%
5521   \fi}%
5522   \def\markdownRendererHeadingOnePrototype#1{\chapter{#1}}%
5523   \def\markdownRendererHeadingTwoPrototype#1{\section{#1}}%
5524   \def\markdownRendererHeadingThreePrototype#1{\subsection{#1}}%
5525   \def\markdownRendererHeadingFourPrototype#1{\subsubsection{#1}}%
5526   \def\markdownRendererHeadingFivePrototype#1{\subsubsubsection{#1}}%
5527   \def\markdownRendererHeadingSixPrototype#1{\subsubsubsubsection{#1}}%
5528   \def\markdownRendererHorizontalRulePrototype{%
5529     \blackrule[height=1pt, width=\hsize]}%
5530   \def\markdownRendererFootnotePrototype#1{\footnote{#1}}%
5531   \stopmodule\protect

  There is a basic implementation of tables.

5532   \newcount\markdownConTeXtRowCounter
5533   \newcount\markdownConTeXtRowTotal
5534   \newcount\markdownConTeXtColumnCounter
5535   \newcount\markdownConTeXtColumnTotal
5536   \newtoks\markdownConTeXtTable
5537   \newtoks\markdownConTeXtTableFloat
5538   \def\markdownRendererTablePrototype#1#2#3{%
5539     \markdownConTeXtTable={}%
5540     \ifx\empty#1\empty
5541       \markdownConTeXtTableFloat={%
5542         \the\markdownConTeXtTable}%
5543     \else
5544       \markdownConTeXtTableFloat={%
5545         \placetable{#1}{\the\markdownConTeXtTable}}%
5546     \fi
5547     \begingroup
5548     \setupTABLE[r][each][topframe=off, bottomframe=off, leftframe=off, rightframe=off]

```



```

5549 \setupTABLE[c][each][topframe=off, bottomframe=off, leftframe=off, rightframe=off]
5550 \setupTABLE[r][1][topframe=on, bottomframe=on]
5551 \setupTABLE[r][#1][bottomframe=on]
5552 \markdownConTeXtRowCounter=0%
5553 \markdownConTeXtRowTotal=#2%
5554 \markdownConTeXtColumnTotal=#3%
5555 \markdownConTeXtRenderTableRow}
5556 \def\markdownConTeXtRenderTableRow#1{%
5557 \markdownConTeXtColumnCounter=0%
5558 \ifnum\markdownConTeXtRowCounter=0\relax
5559 \markdownConTeXtReadAlignments#1%
5560 \markdownConTeXtTable={\bTABLE}%
5561 \else
5562 \markdownConTeXtTable=\expandafter{%
5563 \the\markdownConTeXtTable\bTR}%
5564 \markdownConTeXtRenderTableCell#1%
5565 \markdownConTeXtTable=\expandafter{%
5566 \the\markdownConTeXtTable\eTR}%
5567 \fi
5568 \advance\markdownConTeXtRowCounter by 1\relax
5569 \ifnum\markdownConTeXtRowCounter>\markdownConTeXtRowTotal\relax
5570 \markdownConTeXtTable=\expandafter{%
5571 \the\markdownConTeXtTable\eTABLE}%
5572 \the\markdownConTeXtTableFloat
5573 \endgroup
5574 \expandafter\gobbleoneargument
5575 \fi\markdownConTeXtRenderTableRow}
5576 \def\markdownConTeXtReadAlignments#1{%
5577 \advance\markdownConTeXtColumnCounter by 1\relax
5578 \if#1d%
5579 \setupTABLE[c][\the\markdownConTeXtColumnCounter][align=right]
5580 \fi\if#1l%
5581 \setupTABLE[c][\the\markdownConTeXtColumnCounter][align=right]
5582 \fi\if#1c%
5583 \setupTABLE[c][\the\markdownConTeXtColumnCounter][align=middle]
5584 \fi\if#1r%
5585 \setupTABLE[c][\the\markdownConTeXtColumnCounter][align=left]
5586 \fi
5587 \ifnum\markdownConTeXtColumnCounter<\markdownConTeXtColumnTotal\relax\else
5588 \expandafter\gobbleoneargument
5589 \fi\markdownConTeXtReadAlignments}
5590 \def\markdownConTeXtRenderTableCell#1{%
5591 \advance\markdownConTeXtColumnCounter by 1\relax
5592 \markdownConTeXtTable=\expandafter{%
5593 \the\markdownConTeXtTable\bTD#1\eTD}%
5594 \ifnum\markdownConTeXtColumnCounter<\markdownConTeXtColumnTotal\relax\else
5595 \expandafter\gobbleoneargument

```

References

- [1] Vít Novotný. *TeXový interpret jazyka Markdown (markdown.sty)*. 2015. URL: <https://www.muni.cz/en/research/projects/32984> (visited on 02/19/2018).
- [2] LuaTeX development team. *LuaTeX reference manual*. Feb. 2017. URL: <http://www.luatex.org/svn/trunk/manual/luatex.pdf> (visited on 01/08/2018).
- [3] Anton Sotkov. *File transclusion syntax for Markdown*. Jan. 19, 2017. URL: <https://github.com/iainc/Markdown-Content-Blocks> (visited on 01/08/2018).
- [4] Donald Ervin Knuth. *The TeXbook*. 3rd ed. Addison-Wesley, 1986. ix, 479. ISBN: 0-201-13447-0.
- [5] Frank Mittelbach. *The doc and shortvrb Packages*. Apr. 15, 2017. URL: <http://mirrors.ctan.org/macros/latex/base/doc.pdf> (visited on 02/19/2018).
- [6] Roberto Ierusalimschy. *Programming in Lua*. 3rd ed. Rio de Janeiro: PUC-Rio, 2013. xviii, 347. ISBN: 978-85-903798-5-0.
- [7] Johannes Braams et al. *The L^AT_EX_{2 ϵ} Sources*. Apr. 15, 2017. URL: <http://mirrors.ctan.org/macros/latex/base/source2e.pdf> (visited on 01/08/2018).