A Markdown Interpreter for TEX

Vít Novotný
witiko@mail.muni.cz

Version 2.10.0
August 6, 2021

Contents

1 Introduction

The Markdown package[1] converts markdown[2] markup to TEX commands. The functionality is provided both as a Lua module and as plain TEX, LATEX, and ConTEXt macro packages that can be used to directly typeset TEXdocuments containing markdown markup. Unlike other convertors, 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]

```
local metadata = {
    version   = "2.10.0",
    comment   = "A module for the conversion from markdown to plain TeX",
    author    = "John MacFarlane, Hans Hagen, Vít Novotný",
    copyright = {"2009-2016 John MacFarlane, Hans Hagen",
                 "2016-2021 Vít Novotný"},
    license   = "LPPL 1.3"
}

```

[1] See https://ctan.org/pkg/markdown.
[2] See https://daringfireball.net/projects/markdown/basics.
[3] See 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 GitHub[4] 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 TEX-LATEX 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 TEX implementation of the package draws inspiration from several sources including the source code of LATEX 2_ε, the minted package by Geoffrey M. Poore, which likewise tackles the issue of interfacing with an external interpreter from TEX, 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 LuaTEX 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 LuaTEX $\geq$ 0.72.0 (TEXLive $\geq$ 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 LuaTEX (TEXLive $\geq$ 2008).

[4] See https://github.com/witiko/markdown/issues.
[5] See https://tex.stackexchange.com.

```
local ran_ok, unicode = pcall(require, "unicode")
```

If the Selene Unicode library is unavailable and we are using Lua $\geq$ 5.3, we will use the built-in support for Unicode.

```
if not ran_ok then
  unicode = {["utf8"]={char=utf8.char}}
end
```

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).

```
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 LaTeX Requirements

The LaTeX part of the package requires that the LaTeX 2_ε format is loaded,

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

a TeX engine that extends ε-TeX, all the plain TeX prerequisites (see Section 1.3.2), and the following LaTeX 2ε 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*` LaTeX environment.

```
\RequirePackage{keyval}
```

xstring A package that provides useful macros for manipulating strings of tokens.

```
\RequirePackage{xstring}
```

The following packages are soft prerequisites and will not be loaded if the `plain` package option has been enabled (see Section 2.3.2.1):

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.

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).

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, in the `witiko/dot` LaTeX theme (see Section 2.3.2.2), and to provide default token renderer prototypes (see Section 2.2.4).

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).

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).

gobble A package that provides the `\@gobblethree` TeX command that is used in the default renderer prototype for citations (see Section 2.2.4).

1.3.4 ConTEXt Prerequisites

The ConTEXt 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 ConTEXt 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.

```
21 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.

```
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 TEX 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.

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

`frozenCacheFileName`=⟨*path*⟩ default: `frozenCache.tex`

A path to an output file (frozen cache) that will be created when the `finalizeCache` option is enabled and will contain a mapping between an enumeration of markdown documents and their auxiliary cache files.

The frozen cache makes it possible to later typeset a plain TEX document that contains markdown documents without invoking Lua using the `\markdownOptionFrozenCache` plain TEX option. As a result, the plain TEX document becomes more portable, but further changes in the order and the content of markdown documents will not be reflected.

```
defaultOptions.frozenCacheFileName = "frozenCache.tex"
```

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.

```
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.

```
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.

```
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.

```
defaultOptions.breakableBlockquotes = false
```

`citationNbsps`=`true`, `false` default: `false`

`true` Replace regular spaces with non-breaking spaces inside the prenotes and postnotes of citations produced via the pandoc citation syntax extension.

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

```
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.

```
30 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.''
```

```
31 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.

```
32 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.

```
33 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.

```
34 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
  <pre>
    <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.

```
35 defaultOptions.fencedCode = false
```

`finalizeCache`=`true`, `false` default: `false`

Whether an output file specified with the `frozenCacheFileName` option (frozen cache) that contains a mapping between an enumeration of markdown documents and their auxiliary cache files will be created.

The frozen cache makes it possible to later typeset a plain TEX document that contains markdown documents without invoking Lua using the `\markdownOptionFrozenCache` plain TEX option. As a result, the plain TEX document becomes more portable, but further changes in the order and the content of markdown documents will not be reflected.

```
36 defaultOptions.finalizeCache = 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
```

`frozenCacheCounter`=⟨*number*⟩ default: `0`

The number of the current markdown document that will be stored in an output file (frozen cache) when the `finalizeCache` is enabled. When the document number is 0, then a new frozen cache will be created. Otherwise, the frozen cache will be appended.

Each frozen cache entry will define a TEX macro `\markdownFrozenCache`⟨*number*⟩ that will typeset markdown document number ⟨*number*⟩.

```
38 defaultOptions.frozenCacheCounter = 0
```

`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.

```
39 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.

```
40 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.

```
41 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 interpretted by TEX. This is encouraged when typesetting automatically generated content or markdown documents that were not prepared with this package in mind.

```
42 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.

```
43 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.

```
44 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.

```
45 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.

```
46 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.

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

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

`true`	Convert any ellipses in the input to the `\markdownRendererEllipsis` TEX macro.
`false`	Preserve all ellipses in the input.

```
48 defaultOptions.smartEllipses = false
```

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

`true`	Make the number in the first item of an ordered lists significant. The item numbers will be passed to the `\markdownRendererOlItemWithNumber` TEX macro.
`false`	Ignore the numbers in the ordered list items. Each item will only produce a `\markdownRendererOlItem` TEX macro.

```
49 defaultOptions.startNumber = true
```

`stripIndent`=true, false default: `false`

`true` Strip the minimal indentation of non-blank lines from all lines in a markdown document. Requires that the `preserveTabs` Lua option is `false`:

```
\documentclass{article}
\usepackage[stripIndent]{markdown}
\begin{document}
    \begin{markdown}
        Hello *world*!
    \end{markdown}
\end{document}
```

`false` Do not strip any indentation from the lines in a markdown document.

```
50 defaultOptions.stripIndent = false
```

`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.

```
51 defaultOptions.tableCaptions = false
```

`texComments`=true, false default: `false`

`true` Strip TEX-style comments.

```
\documentclass{article}
\usepackage[texComments]{markdown}
\begin{document}
\begin{markdown}
Hel% this is a comment
```

```
   lo *world*!
\end{markdown}
\end{document}
```

`false` Do not strip TEX-style comments.

```
52 defaultOptions.texComments = false
```

`tightLists`=`true`, `false` default: `true`

`true` Lists whose bullets do not consist of multiple paragraphs will be passed to the `\markdownRendererOlBeginTight`, `\markdownRendererOlEndTight`, `\markdownRendererUlBeginTight`, `\markdownRendererUlEndTight`, `\markdownRendererDlBeginTight`, and `\markdownRendererDlEndTight` 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.

```
53 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.

```
54 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.

```
55
56 HELP_STRING = [[
57 Usage: texlua ]] .. arg[0] .. [[ [OPTIONS] -- [INPUT_FILE] [OUTPUT_FILE]
```

```
where OPTIONS are documented in the Lua interface section of the
technical Markdown package documentation.

When OUTPUT_FILE is unspecified, the result of the conversion will be
written to the standard output. When INPUT_FILE is also unspecified, the
result of the conversion will be read from the standard input.

Report bugs to: witiko@mail.muni.cz
Markdown package home page: <https://github.com/witiko/markdown>]]

VERSION_STRING = [[
markdown-cli.lua (Markdown) ]] .. metadata.version .. [[

Copyright (C) ]] .. table.concat(metadata.copyright,
                                 "\nCopyright (C) ") .. [[

License: ]] .. metadata.license

local function warn(s)
  io.stderr:write("Warning: " .. s .. "\n") end

local function error(s)
  io.stderr:write("Error: " .. s .. "\n")
  os.exit(1) end

local process_options = true
local options = {}
local input_filename
local output_filename
for i = 1, #arg do
  if process_options then
```

After the optional `--` argument has been specified, the remaining arguments are assumed to be input and output filenames. This argument is optional, but encouraged, because it helps resolve ambiguities when deciding whether an option or a filename has been specified.

```
    if arg[i] == "--" then
      process_options = false
      goto continue
```

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

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

The `defaultOptions` table is consulted to identify whether ⟨*value*⟩ should be parsed as a string or as a boolean.

```
      default_type = type(defaultOptions[key])
      if default_type == "boolean" then
        options[key] = (value == "true")
      elseif default_type == "number" then
        options[key] = tonumber(value)
      else
        if default_type ~= "string" then
          if default_type == "nil" then
            warn('Option "' .. key .. '" not recognized.')
          else
            warn('Option "' .. key .. '" type not recognized, please file ' ..
                 'a report to the package maintainer.')
          end
          warn('Parsing the ' .. 'value "' .. value ..'" of option "' ..
               key .. '" as a string.')
        end
        options[key] = value
      end
      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.

```
    elseif arg[i] == "--help" or arg[i] == "-h" then
      print(HELP_STRING)
      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.

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

The first argument that matches none of the above patters 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.

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

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

```
  elseif output_filename == nil then
    output_filename = arg[i]
  else
```

```
126     error('Unexpected argument: "' .. arg[i] .. '".')
127   end
128   ::continue::
129 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.

```
130 \def\markdownLastModified{2021/08/06}%
131 \def\markdownVersion{2.10.0}%
```

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 `\input`ting 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.

```
132 \let\markdownBegin\relax
133 \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 corrolary, 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 TEX [4, p. 46]. As a corrolary, 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 TEX 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 TEX 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 TEX.

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

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

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

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

2.2.2 Options

The plain TEX options are represented by TEX 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 TEX interface.

2.2.2.1 Finalizing and Freezing the Cache

The `\markdownOptionFinalizeCache` option corresponds to the Lua interface `finalizeCache` option, which creates an output file `\markdownOptionFrozenCacheFileName` (frozen cache) that contains a mapping between an enumeration of the markdown documents in the plain TEX document and their auxiliary files cached in the `cacheDir` directory.

```
\let\markdownOptionFinalizeCache\undefined
```

The `\markdownOptionFrozenCache` option uses the mapping previously created by the `\markdownOptionFinalizeCache` option, and uses it to typeset the plain TEX document without invoking Lua. As a result, the plain TEX document becomes more portable, but further changes in the order and the content of markdown documents will not be reflected. It defaults to `false`.

The standard usage of the above two options is as follows:

1. Remove the `cacheDir` cache directory with stale auxiliary cache files.
2. Enable the `\markdownOptionFinalizeCache` option.
4. Typeset the plain TEX document to populate and finalize the cache.
5. Enable the `\markdownOptionFrozenCache` option.
6. Publish the source code of the plain TEX document and the `cacheDir` directory.

2.2.2.2 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 TEX in TEX 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 TEX engines tend to put quotation marks around `\jobname`, when it contains spaces.

```
\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 TEX in `\markdownMode` other than `2`. It defaults to `\jobname.markdown.out`. The same

limitations as in the case of the \markdownOptionHelperScriptFileName macro apply here.

```
\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 TEX in \markdownMode other than 2. It defaults to \jobname.markdown.out. The same limitations apply here as in the case of the \markdownOptionHelperScriptFileName macro.

```
\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.

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

The \markdownOptionOutputDir macro sets the path to the directory that will contain the auxiliary 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.

```
\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 LATEX package. The same limitations apply here as in the case of the \markdownOptionHelperScriptFileName macro.

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

The \markdownOptionFrozenCacheFileName macro corresponds to the Lua interface frozenCacheFileName option that sets the path to an output file (frozen cache) that will contain a mapping between an enumeration of the markdown documents in the plain TEX document and their auxiliary cache files. The option defaults to frozenCache.tex. The same limitations apply here as in the case of the \markdownOptionHelperScriptFileName macro.

```
\def\markdownOptionFrozenCacheFileName{\markdownOptionCacheDir/frozenCache.tex}
```

2.2.2.3 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.

```
\let\markdownOptionBlankBeforeBlockquote\undefined
\let\markdownOptionBlankBeforeCodeFence\undefined
\let\markdownOptionBlankBeforeHeading\undefined
\let\markdownOptionBreakableBlockquotes\undefined
\let\markdownOptionCitations\undefined
\let\markdownOptionCitationNbsps\undefined
\let\markdownOptionContentBlocks\undefined
\let\markdownOptionContentBlocksLanguageMap\undefined
\let\markdownOptionDefinitionLists\undefined
\let\markdownOptionFootnotes\undefined
\let\markdownOptionFencedCode\undefined
\let\markdownOptionHashEnumerators\undefined
\let\markdownOptionHeaderAttributes\undefined
\let\markdownOptionHtml\undefined
\let\markdownOptionHybrid\undefined
\let\markdownOptionInlineFootnotes\undefined
\let\markdownOptionPipeTables\undefined
\let\markdownOptionPreserveTabs\undefined
\let\markdownOptionShiftHeadings\undefined
\let\markdownOptionSlice\undefined
\let\markdownOptionSmartEllipses\undefined
\let\markdownOptionStartNumber\undefined
\let\markdownOptionStripIndent\undefined
\let\markdownOptionTableCaptions\undefined
\let\markdownOptionTeXComments\undefined
\let\markdownOptionTightLists\undefined
```

2.2.2.4 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 3.2.5) or not. Notably, this enables the use of markdown when writing TEX package documentation using the Doc LATEX package [5] or similar. The recognized values of the macro are `true` (discard) and `false` (retain). It defaults to `false`.

```
\def\markdownOptionStripPercentSigns{false}%
```

2.2.3 Token Renderers

The following TEX 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.

```
\def\markdownRendererInterblockSeparator{%
  \markdownRendererInterblockSeparatorPrototype}%
```

2.2.3.2 Line Break Renderer

The `\markdownRendererLineBreak` macro represents a forced line break. The macro receives no arguments.

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

2.2.3.3 Ellipsis Renderer

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

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

2.2.3.4 Non-Breaking Space Renderer

The `\markdownRendererNbsp` macro represents a non-breaking space.

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

2.2.3.5 Special Character Renderers

The following macros replace any special plain TEX characters, including the active pipe character (`|`) of ConTEXt, in the input text. These macros will only be produced, when the `hybrid` option is `false`.

```
\def\markdownRendererLeftBrace{%
  \markdownRendererLeftBracePrototype}%
\def\markdownRendererRightBrace{%
  \markdownRendererRightBracePrototype}%
\def\markdownRendererDollarSign{%
  \markdownRendererDollarSignPrototype}%
\def\markdownRendererPercentSign{%
  \markdownRendererPercentSignPrototype}%
\def\markdownRendererAmpersand{%
  \markdownRendererAmpersandPrototype}%
\def\markdownRendererUnderscore{%
  \markdownRendererUnderscorePrototype}%
\def\markdownRendererHash{%
  \markdownRendererHashPrototype}%
\def\markdownRendererCircumflex{%
```

```
  \markdownRendererCircumflexPrototype}%
\def\markdownRendererBackslash{%
  \markdownRendererBackslashPrototype}%
\def\markdownRendererTilde{%
  \markdownRendererTildePrototype}%
\def\markdownRendererPipe{%
  \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.

```
\def\markdownRendererCodeSpan{%
  \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.

```
\def\markdownRendererLink{%
  \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.

```
\def\markdownRendererImage{%
  \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.

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

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

```
\def\markdownRendererContentBlockOnlineImage{%
  \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: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 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 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.

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

2.2.3.10 Bullet List Renderers

The `\markdownRendererUlBegin` 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.

```
\def\markdownRendererUlBegin{%
  \markdownRendererUlBeginPrototype}%
```

The `\markdownRendererUlBeginTight` 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.

```
\def\markdownRendererUlBeginTight{%
  \markdownRendererUlBeginTightPrototype}%
```

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

```
\def\markdownRendererUlItem{%
  \markdownRendererUlItemPrototype}%
```

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

```
\def\markdownRendererUlItemEnd{%
  \markdownRendererUlItemEndPrototype}%
```

The `\markdownRendererUlEnd` 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.

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

```
\def\markdownRendererUlEnd{%
  \markdownRendererUlEndPrototype}%
```

The `\markdownRendererUlEndTight` 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.

```
\def\markdownRendererUlEndTight{%
  \markdownRendererUlEndTightPrototype}%
```

2.2.3.11 Ordered List Renderers

The `\markdownRendererOlBegin` 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.

```
\def\markdownRendererOlBegin{%
  \markdownRendererOlBeginPrototype}%
```

The `\markdownRendererOlBeginTight` 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.

```
\def\markdownRendererOlBeginTight{%
  \markdownRendererOlBeginTightPrototype}%
```

The `\markdownRendererOlItem` 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.

```
\def\markdownRendererOlItem{%
  \markdownRendererOlItemPrototype}%
```

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

```
\def\markdownRendererOlItemEnd{%
  \markdownRendererOlItemEndPrototype}%
```

The `\markdownRendererOlItemWithNumber` macro represents an item in an ordered list. This macro will only be produced, when the `startNumber` option is enabled. The macro receives a single numeric argument that corresponds to the item number.

```
\def\markdownRendererOlItemWithNumber{%
  \markdownRendererOlItemWithNumberPrototype}%
```

The `\markdownRendererOlEnd` 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.

```
\def\markdownRendererOlEnd{%
  \markdownRendererOlEndPrototype}%
```

The `\markdownRendererOlEndTight` 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.

```
\def\markdownRendererOlEndTight{%
  \markdownRendererOlEndTightPrototype}%
```

2.2.3.12 Definition List Renderers

The following macros are only produced, when the `definitionLists` option is enabled.

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.

```
\def\markdownRendererDlBegin{%
  \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.

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

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

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

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

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

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

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

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

```
\def\markdownRendererDlDefinitionEnd{%
  \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.

```
\def\markdownRendererDlEnd{%
  \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.

```
\def\markdownRendererDlEndTight{%
  \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.

```
\def\markdownRendererEmphasis{%
  \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.

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

2.2.3.14 Block Quote Renderers

The \markdownRendererBlockQuoteBegin macro represents the beginning of a block quote. The macro receives no arguments.

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

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

```
\def\markdownRendererBlockQuoteEnd{%
  \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 contaning the code block contents.

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

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

```
\def\markdownRendererInputFencedCode{%
  \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.

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

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

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

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

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

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

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

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

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

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

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

2.2.3.17 Horizontal Rule Renderer

The \markdownRendererHorizontalRule macro represents a horizontal rule. The macro receives no arguments.

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

2.2.3.18 Footnote Renderer

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

```
\def\markdownRendererFootnote{%
  \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 enabled. 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.

```
\def\markdownRendererCite{%
  \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 enabled. The macro receives parameters in the same format as the `\markdownRendererCite` macro.

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

2.2.3.21 Table Renderer

The `\markdownRendererTable` macro represents a table. This macro will only be produced, when the `pipeTables` option is enabled. 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.

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

2.2.3.22 Inline HTML Comment Renderer

The `\markdownRendererInlineHtmlComment` macro represents the contents of an inline HTML comment. This macro will only be

produced, when the `html` option is enabled. The macro receives a single argument that corresponds to the contents of the HTML comment.

```
\def\markdownRendererInlineHtmlComment{%
  \markdownRendererInlineHtmlCommentPrototype}%
```

2.2.4 Token Renderer Prototypes

The following TEX 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 LATEX and ConTEXt implementations (see sections 3.3 and 3.4).

```
\def\markdownRendererInterblockSeparatorPrototype{}%
\def\markdownRendererLineBreakPrototype{}%
\def\markdownRendererEllipsisPrototype{}%
\def\markdownRendererNbspPrototype{}%
\def\markdownRendererLeftBracePrototype{}%
\def\markdownRendererRightBracePrototype{}%
\def\markdownRendererDollarSignPrototype{}%
\def\markdownRendererPercentSignPrototype{}%
\def\markdownRendererAmpersandPrototype{}%
\def\markdownRendererUnderscorePrototype{}%
\def\markdownRendererHashPrototype{}%
\def\markdownRendererCircumflexPrototype{}%
\def\markdownRendererBackslashPrototype{}%
\def\markdownRendererTildePrototype{}%
\def\markdownRendererPipePrototype{}%
\def\markdownRendererCodeSpanPrototype#1{}%
\def\markdownRendererLinkPrototype#1#2#3#4{}%
\def\markdownRendererImagePrototype#1#2#3#4{}%
\def\markdownRendererContentBlockPrototype#1#2#3#4{}%
\def\markdownRendererContentBlockOnlineImagePrototype#1#2#3#4{}%
\def\markdownRendererContentBlockCodePrototype#1#2#3#4#5{}%
\def\markdownRendererUlBeginPrototype{}%
\def\markdownRendererUlBeginTightPrototype{}%
\def\markdownRendererUlItemPrototype{}%
\def\markdownRendererUlItemEndPrototype{}%
\def\markdownRendererUlEndPrototype{}%
\def\markdownRendererUlEndTightPrototype{}%
\def\markdownRendererOlBeginPrototype{}%
\def\markdownRendererOlBeginTightPrototype{}%
\def\markdownRendererOlItemPrototype{}%
\def\markdownRendererOlItemWithNumberPrototype#1{}%
\def\markdownRendererOlItemEndPrototype{}%
\def\markdownRendererOlEndPrototype{}%
\def\markdownRendererOlEndTightPrototype{}%
```

```
\def\markdownRendererDlBeginPrototype{}%
\def\markdownRendererDlBeginTightPrototype{}%
\def\markdownRendererDlItemPrototype#1{}%
\def\markdownRendererDlItemEndPrototype{}%
\def\markdownRendererDlDefinitionBeginPrototype{}%
\def\markdownRendererDlDefinitionEndPrototype{}%
\def\markdownRendererDlEndPrototype{}%
\def\markdownRendererDlEndTightPrototype{}%
\def\markdownRendererEmphasisPrototype#1{}%
\def\markdownRendererStrongEmphasisPrototype#1{}%
\def\markdownRendererBlockQuoteBeginPrototype{}%
\def\markdownRendererBlockQuoteEndPrototype{}%
\def\markdownRendererInputVerbatimPrototype#1{}%
\def\markdownRendererInputFencedCodePrototype#1#2{}%
\def\markdownRendererHeadingOnePrototype#1{}%
\def\markdownRendererHeadingTwoPrototype#1{}%
\def\markdownRendererHeadingThreePrototype#1{}%
\def\markdownRendererHeadingFourPrototype#1{}%
\def\markdownRendererHeadingFivePrototype#1{}%
\def\markdownRendererHeadingSixPrototype#1{}%
\def\markdownRendererHorizontalRulePrototype{}%
\def\markdownRendererFootnotePrototype#1{}%
\def\markdownRendererCitePrototype#1{}%
\def\markdownRendererTextCitePrototype#1{}%
\def\markdownRendererTablePrototype#1#2#3{}%
\def\markdownRendererInlineHtmlCommentPrototype#1{}%
```

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

```
\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.

```
\let\markdownReadAndConvert\relax
\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*.

```
  \catcode`\|=0\catcode`\\=12%
  |gdef|markdownBegin{%
    |markdownReadAndConvert{\markdownEnd}%
                           {|markdownEnd}}%
|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 3.2.7), 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.

```
\ifx\markdownMode\undefined
  \ifx\directlua\undefined
    \def\markdownMode{0}%
  \else
    \def\markdownMode{2}%
  \fi
\fi
```

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

```
\def\markdownLuaRegisterIBCallback#1{\relax}%
\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 2.3.2.5) and `markdownRendererPrototypes` (see Section 2.3.2.6) keys. This limitation is due to the way LATEX 2$_\varepsilon$ 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.

```
\newenvironment{markdown}\relax\relax
\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}
\usepackage{markdown}
\begin{document}
% ...
\begin{markdown}
_Hello_ **world** ...
\end{markdown}
% ...
\end{document}
```

```
\documentclass{article}
\usepackage{markdown}
\begin{document}
% ...
\begin{markdown*}{smartEllipses}
_Hello_ **world** ...
\end{markdown*}
% ...
\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 influnce 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 ⟨*key*⟩=⟨*value*⟩ pairs. For boolean options, the =⟨*value*⟩ part is optional, and ⟨*key*⟩ will be interpreted as ⟨*key*⟩=`true` if the =⟨*value*⟩ part has been omitted.

Except for the `plain` option described in Section 2.3.2.1, and the LATEX themes described in Section 2.3.2.2, and the LATEX setup snippets described in Section 2.3.2.3, 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, when using the `markdown*` LATEX environment or the `\markdownInput` macro (see Section 2.3), or via the `\markdownSetup` macro. The `\markdownSetup` macro receives the options to set up as its only argument:

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

We may also store LATEX options as *setup snippets* and invoke them later using the `\markdownSetupSnippet` macro. The `\markdownSetupSnippet` macro receives two arguments: the name of the setup snippet and the options to store:

```
\newcommand\markdownSetupSnippet[2]{%
  \@ifundefined
    {markdownLaTeXSetupSnippet\markdownLaTeXThemeName#1}{%
      \newtoks\next
        \next={#2}%
      \expandafter\let\csname markdownLaTeXSetupSnippet%
```

```
      \markdownLaTeXThemeName#1\endcsname=\next
  }{%
    \markdownWarning
      {Redefined setup snippet \markdownLaTeXThemeName#1}%
    \csname markdownLaTeXSetupSnippet%
      \markdownLaTeXThemeName#1\endcsname={#2}%
  }}%
```

See Section 2.3.2.2 for information on interactions between setup snippets and LaTeX themes. See Section 2.3.2.3 for information about invoking the stored setup snippets.

2.3.2.1 No default token renderer prototypes Default token renderer prototypes require LaTeX packages that may clash with other packages used in a document. Additionally, if we redefine token renderers and renderer prototypes ourselves, the default definitions will bring no benefit to us. Using the `plain` package option, we can keep the default definitions from the plain TeX implementation (see Section 3.2.3) and prevent the soft LaTeX prerequisites in Section 1.3.3 from being loaded:

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

```
\newif\ifmarkdownLaTeXPlain
  \markdownLaTeXPlainfalse
\define@key{markdownOptions}{plain}[true]{%
  \ifmarkdownLaTeXLoaded
    \markdownWarning
      {The plain option must be specified when loading the package}%
  \else
    \markdownLaTeXPlaintrue
  \fi}
```

2.3.2.2 LaTeX themes

User-contributed LaTeX themes for the Markdown package provide a domain-specific interpretation of some Markdown tokens. Similarly to LaTeX packages, themes allow the authors to achieve a specific look and other high-level goals without low-level programming.

The LaTeX option with key `theme` loads a LaTeX package (further referred to as *a theme*) named `markdowntheme`⟨*munged theme name*⟩`.sty`, where the *munged theme name* is the *theme name* after a substitution of all forward slashes (`/`) for an underscore (`_`), the theme name is a value that is *qualified* and contains no underscores, and a value is qualified if and only if it contains at least one forward slash. Themes are inspired by the Beamer LaTeX package, which provides similar functionality with its `\usetheme` macro [6, Section 15.1].

Theme names must be qualified to minimize naming conflicts between different themes intended for a single LaTeX document class or for a single LaTeX package.

The preferred format of a theme name is ⟨*theme author*⟩`/`⟨*target LaTeX document class or package*⟩`/`⟨*private naming scheme*⟩, where the *private naming scheme* may contain additional forward slashes. For example, a theme by a user `witiko` for the MU theme of the Beamer document class may have the name `witiko/beamer/MU`.

Theme names are munged, because LaTeX packages are identified only by their filenames, not by their pathnames. [7] Therefore, we can't store the qualified theme names directly using directories, but we must encode the individual segments of the qualified theme in the filename. For example, loading a theme named `witiko/beamer/MU` would load a LaTeX package named `markdownthemewitiko_beamer_MU.sty`.

If the LaTeX option with key `theme` is (repeatedly) specified in the `\usepackage` macro, the loading of the theme(s) will be postponed in first-in-first-out order until after the Markdown LaTeX package has been loaded. Otherwise, the theme(s) will be loaded immediately. For example, there is a theme named `witiko/dot`, which typesets fenced code blocks with the `dot` infostring as images of directed graphs rendered by the Graphviz tools. The following code would first load the Markdown package, then the `markdownthemewitiko_beamer_MU.sty` LaTeX package, and finally the `markdownthemewitiko_dot.sty` LaTeX package:

```
\usepackage[
  theme = witiko/beamer/MU,
  theme = witiko/dot,
]{markdown}
```

```
\newif\ifmarkdownLaTeXLoaded
  \markdownLaTeXLoadedfalse
\AtEndOfPackage{\markdownLaTeXLoadedtrue}%
\define@key{markdownOptions}{theme}{%
  \IfSubStr{#1}{/}{}{%
    \markdownError
      {Won't load theme with unqualified name #1}%
      {Theme names must contain at least one forward slash}}%
  \StrSubstitute{#1}{/}{_}[\markdownLaTeXThemePackageName]%
  \edef\markdownLaTeXThemePackageName{%
    markdowntheme\markdownLaTeXThemePackageName}%
  \expandafter\markdownLaTeXThemeLoad\expandafter{%
    \markdownLaTeXThemePackageName}{#1/}}%
\newcommand\markdownLaTeXThemeName{}%
\newcommand\markdownLaTeXThemeLoad[2]{%
  \ifmarkdownLaTeXLoaded
    \def\markdownLaTeXThemeName{#2}%
    \RequirePackage{#1}%
    \def\markdownLaTeXThemeName{}%
  \else
```

```
    \AtEndOfPackage{%
      \def\markdownLaTeXThemeName{#2}%
      \RequirePackage{#1}%
      \def\markdownLaTeXThemeName{}}%
  \fi}%
```

The LaTeX themes have a useful synergy with the setup snippets (see Section 2.3.2): To make it less likely that different themes will define setup snippets with the same name, we will prepend ⟨*theme name*⟩`/` before the snippet name and use the result as the snippet name. For example, if the `witiko/dot` theme defines the `product` setup snippet, the setup snippet will be available under the name `witiko/dot/product`.Due to limitations of LaTeX, themes may not be loaded after the beginning of a LaTeX document.

```
\@onlypreamble\KV@markdownOptions@theme
```

Example themes provided with the Markdown package include:

witiko/dot A theme that typesets fenced code blocks with the `dot …` infostring as images of directed graphs rendered by the Graphviz tools. The right tail of the infostring is used as the image title.

```
\documentclass{article}
\usepackage[theme=witiko/dot]{markdown}
\setkeys{Gin}{
  width = \columnwidth,
  height = 0.65\paperheight,
  keepaspectratio}
\begin{document}
\begin{markdown}
``` dot Various formats of mathemathical formulae
digraph tree {
  margin = 0;
  rankdir = "LR";

  latex -> pmml;
  latex -> cmml;
  pmml -> slt;
  cmml -> opt;
  cmml -> prefix;
  cmml -> infix;
  pmml -> mterms [style=dashed];
  cmml -> mterms;

  latex [label = "LaTeX"];
```
```

```
  pmml [label = "Presentation MathML"];
  cmml [label = "Content MathML"];
  slt [label = "Symbol Layout Tree"];
  opt [label = "Operator Tree"];
  prefix [label = "Prefix"];
  infix [label = "Infix"];
  mterms [label = "M-Terms"];
}
```
\end{markdown}
\end{document}
```

Typesetting the above document produces the output shown in Figure 1.

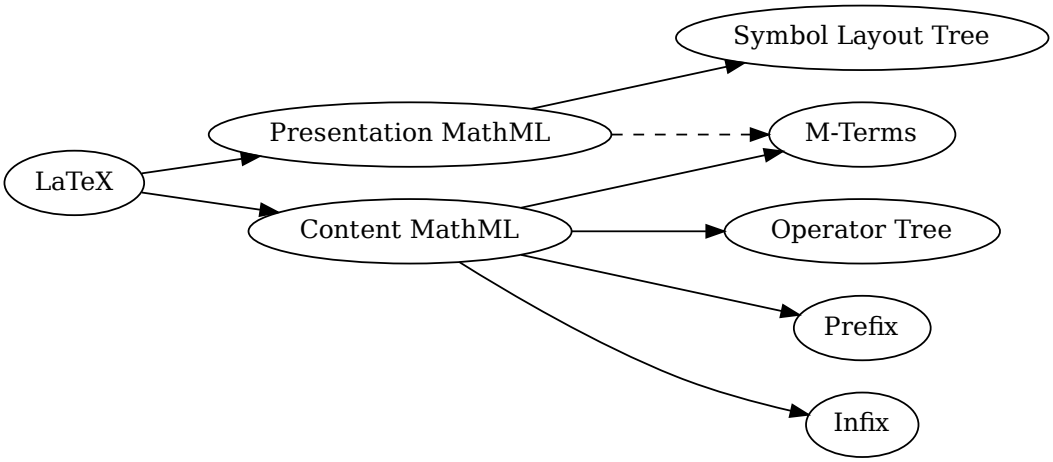


**Figure 1: Various formats of mathemathical formulae**

The theme requires a Unix-like operating system with GNU Diffutils and Graphviz installed. The theme also requires shell access unless the `\markdownOptionFrozenCache` plain TEX option is enabled.

```
\ProvidesPackage{markdownthemewitiko_dot}[2021/03/09]%
```

witiko/graphicx/http A theme that adds support for downloading images whose URL has the http or https protocol.

```
\documentclass{article}
\usepackage[theme=witiko/graphicx/http]{markdown}
\begin{document}
```
```

```
\begin{markdown}
![img](https://github.com/witiko/markdown/raw/master/banner.png
       "The banner of the Markdown package")
\end{markdown}
\end{document}
```

Typesetting the above document produces the output shown in Figure 2. The

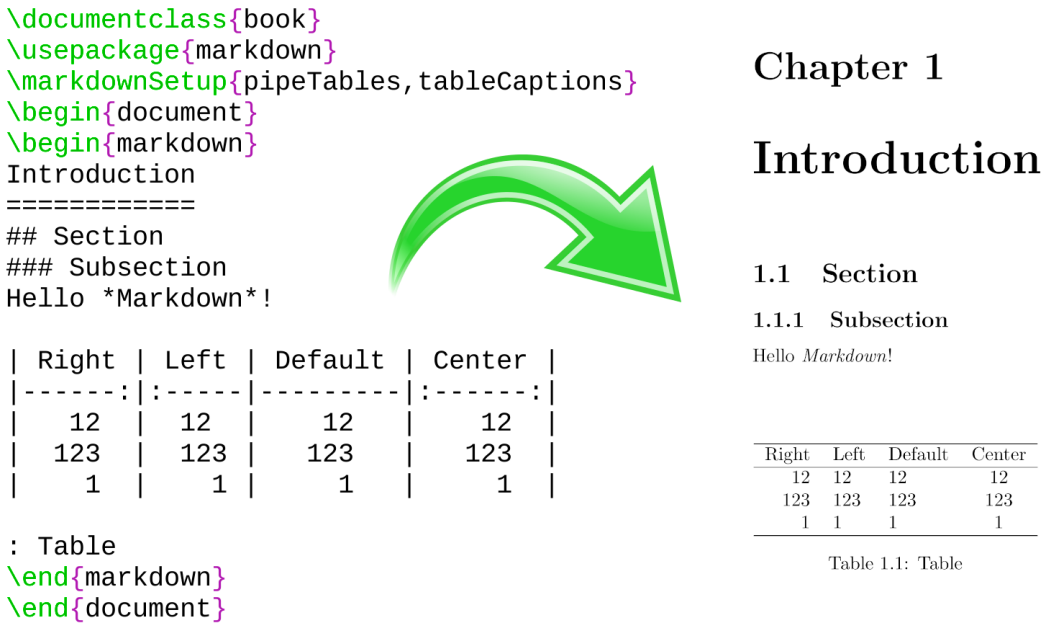

Figure 2: The banner of the Markdown package

theme requires the catchfile LaTeX package and a Unix-like operating system with GNU Coreutils `md5sum` and either GNU Wget or cURL installed. The theme also requires shell access unless the `\markdownOptionFrozenCache` plain TeX option is enabled.

```
\ProvidesPackage{markdownthemewitiko_graphicx_http}[2021/03/22]%
\RequirePackage{catchfile}
```

witiko/tilde A theme that makes tilde (`~`) always typeset the non-breaking space even when the `hybrid` Lua option is `false`.

```
\documentclass{article}
\usepackage[theme=witiko/tilde]{markdown}
\begin{document}
```

```
\begin{markdown}
Bartel~Leendert van~der~Waerden
\end{markdown}
\end{document}
```

Typesetting the above document produces the following text: “Bartel Leendert van der Waerden”.

```
\ProvidesPackage{markdownthemewitiko_tilde}[2021/03/22]%
```

Please, see Section 3.3.3.1 for implementation details of the example themes.

2.3.2.3 LaTeX setup snippets

The LaTeX option with key `snippet` invokes a snippet named ⟨*value*⟩:

```
\define@key{markdownOptions}{snippet}{%
  \expandafter\markdownSetup\expandafter{%
    \the\csname markdownLaTeXSetupSnippet#1\endcsname}}%
```

Here is how we can use setup snippets to store options and invoke them later:

```
\markdownSetupSnippet{romanNumerals}{
  renderers = {
      olItemWithNumber = {%
          \item[\romannumeral#1\relax.]%
      },
  },
}
\begin{markdown}

The following ordered list will be preceded by arabic numerals:

1. wahid
2. aithnayn

\end{markdown}
\begin{markdown*}{snippet=romanNumerals}

The following ordered list will be preceded by roman numerals:

3. tres
4. quattuor

\end{markdown*}
```

2.3.2.4 Plain TEX Interface Options

The following options map directly to the option macros exposed by the plain TEX interface (see Section 2.2.2).

```
\define@key{markdownOptions}{helperScriptFileName}{%
  \def\markdownOptionHelperScriptFileName{#1}}%
\define@key{markdownOptions}{inputTempFileName}{%
  \def\markdownOptionInputTempFileName{#1}}%
\define@key{markdownOptions}{outputTempFileName}{%
  \def\markdownOptionOutputTempFileName{#1}}%
\define@key{markdownOptions}{errorTempFileName}{%
  \def\markdownOptionErrorTempFileName{#1}}%
\define@key{markdownOptions}{cacheDir}{%
  \def\markdownOptionCacheDir{#1}}%
\define@key{markdownOptions}{outputDir}{%
  \def\markdownOptionOutputDir{#1}}%
\define@key{markdownOptions}{blankBeforeBlockquote}[true]{%
  \def\markdownOptionBlankBeforeBlockquote{#1}}%
\define@key{markdownOptions}{blankBeforeCodeFence}[true]{%
  \def\markdownOptionBlankBeforeCodeFence{#1}}%
\define@key{markdownOptions}{blankBeforeHeading}[true]{%
  \def\markdownOptionBlankBeforeHeading{#1}}%
\define@key{markdownOptions}{breakableBlockquotes}[true]{%
  \def\markdownOptionBreakableBlockquotes{#1}}%
\define@key{markdownOptions}{citations}[true]{%
  \def\markdownOptionCitations{#1}}%
\define@key{markdownOptions}{citationNbsps}[true]{%
  \def\markdownOptionCitationNbsps{#1}}%
\define@key{markdownOptions}{contentBlocks}[true]{%
  \def\markdownOptionContentBlocks{#1}}%
\define@key{markdownOptions}{codeSpans}[true]{%
  \def\markdownOptionCodeSpans{#1}}%
\define@key{markdownOptions}{contentBlocksLanguageMap}{%
  \def\markdownOptionContentBlocksLanguageMap{#1}}%
\define@key{markdownOptions}{definitionLists}[true]{%
  \def\markdownOptionDefinitionLists{#1}}%
\define@key{markdownOptions}{footnotes}[true]{%
  \def\markdownOptionFootnotes{#1}}%
\define@key{markdownOptions}{fencedCode}[true]{%
  \def\markdownOptionFencedCode{#1}}%
\define@key{markdownOptions}{hashEnumerators}[true]{%
  \def\markdownOptionHashEnumerators{#1}}%
\define@key{markdownOptions}{headerAttributes}[true]{%
  \def\markdownOptionHeaderAttributes{#1}}%
\define@key{markdownOptions}{html}[true]{%
  \def\markdownOptionHtml{#1}}%
\define@key{markdownOptions}{hybrid}[true]{%
  \def\markdownOptionHybrid{#1}}%
\define@key{markdownOptions}{inlineFootnotes}[true]{%
```

```
  \def\markdownOptionInlineFootnotes{#1}%
\define@key{markdownOptions}{pipeTables}[true]{%
  \def\markdownOptionPipeTables{#1}%
\define@key{markdownOptions}{preserveTabs}[true]{%
  \def\markdownOptionPreserveTabs{#1}%
\define@key{markdownOptions}{smartEllipses}[true]{%
  \def\markdownOptionSmartEllipses{#1}%
\define@key{markdownOptions}{shiftHeadings}{%
  \def\markdownOptionShiftHeadings{#1}%
\define@key{markdownOptions}{slice}{%
  \def\markdownOptionSlice{#1}%
\define@key{markdownOptions}{startNumber}[true]{%
  \def\markdownOptionStartNumber{#1}%
\define@key{markdownOptions}{stripIndent}[true]{%
  \def\markdownOptionStripIndent{#1}%
\define@key{markdownOptions}{tableCaptions}[true]{%
  \def\markdownOptionTableCaptions{#1}%
\define@key{markdownOptions}{texComments}[true]{%
  \def\markdownOptionTeXComments{#1}%
\define@key{markdownOptions}{tightLists}[true]{%
  \def\markdownOptionTightLists{#1}%
\define@key{markdownOptions}{underscores}[true]{%
  \def\markdownOptionUnderscores{#1}%
\define@key{markdownOptions}{stripPercentSigns}[true]{%
  \def\markdownOptionStripPercentSigns{#1}%
```

The `\markdownOptionFinalizeCache` and `\markdownOptionFrozenCache` plain TEX options are exposed through LATEX options with keys `finalizeCache` and `frozenCache`.

To ensure compatibility with the `minted` package [8, Section 5.1], which supports the `finalizecache` and `frozencache` package options with similar semantics, the Markdown package also recognizes these as aliases and recognizes them as document class options. By passing `finalizecache` and `frozencache` as document class options, you may conveniently control the behavior of both packages at once:

```
\documentclass[frozencache]{article}
\usepackage{markdown,minted}
\begin{document}
% ...
\end{document}
```

We hope that other packages will support the `finalizecache` and `frozencache` package options in the future, so that they can become a standard interface for preparing LATEX document sources for distribution.

```
\define@key{markdownOptions}{finalizeCache}[true]{%
```

```
497   \def\markdownOptionFinalizeCache{#1}%
498 \DeclareOption{finalizecache}{\markdownSetup{finalizeCache}}
499 \define@key{markdownOptions}{frozenCache}[true]{%
500   \def\markdownOptionFrozenCache{#1}%
501 \DeclareOption{frozencache}{\markdownSetup{frozenCache}}
502 \define@key{markdownOptions}{frozenCacheFileName}{%
503   \def\markdownOptionFrozenCacheFileName{#1}%
```

The following example LaTeX code showcases a possible configuration of plain TeX interface options `\markdownOptionHybrid`, `\markdownOptionSmartEllipses`, and `\markdownOptionCacheDir`.

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

2.3.2.5 Plain TeX Markdown Token Renderers

The LaTeX 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 TeX interface (see Section 2.2.3).

```
504 \define@key{markdownRenderers}{interblockSeparator}{%
505   \renewcommand\markdownRendererInterblockSeparator{#1}}%
506 \define@key{markdownRenderers}{lineBreak}{%
507   \renewcommand\markdownRendererLineBreak{#1}}%
508 \define@key{markdownRenderers}{ellipsis}{%
509   \renewcommand\markdownRendererEllipsis{#1}}%
510 \define@key{markdownRenderers}{nbsp}{%
511   \renewcommand\markdownRendererNbsp{#1}}%
512 \define@key{markdownRenderers}{leftBrace}{%
513   \renewcommand\markdownRendererLeftBrace{#1}}%
514 \define@key{markdownRenderers}{rightBrace}{%
515   \renewcommand\markdownRendererRightBrace{#1}}%
516 \define@key{markdownRenderers}{dollarSign}{%
517   \renewcommand\markdownRendererDollarSign{#1}}%
518 \define@key{markdownRenderers}{percentSign}{%
519   \renewcommand\markdownRendererPercentSign{#1}}%
520 \define@key{markdownRenderers}{ampersand}{%
521   \renewcommand\markdownRendererAmpersand{#1}}%
522 \define@key{markdownRenderers}{underscore}{%
523   \renewcommand\markdownRendererUnderscore{#1}}%
524 \define@key{markdownRenderers}{hash}{%
525   \renewcommand\markdownRendererHash{#1}}%
526 \define@key{markdownRenderers}{circumflex}{%
```

```
  \renewcommand\markdownRendererCircumflex{#1}%
\define@key{markdownRenderers}{backslash}{%
  \renewcommand\markdownRendererBackslash{#1}%
\define@key{markdownRenderers}{tilde}{%
  \renewcommand\markdownRendererTilde{#1}%
\define@key{markdownRenderers}{pipe}{%
  \renewcommand\markdownRendererPipe{#1}%
\define@key{markdownRenderers}{codeSpan}{%
  \renewcommand\markdownRendererCodeSpan[1]{#1}%
\define@key{markdownRenderers}{link}{%
  \renewcommand\markdownRendererLink[4]{#1}%
\define@key{markdownRenderers}{contentBlock}{%
  \renewcommand\markdownRendererContentBlock[4]{#1}%
\define@key{markdownRenderers}{contentBlockOnlineImage}{%
  \renewcommand\markdownRendererContentBlockOnlineImage[4]{#1}%
\define@key{markdownRenderers}{contentBlockCode}{%
  \renewcommand\markdownRendererContentBlockCode[5]{#1}%
\define@key{markdownRenderers}{image}{%
  \renewcommand\markdownRendererImage[4]{#1}%
\define@key{markdownRenderers}{ulBegin}{%
  \renewcommand\markdownRendererUlBegin{#1}%
\define@key{markdownRenderers}{ulBeginTight}{%
  \renewcommand\markdownRendererUlBeginTight{#1}%
\define@key{markdownRenderers}{ulItem}{%
  \renewcommand\markdownRendererUlItem{#1}%
\define@key{markdownRenderers}{ulItemEnd}{%
  \renewcommand\markdownRendererUlItemEnd{#1}%
\define@key{markdownRenderers}{ulEnd}{%
  \renewcommand\markdownRendererUlEnd{#1}%
\define@key{markdownRenderers}{ulEndTight}{%
  \renewcommand\markdownRendererUlEndTight{#1}%
\define@key{markdownRenderers}{olBegin}{%
  \renewcommand\markdownRendererOlBegin{#1}%
\define@key{markdownRenderers}{olBeginTight}{%
  \renewcommand\markdownRendererOlBeginTight{#1}%
\define@key{markdownRenderers}{olItem}{%
  \renewcommand\markdownRendererOlItem{#1}%
\define@key{markdownRenderers}{olItemWithNumber}{%
  \renewcommand\markdownRendererOlItemWithNumber[1]{#1}%
\define@key{markdownRenderers}{olItemEnd}{%
  \renewcommand\markdownRendererOlItemEnd{#1}%
\define@key{markdownRenderers}{olEnd}{%
  \renewcommand\markdownRendererOlEnd{#1}%
\define@key{markdownRenderers}{olEndTight}{%
  \renewcommand\markdownRendererOlEndTight{#1}%
\define@key{markdownRenderers}{dlBegin}{%
  \renewcommand\markdownRendererDlBegin{#1}%
```

```
\define@key{markdownRenderers}{dlBeginTight}{%
  \renewcommand\markdownRendererDlBeginTight{#1}}%
\define@key{markdownRenderers}{dlItem}{%
  \renewcommand\markdownRendererDlItem[1]{#1}}%
\define@key{markdownRenderers}{dlItemEnd}{%
  \renewcommand\markdownRendererDlItemEnd{#1}}%
\define@key{markdownRenderers}{dlDefinitionBegin}{%
  \renewcommand\markdownRendererDlDefinitionBegin{#1}}%
\define@key{markdownRenderers}{dlDefinitionEnd}{%
  \renewcommand\markdownRendererDlDefinitionEnd{#1}}%
\define@key{markdownRenderers}{dlEnd}{%
  \renewcommand\markdownRendererDlEnd{#1}}%
\define@key{markdownRenderers}{dlEndTight}{%
  \renewcommand\markdownRendererDlEndTight{#1}}%
\define@key{markdownRenderers}{emphasis}{%
  \renewcommand\markdownRendererEmphasis[1]{#1}}%
\define@key{markdownRenderers}{strongEmphasis}{%
  \renewcommand\markdownRendererStrongEmphasis[1]{#1}}%
\define@key{markdownRenderers}{blockQuoteBegin}{%
  \renewcommand\markdownRendererBlockQuoteBegin{#1}}%
\define@key{markdownRenderers}{blockQuoteEnd}{%
  \renewcommand\markdownRendererBlockQuoteEnd{#1}}%
\define@key{markdownRenderers}{inputVerbatim}{%
  \renewcommand\markdownRendererInputVerbatim[1]{#1}}%
\define@key{markdownRenderers}{inputFencedCode}{%
  \renewcommand\markdownRendererInputFencedCode[2]{#1}}%
\define@key{markdownRenderers}{headingOne}{%
  \renewcommand\markdownRendererHeadingOne[1]{#1}}%
\define@key{markdownRenderers}{headingTwo}{%
  \renewcommand\markdownRendererHeadingTwo[1]{#1}}%
\define@key{markdownRenderers}{headingThree}{%
  \renewcommand\markdownRendererHeadingThree[1]{#1}}%
\define@key{markdownRenderers}{headingFour}{%
  \renewcommand\markdownRendererHeadingFour[1]{#1}}%
\define@key{markdownRenderers}{headingFive}{%
  \renewcommand\markdownRendererHeadingFive[1]{#1}}%
\define@key{markdownRenderers}{headingSix}{%
  \renewcommand\markdownRendererHeadingSix[1]{#1}}%
\define@key{markdownRenderers}{horizontalRule}{%
  \renewcommand\markdownRendererHorizontalRule{#1}}%
\define@key{markdownRenderers}{footnote}{%
  \renewcommand\markdownRendererFootnote[1]{#1}}%
\define@key{markdownRenderers}{cite}{%
  \renewcommand\markdownRendererCite[1]{#1}}%
\define@key{markdownRenderers}{textCite}{%
  \renewcommand\markdownRendererTextCite[1]{#1}}%
\define@key{markdownRenderers}{table}{%
```

```
  \renewcommand\markdownRendererTable[3]{#1}%
\define@key{markdownRenderers}{inlineHtmlComment}{%
  \renewcommand\markdownRendererInlineHtmlComment[1]{#1}%
```

The following example LaTeX 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`.
  }
}
```

2.3.2.6 Plain TeX Markdown Token Renderer Prototypes

The LaTeX 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 TeX interface (see Section 2.2.4).

```
\define@key{markdownRendererPrototypes}{interblockSeparator}{%
  \renewcommand\markdownRendererInterblockSeparatorPrototype{#1}}%
\define@key{markdownRendererPrototypes}{lineBreak}{%
  \renewcommand\markdownRendererLineBreakPrototype{#1}}%
\define@key{markdownRendererPrototypes}{ellipsis}{%
  \renewcommand\markdownRendererEllipsisPrototype{#1}}%
\define@key{markdownRendererPrototypes}{nbsp}{%
  \renewcommand\markdownRendererNbspPrototype{#1}}%
\define@key{markdownRendererPrototypes}{leftBrace}{%
  \renewcommand\markdownRendererLeftBracePrototype{#1}}%
\define@key{markdownRendererPrototypes}{rightBrace}{%
  \renewcommand\markdownRendererRightBracePrototype{#1}}%
\define@key{markdownRendererPrototypes}{dollarSign}{%
  \renewcommand\markdownRendererDollarSignPrototype{#1}}%
\define@key{markdownRendererPrototypes}{percentSign}{%
  \renewcommand\markdownRendererPercentSignPrototype{#1}}%
\define@key{markdownRendererPrototypes}{ampersand}{%
  \renewcommand\markdownRendererAmpersandPrototype{#1}}%
\define@key{markdownRendererPrototypes}{underscore}{%
  \renewcommand\markdownRendererUnderscorePrototype{#1}}%
\define@key{markdownRendererPrototypes}{hash}{%
  \renewcommand\markdownRendererHashPrototype{#1}}%
\define@key{markdownRendererPrototypes}{circumflex}{%
  \renewcommand\markdownRendererCircumflexPrototype{#1}}%
\define@key{markdownRendererPrototypes}{backslash}{%
  \renewcommand\markdownRendererBackslashPrototype{#1}}%
```

```
\define@key{markdownRendererPrototypes}{tilde}{%
  \renewcommand\markdownRendererTildePrototype{#1}}%
\define@key{markdownRendererPrototypes}{pipe}{%
  \renewcommand\markdownRendererPipePrototype{#1}}%
\define@key{markdownRendererPrototypes}{codeSpan}{%
  \renewcommand\markdownRendererCodeSpanPrototype[1]{#1}}%
\define@key{markdownRendererPrototypes}{link}{%
  \renewcommand\markdownRendererLinkPrototype[4]{#1}}%
\define@key{markdownRendererPrototypes}{contentBlock}{%
  \renewcommand\markdownRendererContentBlockPrototype[4]{#1}}%
\define@key{markdownRendererPrototypes}{contentBlockOnlineImage}{%
  \renewcommand\markdownRendererContentBlockOnlineImagePrototype[4]{#1}}%
\define@key{markdownRendererPrototypes}{contentBlockCode}{%
  \renewcommand\markdownRendererContentBlockCodePrototype[5]{#1}}%
\define@key{markdownRendererPrototypes}{image}{%
  \renewcommand\markdownRendererImagePrototype[4]{#1}}%
\define@key{markdownRendererPrototypes}{ulBegin}{%
  \renewcommand\markdownRendererUlBeginPrototype{#1}}%
\define@key{markdownRendererPrototypes}{ulBeginTight}{%
  \renewcommand\markdownRendererUlBeginTightPrototype{#1}}%
\define@key{markdownRendererPrototypes}{ulItem}{%
  \renewcommand\markdownRendererUlItemPrototype{#1}}%
\define@key{markdownRendererPrototypes}{ulItemEnd}{%
  \renewcommand\markdownRendererUlItemEndPrototype{#1}}%
\define@key{markdownRendererPrototypes}{ulEnd}{%
  \renewcommand\markdownRendererUlEndPrototype{#1}}%
\define@key{markdownRendererPrototypes}{ulEndTight}{%
  \renewcommand\markdownRendererUlEndTightPrototype{#1}}%
\define@key{markdownRendererPrototypes}{olBegin}{%
  \renewcommand\markdownRendererOlBeginPrototype{#1}}%
\define@key{markdownRendererPrototypes}{olBeginTight}{%
  \renewcommand\markdownRendererOlBeginTightPrototype{#1}}%
\define@key{markdownRendererPrototypes}{olItem}{%
  \renewcommand\markdownRendererOlItemPrototype{#1}}%
\define@key{markdownRendererPrototypes}{olItemWithNumber}{%
  \renewcommand\markdownRendererOlItemWithNumberPrototype[1]{#1}}%
\define@key{markdownRendererPrototypes}{olItemEnd}{%
  \renewcommand\markdownRendererOlItemEndPrototype{#1}}%
\define@key{markdownRendererPrototypes}{olEnd}{%
  \renewcommand\markdownRendererOlEndPrototype{#1}}%
\define@key{markdownRendererPrototypes}{olEndTight}{%
  \renewcommand\markdownRendererOlEndTightPrototype{#1}}%
\define@key{markdownRendererPrototypes}{dlBegin}{%
  \renewcommand\markdownRendererDlBeginPrototype{#1}}%
\define@key{markdownRendererPrototypes}{dlBeginTight}{%
  \renewcommand\markdownRendererDlBeginTightPrototype{#1}}%
\define@key{markdownRendererPrototypes}{dlItem}{%
```

```
  \renewcommand\markdownRendererDlItemPrototype[1]{#1}%
\define@key{markdownRendererPrototypes}{dlItemEnd}{%
  \renewcommand\markdownRendererDlItemEndPrototype{#1}%
\define@key{markdownRendererPrototypes}{dlDefinitionBegin}{%
  \renewcommand\markdownRendererDlDefinitionBeginPrototype{#1}%
\define@key{markdownRendererPrototypes}{dlDefinitionEnd}{%
  \renewcommand\markdownRendererDlDefinitionEndPrototype{#1}%
\define@key{markdownRendererPrototypes}{dlEnd}{%
  \renewcommand\markdownRendererDlEndPrototype{#1}%
\define@key{markdownRendererPrototypes}{dlEndTight}{%
  \renewcommand\markdownRendererDlEndTightPrototype{#1}%
\define@key{markdownRendererPrototypes}{emphasis}{%
  \renewcommand\markdownRendererEmphasisPrototype[1]{#1}%
\define@key{markdownRendererPrototypes}{strongEmphasis}{%
  \renewcommand\markdownRendererStrongEmphasisPrototype[1]{#1}%
\define@key{markdownRendererPrototypes}{blockQuoteBegin}{%
  \renewcommand\markdownRendererBlockQuoteBeginPrototype{#1}%
\define@key{markdownRendererPrototypes}{blockQuoteEnd}{%
  \renewcommand\markdownRendererBlockQuoteEndPrototype{#1}%
\define@key{markdownRendererPrototypes}{inputVerbatim}{%
  \renewcommand\markdownRendererInputVerbatimPrototype[1]{#1}%
\define@key{markdownRendererPrototypes}{inputFencedCode}{%
  \renewcommand\markdownRendererInputFencedCodePrototype[2]{#1}%
\define@key{markdownRendererPrototypes}{headingOne}{%
  \renewcommand\markdownRendererHeadingOnePrototype[1]{#1}%
\define@key{markdownRendererPrototypes}{headingTwo}{%
  \renewcommand\markdownRendererHeadingTwoPrototype[1]{#1}%
\define@key{markdownRendererPrototypes}{headingThree}{%
  \renewcommand\markdownRendererHeadingThreePrototype[1]{#1}%
\define@key{markdownRendererPrototypes}{headingFour}{%
  \renewcommand\markdownRendererHeadingFourPrototype[1]{#1}%
\define@key{markdownRendererPrototypes}{headingFive}{%
  \renewcommand\markdownRendererHeadingFivePrototype[1]{#1}%
\define@key{markdownRendererPrototypes}{headingSix}{%
  \renewcommand\markdownRendererHeadingSixPrototype[1]{#1}%
\define@key{markdownRendererPrototypes}{horizontalRule}{%
  \renewcommand\markdownRendererHorizontalRulePrototype{#1}%
\define@key{markdownRendererPrototypes}{footnote}{%
  \renewcommand\markdownRendererFootnotePrototype[1]{#1}%
\define@key{markdownRendererPrototypes}{cite}{%
  \renewcommand\markdownRendererCitePrototype[1]{#1}%
\define@key{markdownRendererPrototypes}{textCite}{%
  \renewcommand\markdownRendererTextCitePrototype[1]{#1}%
\define@key{markdownRendererPrototypes}{table}{%
  \renewcommand\markdownRendererTablePrototype[3]{#1}%
\define@key{markdownRendererPrototypes}{inlineHtmlComment}{%
  \renewcommand\markdownRendererInlineHtmlCommentPrototype[1]{#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).

```
\writestatus{loading}{ConTeXt User Module / markdown}%
\startmodule[markdown]
\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 `\input`ting the file.

2.4.1 Typesetting Markdown

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

```
\let\startmarkdown\relax
\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 TEX.

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 TEX writer were hidden behind the converter functions exposed by the Lua interface (see Section 2.1).

```
749 local upper, gsub, format, length =
750   string.upper, string.gsub, string.format, string.len
751 local concat = table.concat
752 local P, R, S, V, C, Cg, Cb, Cmt, Cc, Ct, B, Cs, any =
753   lpeg.P, lpeg.R, lpeg.S, lpeg.V, lpeg.C, lpeg.Cg, lpeg.Cb,
754   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 TEX 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.

```
755 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.

```
756 function util.err(msg, exit_code)
757   io.stderr:write("markdown.lua: " .. msg .. "\n")
758   os.exit(exit_code or 1)
759 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.

```
function util.cache(dir, string, salt, transform, suffix)
  local digest = md5.sumhexa(string .. (salt or ""))
  local name = util.pathname(dir, digest .. suffix)
  local file = io.open(name, "r")
  if file == nil then -- If no cache entry exists, then create a new one.
    local file = assert(io.open(name, "w"),
      [[could not open file "]] .. name .. [[" for writing]])
    local result = string
    if transform ~= nil then
      result = transform(result)
    end
    assert(file:write(result))
    assert(file:close())
  end
  return name
end
```

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

```
function util.table_copy(t)
  local u = { }
  for k, v in pairs(t) do u[k] = v end
  return setmetatable(u, getmetatable(t))
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 [9, Chapter 21].

```
function util.expand_tabs_in_line(s, tabstop)
  local tab = tabstop or 4
  local corr = 0
  return (s:gsub("()\t", function(p)
            local sp = tab - (p - 1 + corr) % tab
            corr = corr - 1 + sp
            return string.rep(" ", sp)
          end))
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.

```
function util.walk(t, f)
```

```
  local typ = type(t)
  if typ == "string" then
    f(t)
  elseif typ == "table" then
    local i = 1
    local n
    n = t[i]
    while n do
      util.walk(n, f)
      i = i + 1
      n = t[i]
    end
  elseif typ == "function" then
    local ok, val = pcall(t)
    if ok then
      util.walk(val,f)
    end
  else
    f(tostring(t))
  end
end
```

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

```
function util.flatten(ary)
  local new = {}
  for _,v in ipairs(ary) do
    if type(v) == "table" then
      for _,w in ipairs(util.flatten(v)) do
        new[#new + 1] = w
      end
    else
      new[#new + 1] = v
    end
  end
  return new
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.

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

```
function util.rope_last(rope)
  if #rope == 0 then
    return nil
  else
    local l = rope[#rope]
    if type(l) == "table" then
      return util.rope_last(l)
    else
      return l
    end
  end
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`.

```
function util.intersperse(ary, x)
  local new = {}
  local l = #ary
  for i,v in ipairs(ary) do
    local n = #new
    new[n + 1] = v
    if i ~= l then
      new[n + 2] = x
    end
  end
  return new
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`.

```
function util.map(ary, f)
  local new = {}
  for i,v in ipairs(ary) do
    new[i] = f(v)
  end
  return new
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 occurances of escapable strings and characters (in this order).

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

```
function util.escaper(char_escapes, string_escapes)
```

Build a string of escapable characters.

```
  local char_escapes_list = ""
```

```
863   for i,_ in pairs(char_escapes) do
864     char_escapes_list = char_escapes_list .. i
865   end
```

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

```
866   local escapable = S(char_escapes_list) / char_escapes
```

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

$$\sum_{\texttt{(k,v)}\in\texttt{string\_escapes}} \texttt{P(k) / v} + \texttt{escapable}$$

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

```
867   if string_escapes then
868     for k,v in pairs(string_escapes) do
869       escapable = P(k) / v + escapable
870     end
871   end
```

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

```
872   local escape_string = Cs((escapable + any)^0)
```

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

```
873   return function(s)
874     return lpeg.match(escape_string, s)
875   end
876 end
```

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

```
877 function util.pathname(dir, file)
878   if #dir == 0 then
879     return file
880   else
881     return dir .. "/" .. file
882   end
883 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.

```
local entities = {}

local character_entities = {
  ["Tab"] = 9,
  ["NewLine"] = 10,
  ["excl"] = 33,
  ["quot"] = 34,
  ["QUOT"] = 34,
  ["num"] = 35,
  ["dollar"] = 36,
  ["percnt"] = 37,
  ["amp"] = 38,
  ["AMP"] = 38,
  ["apos"] = 39,
  ["lpar"] = 40,
  ["rpar"] = 41,
  ["ast"] = 42,
  ["midast"] = 42,
  ["plus"] = 43,
  ["comma"] = 44,
  ["period"] = 46,
  ["sol"] = 47,
  ["colon"] = 58,
  ["semi"] = 59,
  ["lt"] = 60,
  ["LT"] = 60,
  ["equals"] = 61,
  ["gt"] = 62,
  ["GT"] = 62,
  ["quest"] = 63,
  ["commat"] = 64,
  ["lsqb"] = 91,
  ["lbrack"] = 91,
  ["bsol"] = 92,
  ["rsqb"] = 93,
  ["rbrack"] = 93,
  ["Hat"] = 94,
  ["lowbar"] = 95,
  ["grave"] = 96,
  ["DiacriticalGrave"] = 96,
  ["lcub"] = 123,
  ["lbrace"] = 123,
  ["verbar"] = 124,
  ["vert"] = 124,
  ["VerticalLine"] = 124,
  ["rcub"] = 125,
  ["rbrace"] = 125,
```

```
  ["nbsp"] = 160,
  ["NonBreakingSpace"] = 160,
  ["iexcl"] = 161,
  ["cent"] = 162,
  ["pound"] = 163,
  ["curren"] = 164,
  ["yen"] = 165,
  ["brvbar"] = 166,
  ["sect"] = 167,
  ["Dot"] = 168,
  ["die"] = 168,
  ["DoubleDot"] = 168,
  ["uml"] = 168,
  ["copy"] = 169,
  ["COPY"] = 169,
  ["ordf"] = 170,
  ["laquo"] = 171,
  ["not"] = 172,
  ["shy"] = 173,
  ["reg"] = 174,
  ["circledR"] = 174,
  ["REG"] = 174,
  ["macr"] = 175,
  ["OverBar"] = 175,
  ["strns"] = 175,
  ["deg"] = 176,
  ["plusmn"] = 177,
  ["pm"] = 177,
  ["PlusMinus"] = 177,
  ["sup2"] = 178,
  ["sup3"] = 179,
  ["acute"] = 180,
  ["DiacriticalAcute"] = 180,
  ["micro"] = 181,
  ["para"] = 182,
  ["middot"] = 183,
  ["centerdot"] = 183,
  ["CenterDot"] = 183,
  ["cedil"] = 184,
  ["Cedilla"] = 184,
  ["sup1"] = 185,
  ["ordm"] = 186,
  ["raquo"] = 187,
  ["frac14"] = 188,
  ["frac12"] = 189,
  ["half"] = 189,
  ["frac34"] = 190,
```

```
978   ["iquest"] = 191,
979   ["Agrave"] = 192,
980   ["Aacute"] = 193,
981   ["Acirc"] = 194,
982   ["Atilde"] = 195,
983   ["Auml"] = 196,
984   ["Aring"] = 197,
985   ["AElig"] = 198,
986   ["Ccedil"] = 199,
987   ["Egrave"] = 200,
988   ["Eacute"] = 201,
989   ["Ecirc"] = 202,
990   ["Euml"] = 203,
991   ["Igrave"] = 204,
992   ["Iacute"] = 205,
993   ["Icirc"] = 206,
994   ["Iuml"] = 207,
995   ["ETH"] = 208,
996   ["Ntilde"] = 209,
997   ["Ograve"] = 210,
998   ["Oacute"] = 211,
999   ["Ocirc"] = 212,
1000  ["Otilde"] = 213,
1001  ["Ouml"] = 214,
1002  ["times"] = 215,
1003  ["Oslash"] = 216,
1004  ["Ugrave"] = 217,
1005  ["Uacute"] = 218,
1006  ["Ucirc"] = 219,
1007  ["Uuml"] = 220,
1008  ["Yacute"] = 221,
1009  ["THORN"] = 222,
1010  ["szlig"] = 223,
1011  ["agrave"] = 224,
1012  ["aacute"] = 225,
1013  ["acirc"] = 226,
1014  ["atilde"] = 227,
1015  ["auml"] = 228,
1016  ["aring"] = 229,
1017  ["aelig"] = 230,
1018  ["ccedil"] = 231,
1019  ["egrave"] = 232,
1020  ["eacute"] = 233,
1021  ["ecirc"] = 234,
1022  ["euml"] = 235,
1023  ["igrave"] = 236,
1024  ["iacute"] = 237,
```

```
1025    ["icirc"] = 238,
1026    ["iuml"] = 239,
1027    ["eth"] = 240,
1028    ["ntilde"] = 241,
1029    ["ograve"] = 242,
1030    ["oacute"] = 243,
1031    ["ocirc"] = 244,
1032    ["otilde"] = 245,
1033    ["ouml"] = 246,
1034    ["divide"] = 247,
1035    ["div"] = 247,
1036    ["oslash"] = 248,
1037    ["ugrave"] = 249,
1038    ["uacute"] = 250,
1039    ["ucirc"] = 251,
1040    ["uuml"] = 252,
1041    ["yacute"] = 253,
1042    ["thorn"] = 254,
1043    ["yuml"] = 255,
1044    ["Amacr"] = 256,
1045    ["amacr"] = 257,
1046    ["Abreve"] = 258,
1047    ["abreve"] = 259,
1048    ["Aogon"] = 260,
1049    ["aogon"] = 261,
1050    ["Cacute"] = 262,
1051    ["cacute"] = 263,
1052    ["Ccirc"] = 264,
1053    ["ccirc"] = 265,
1054    ["Cdot"] = 266,
1055    ["cdot"] = 267,
1056    ["Ccaron"] = 268,
1057    ["ccaron"] = 269,
1058    ["Dcaron"] = 270,
1059    ["dcaron"] = 271,
1060    ["Dstrok"] = 272,
1061    ["dstrok"] = 273,
1062    ["Emacr"] = 274,
1063    ["emacr"] = 275,
1064    ["Edot"] = 278,
1065    ["edot"] = 279,
1066    ["Eogon"] = 280,
1067    ["eogon"] = 281,
1068    ["Ecaron"] = 282,
1069    ["ecaron"] = 283,
1070    ["Gcirc"] = 284,
1071    ["gcirc"] = 285,
```

```
1072    ["Gbreve"] = 286,
1073    ["gbreve"] = 287,
1074    ["Gdot"] = 288,
1075    ["gdot"] = 289,
1076    ["Gcedil"] = 290,
1077    ["Hcirc"] = 292,
1078    ["hcirc"] = 293,
1079    ["Hstrok"] = 294,
1080    ["hstrok"] = 295,
1081    ["Itilde"] = 296,
1082    ["itilde"] = 297,
1083    ["Imacr"] = 298,
1084    ["imacr"] = 299,
1085    ["Iogon"] = 302,
1086    ["iogon"] = 303,
1087    ["Idot"] = 304,
1088    ["imath"] = 305,
1089    ["inodot"] = 305,
1090    ["IJlig"] = 306,
1091    ["ijlig"] = 307,
1092    ["Jcirc"] = 308,
1093    ["jcirc"] = 309,
1094    ["Kcedil"] = 310,
1095    ["kcedil"] = 311,
1096    ["kgreen"] = 312,
1097    ["Lacute"] = 313,
1098    ["lacute"] = 314,
1099    ["Lcedil"] = 315,
1100    ["lcedil"] = 316,
1101    ["Lcaron"] = 317,
1102    ["lcaron"] = 318,
1103    ["Lmidot"] = 319,
1104    ["lmidot"] = 320,
1105    ["Lstrok"] = 321,
1106    ["lstrok"] = 322,
1107    ["Nacute"] = 323,
1108    ["nacute"] = 324,
1109    ["Ncedil"] = 325,
1110    ["ncedil"] = 326,
1111    ["Ncaron"] = 327,
1112    ["ncaron"] = 328,
1113    ["napos"] = 329,
1114    ["ENG"] = 330,
1115    ["eng"] = 331,
1116    ["Omacr"] = 332,
1117    ["omacr"] = 333,
1118    ["Odblac"] = 336,
```

```
  ["odblac"] = 337,
  ["OElig"] = 338,
  ["oelig"] = 339,
  ["Racute"] = 340,
  ["racute"] = 341,
  ["Rcedil"] = 342,
  ["rcedil"] = 343,
  ["Rcaron"] = 344,
  ["rcaron"] = 345,
  ["Sacute"] = 346,
  ["sacute"] = 347,
  ["Scirc"] = 348,
  ["scirc"] = 349,
  ["Scedil"] = 350,
  ["scedil"] = 351,
  ["Scaron"] = 352,
  ["scaron"] = 353,
  ["Tcedil"] = 354,
  ["tcedil"] = 355,
  ["Tcaron"] = 356,
  ["tcaron"] = 357,
  ["Tstrok"] = 358,
  ["tstrok"] = 359,
  ["Utilde"] = 360,
  ["utilde"] = 361,
  ["Umacr"] = 362,
  ["umacr"] = 363,
  ["Ubreve"] = 364,
  ["ubreve"] = 365,
  ["Uring"] = 366,
  ["uring"] = 367,
  ["Udblac"] = 368,
  ["udblac"] = 369,
  ["Uogon"] = 370,
  ["uogon"] = 371,
  ["Wcirc"] = 372,
  ["wcirc"] = 373,
  ["Ycirc"] = 374,
  ["ycirc"] = 375,
  ["Yuml"] = 376,
  ["Zacute"] = 377,
  ["zacute"] = 378,
  ["Zdot"] = 379,
  ["zdot"] = 380,
  ["Zcaron"] = 381,
  ["zcaron"] = 382,
  ["fnof"] = 402,
```

```
  ["imped"] = 437,
  ["gacute"] = 501,
  ["jmath"] = 567,
  ["circ"] = 710,
  ["caron"] = 711,
  ["Hacek"] = 711,
  ["breve"] = 728,
  ["Breve"] = 728,
  ["dot"] = 729,
  ["DiacriticalDot"] = 729,
  ["ring"] = 730,
  ["ogon"] = 731,
  ["tilde"] = 732,
  ["DiacriticalTilde"] = 732,
  ["dblac"] = 733,
  ["DiacriticalDoubleAcute"] = 733,
  ["DownBreve"] = 785,
  ["UnderBar"] = 818,
  ["Alpha"] = 913,
  ["Beta"] = 914,
  ["Gamma"] = 915,
  ["Delta"] = 916,
  ["Epsilon"] = 917,
  ["Zeta"] = 918,
  ["Eta"] = 919,
  ["Theta"] = 920,
  ["Iota"] = 921,
  ["Kappa"] = 922,
  ["Lambda"] = 923,
  ["Mu"] = 924,
  ["Nu"] = 925,
  ["Xi"] = 926,
  ["Omicron"] = 927,
  ["Pi"] = 928,
  ["Rho"] = 929,
  ["Sigma"] = 931,
  ["Tau"] = 932,
  ["Upsilon"] = 933,
  ["Phi"] = 934,
  ["Chi"] = 935,
  ["Psi"] = 936,
  ["Omega"] = 937,
  ["alpha"] = 945,
  ["beta"] = 946,
  ["gamma"] = 947,
  ["delta"] = 948,
  ["epsiv"] = 949,
```

```
["varepsilon"] = 949,
["epsilon"] = 949,
["zeta"] = 950,
["eta"] = 951,
["theta"] = 952,
["iota"] = 953,
["kappa"] = 954,
["lambda"] = 955,
["mu"] = 956,
["nu"] = 957,
["xi"] = 958,
["omicron"] = 959,
["pi"] = 960,
["rho"] = 961,
["sigmav"] = 962,
["varsigma"] = 962,
["sigmaf"] = 962,
["sigma"] = 963,
["tau"] = 964,
["upsi"] = 965,
["upsilon"] = 965,
["phi"] = 966,
["phiv"] = 966,
["varphi"] = 966,
["chi"] = 967,
["psi"] = 968,
["omega"] = 969,
["thetav"] = 977,
["vartheta"] = 977,
["thetasym"] = 977,
["Upsi"] = 978,
["upsih"] = 978,
["straightphi"] = 981,
["piv"] = 982,
["varpi"] = 982,
["Gammad"] = 988,
["gammad"] = 989,
["digamma"] = 989,
["kappav"] = 1008,
["varkappa"] = 1008,
["rhov"] = 1009,
["varrho"] = 1009,
["epsi"] = 1013,
["straightepsilon"] = 1013,
["bepsi"] = 1014,
["backepsilon"] = 1014,
["IOcy"] = 1025,
```

```
["DJcy"] = 1026,
["GJcy"] = 1027,
["Jukcy"] = 1028,
["DScy"] = 1029,
["Iukcy"] = 1030,
["YIcy"] = 1031,
["Jsercy"] = 1032,
["LJcy"] = 1033,
["NJcy"] = 1034,
["TSHcy"] = 1035,
["KJcy"] = 1036,
["Ubrcy"] = 1038,
["DZcy"] = 1039,
["Acy"] = 1040,
["Bcy"] = 1041,
["Vcy"] = 1042,
["Gcy"] = 1043,
["Dcy"] = 1044,
["IEcy"] = 1045,
["ZHcy"] = 1046,
["Zcy"] = 1047,
["Icy"] = 1048,
["Jcy"] = 1049,
["Kcy"] = 1050,
["Lcy"] = 1051,
["Mcy"] = 1052,
["Ncy"] = 1053,
["Ocy"] = 1054,
["Pcy"] = 1055,
["Rcy"] = 1056,
["Scy"] = 1057,
["Tcy"] = 1058,
["Ucy"] = 1059,
["Fcy"] = 1060,
["KHcy"] = 1061,
["TScy"] = 1062,
["CHcy"] = 1063,
["SHcy"] = 1064,
["SHCHcy"] = 1065,
["HARDcy"] = 1066,
["Ycy"] = 1067,
["SOFTcy"] = 1068,
["Ecy"] = 1069,
["YUcy"] = 1070,
["YAcy"] = 1071,
["acy"] = 1072,
["bcy"] = 1073,
```

```
1307    ["vcy"] = 1074,
1308    ["gcy"] = 1075,
1309    ["dcy"] = 1076,
1310    ["iecy"] = 1077,
1311    ["zhcy"] = 1078,
1312    ["zcy"] = 1079,
1313    ["icy"] = 1080,
1314    ["jcy"] = 1081,
1315    ["kcy"] = 1082,
1316    ["lcy"] = 1083,
1317    ["mcy"] = 1084,
1318    ["ncy"] = 1085,
1319    ["ocy"] = 1086,
1320    ["pcy"] = 1087,
1321    ["rcy"] = 1088,
1322    ["scy"] = 1089,
1323    ["tcy"] = 1090,
1324    ["ucy"] = 1091,
1325    ["fcy"] = 1092,
1326    ["khcy"] = 1093,
1327    ["tscy"] = 1094,
1328    ["chcy"] = 1095,
1329    ["shcy"] = 1096,
1330    ["shchcy"] = 1097,
1331    ["hardcy"] = 1098,
1332    ["ycy"] = 1099,
1333    ["softcy"] = 1100,
1334    ["ecy"] = 1101,
1335    ["yucy"] = 1102,
1336    ["yacy"] = 1103,
1337    ["iocy"] = 1105,
1338    ["djcy"] = 1106,
1339    ["gjcy"] = 1107,
1340    ["jukcy"] = 1108,
1341    ["dscy"] = 1109,
1342    ["iukcy"] = 1110,
1343    ["yicy"] = 1111,
1344    ["jsercy"] = 1112,
1345    ["ljcy"] = 1113,
1346    ["njcy"] = 1114,
1347    ["tshcy"] = 1115,
1348    ["kjcy"] = 1116,
1349    ["ubrcy"] = 1118,
1350    ["dzcy"] = 1119,
1351    ["ensp"] = 8194,
1352    ["emsp"] = 8195,
1353    ["emsp13"] = 8196,
```

```
["emsp14"] = 8197,
["numsp"] = 8199,
["puncsp"] = 8200,
["thinsp"] = 8201,
["ThinSpace"] = 8201,
["hairsp"] = 8202,
["VeryThinSpace"] = 8202,
["ZeroWidthSpace"] = 8203,
["NegativeVeryThinSpace"] = 8203,
["NegativeThinSpace"] = 8203,
["NegativeMediumSpace"] = 8203,
["NegativeThickSpace"] = 8203,
["zwnj"] = 8204,
["zwj"] = 8205,
["lrm"] = 8206,
["rlm"] = 8207,
["hyphen"] = 8208,
["dash"] = 8208,
["ndash"] = 8211,
["mdash"] = 8212,
["horbar"] = 8213,
["Verbar"] = 8214,
["Vert"] = 8214,
["lsquo"] = 8216,
["OpenCurlyQuote"] = 8216,
["rsquo"] = 8217,
["rsquor"] = 8217,
["CloseCurlyQuote"] = 8217,
["lsquor"] = 8218,
["sbquo"] = 8218,
["ldquo"] = 8220,
["OpenCurlyDoubleQuote"] = 8220,
["rdquo"] = 8221,
["rdquor"] = 8221,
["CloseCurlyDoubleQuote"] = 8221,
["ldquor"] = 8222,
["bdquo"] = 8222,
["dagger"] = 8224,
["Dagger"] = 8225,
["ddagger"] = 8225,
["bull"] = 8226,
["bullet"] = 8226,
["nldr"] = 8229,
["hellip"] = 8230,
["mldr"] = 8230,
["permil"] = 8240,
["pertenk"] = 8241,
```

```
["prime"] = 8242,
["Prime"] = 8243,
["tprime"] = 8244,
["bprime"] = 8245,
["backprime"] = 8245,
["lsaquo"] = 8249,
["rsaquo"] = 8250,
["oline"] = 8254,
["caret"] = 8257,
["hybull"] = 8259,
["frasl"] = 8260,
["bsemi"] = 8271,
["qprime"] = 8279,
["MediumSpace"] = 8287,
["NoBreak"] = 8288,
["ApplyFunction"] = 8289,
["af"] = 8289,
["InvisibleTimes"] = 8290,
["it"] = 8290,
["InvisibleComma"] = 8291,
["ic"] = 8291,
["euro"] = 8364,
["tdot"] = 8411,
["TripleDot"] = 8411,
["DotDot"] = 8412,
["Copf"] = 8450,
["complexes"] = 8450,
["incare"] = 8453,
["gscr"] = 8458,
["hamilt"] = 8459,
["HilbertSpace"] = 8459,
["Hscr"] = 8459,
["Hfr"] = 8460,
["Poincareplane"] = 8460,
["quaternions"] = 8461,
["Hopf"] = 8461,
["planckh"] = 8462,
["planck"] = 8463,
["hbar"] = 8463,
["plankv"] = 8463,
["hslash"] = 8463,
["Iscr"] = 8464,
["imagline"] = 8464,
["image"] = 8465,
["Im"] = 8465,
["imagpart"] = 8465,
["Ifr"] = 8465,
```

```
1448    ["Lscr"] = 8466,
1449    ["lagran"] = 8466,
1450    ["Laplacetrf"] = 8466,
1451    ["ell"] = 8467,
1452    ["Nopf"] = 8469,
1453    ["naturals"] = 8469,
1454    ["numero"] = 8470,
1455    ["copysr"] = 8471,
1456    ["weierp"] = 8472,
1457    ["wp"] = 8472,
1458    ["Popf"] = 8473,
1459    ["primes"] = 8473,
1460    ["rationals"] = 8474,
1461    ["Qopf"] = 8474,
1462    ["Rscr"] = 8475,
1463    ["realine"] = 8475,
1464    ["real"] = 8476,
1465    ["Re"] = 8476,
1466    ["realpart"] = 8476,
1467    ["Rfr"] = 8476,
1468    ["reals"] = 8477,
1469    ["Ropf"] = 8477,
1470    ["rx"] = 8478,
1471    ["trade"] = 8482,
1472    ["TRADE"] = 8482,
1473    ["integers"] = 8484,
1474    ["Zopf"] = 8484,
1475    ["ohm"] = 8486,
1476    ["mho"] = 8487,
1477    ["Zfr"] = 8488,
1478    ["zeetrf"] = 8488,
1479    ["iiota"] = 8489,
1480    ["angst"] = 8491,
1481    ["bernou"] = 8492,
1482    ["Bernoullis"] = 8492,
1483    ["Bscr"] = 8492,
1484    ["Cfr"] = 8493,
1485    ["Cayleys"] = 8493,
1486    ["escr"] = 8495,
1487    ["Escr"] = 8496,
1488    ["expectation"] = 8496,
1489    ["Fscr"] = 8497,
1490    ["Fouriertrf"] = 8497,
1491    ["phmmat"] = 8499,
1492    ["Mellintrf"] = 8499,
1493    ["Mscr"] = 8499,
1494    ["order"] = 8500,
```

```
["orderof"] = 8500,
["oscr"] = 8500,
["alefsym"] = 8501,
["aleph"] = 8501,
["beth"] = 8502,
["gimel"] = 8503,
["daleth"] = 8504,
["CapitalDifferentialD"] = 8517,
["DD"] = 8517,
["DifferentialD"] = 8518,
["dd"] = 8518,
["ExponentialE"] = 8519,
["exponentiale"] = 8519,
["ee"] = 8519,
["ImaginaryI"] = 8520,
["ii"] = 8520,
["frac13"] = 8531,
["frac23"] = 8532,
["frac15"] = 8533,
["frac25"] = 8534,
["frac35"] = 8535,
["frac45"] = 8536,
["frac16"] = 8537,
["frac56"] = 8538,
["frac18"] = 8539,
["frac38"] = 8540,
["frac58"] = 8541,
["frac78"] = 8542,
["larr"] = 8592,
["leftarrow"] = 8592,
["LeftArrow"] = 8592,
["slarr"] = 8592,
["ShortLeftArrow"] = 8592,
["uarr"] = 8593,
["uparrow"] = 8593,
["UpArrow"] = 8593,
["ShortUpArrow"] = 8593,
["rarr"] = 8594,
["rightarrow"] = 8594,
["RightArrow"] = 8594,
["srarr"] = 8594,
["ShortRightArrow"] = 8594,
["darr"] = 8595,
["downarrow"] = 8595,
["DownArrow"] = 8595,
["ShortDownArrow"] = 8595,
["harr"] = 8596,
```

```
["leftrightarrow"] = 8596,
["LeftRightArrow"] = 8596,
["varr"] = 8597,
["updownarrow"] = 8597,
["UpDownArrow"] = 8597,
["nwarr"] = 8598,
["UpperLeftArrow"] = 8598,
["nwarrow"] = 8598,
["nearr"] = 8599,
["UpperRightArrow"] = 8599,
["nearrow"] = 8599,
["searr"] = 8600,
["searrow"] = 8600,
["LowerRightArrow"] = 8600,
["swarr"] = 8601,
["swarrow"] = 8601,
["LowerLeftArrow"] = 8601,
["nlarr"] = 8602,
["nleftarrow"] = 8602,
["nrarr"] = 8603,
["nrightarrow"] = 8603,
["rarrw"] = 8605,
["rightsquigarrow"] = 8605,
["Larr"] = 8606,
["twoheadleftarrow"] = 8606,
["Uarr"] = 8607,
["Rarr"] = 8608,
["twoheadrightarrow"] = 8608,
["Darr"] = 8609,
["larrtl"] = 8610,
["leftarrowtail"] = 8610,
["rarrtl"] = 8611,
["rightarrowtail"] = 8611,
["LeftTeeArrow"] = 8612,
["mapstoleft"] = 8612,
["UpTeeArrow"] = 8613,
["mapstoup"] = 8613,
["map"] = 8614,
["RightTeeArrow"] = 8614,
["mapsto"] = 8614,
["DownTeeArrow"] = 8615,
["mapstodown"] = 8615,
["larrhk"] = 8617,
["hookleftarrow"] = 8617,
["rarrhk"] = 8618,
["hookrightarrow"] = 8618,
["larrlp"] = 8619,
```

```
["looparrowleft"] = 8619,
["rarrlp"] = 8620,
["looparrowright"] = 8620,
["harrw"] = 8621,
["leftrightsquigarrow"] = 8621,
["nharr"] = 8622,
["nleftrightarrow"] = 8622,
["lsh"] = 8624,
["Lsh"] = 8624,
["rsh"] = 8625,
["Rsh"] = 8625,
["ldsh"] = 8626,
["rdsh"] = 8627,
["crarr"] = 8629,
["cularr"] = 8630,
["curvearrowleft"] = 8630,
["curarr"] = 8631,
["curvearrowright"] = 8631,
["olarr"] = 8634,
["circlearrowleft"] = 8634,
["orarr"] = 8635,
["circlearrowright"] = 8635,
["lharu"] = 8636,
["LeftVector"] = 8636,
["leftharpoonup"] = 8636,
["lhard"] = 8637,
["leftharpoondown"] = 8637,
["DownLeftVector"] = 8637,
["uharr"] = 8638,
["upharpoonright"] = 8638,
["RightUpVector"] = 8638,
["uharl"] = 8639,
["upharpoonleft"] = 8639,
["LeftUpVector"] = 8639,
["rharu"] = 8640,
["RightVector"] = 8640,
["rightharpoonup"] = 8640,
["rhard"] = 8641,
["rightharpoondown"] = 8641,
["DownRightVector"] = 8641,
["dharr"] = 8642,
["RightDownVector"] = 8642,
["downharpoonright"] = 8642,
["dharl"] = 8643,
["LeftDownVector"] = 8643,
["downharpoonleft"] = 8643,
["rlarr"] = 8644,
```

```
["rightleftarrows"] = 8644,
["RightArrowLeftArrow"] = 8644,
["udarr"] = 8645,
["UpArrowDownArrow"] = 8645,
["lrarr"] = 8646,
["leftrightarrows"] = 8646,
["LeftArrowRightArrow"] = 8646,
["llarr"] = 8647,
["leftleftarrows"] = 8647,
["uuarr"] = 8648,
["upuparrows"] = 8648,
["rrarr"] = 8649,
["rightrightarrows"] = 8649,
["ddarr"] = 8650,
["downdownarrows"] = 8650,
["lrhar"] = 8651,
["ReverseEquilibrium"] = 8651,
["leftrightharpoons"] = 8651,
["rlhar"] = 8652,
["rightleftharpoons"] = 8652,
["Equilibrium"] = 8652,
["nlArr"] = 8653,
["nLeftarrow"] = 8653,
["nhArr"] = 8654,
["nLeftrightarrow"] = 8654,
["nrArr"] = 8655,
["nRightarrow"] = 8655,
["lArr"] = 8656,
["Leftarrow"] = 8656,
["DoubleLeftArrow"] = 8656,
["uArr"] = 8657,
["Uparrow"] = 8657,
["DoubleUpArrow"] = 8657,
["rArr"] = 8658,
["Rightarrow"] = 8658,
["Implies"] = 8658,
["DoubleRightArrow"] = 8658,
["dArr"] = 8659,
["Downarrow"] = 8659,
["DoubleDownArrow"] = 8659,
["hArr"] = 8660,
["Leftrightarrow"] = 8660,
["DoubleLeftRightArrow"] = 8660,
["iff"] = 8660,
["vArr"] = 8661,
["Updownarrow"] = 8661,
["DoubleUpDownArrow"] = 8661,
```

```
["nwArr"] = 8662,
["neArr"] = 8663,
["seArr"] = 8664,
["swArr"] = 8665,
["lAarr"] = 8666,
["Lleftarrow"] = 8666,
["rAarr"] = 8667,
["Rrightarrow"] = 8667,
["zigrarr"] = 8669,
["larrb"] = 8676,
["LeftArrowBar"] = 8676,
["rarrb"] = 8677,
["RightArrowBar"] = 8677,
["duarr"] = 8693,
["DownArrowUpArrow"] = 8693,
["loarr"] = 8701,
["roarr"] = 8702,
["hoarr"] = 8703,
["forall"] = 8704,
["ForAll"] = 8704,
["comp"] = 8705,
["complement"] = 8705,
["part"] = 8706,
["PartialD"] = 8706,
["exist"] = 8707,
["Exists"] = 8707,
["nexist"] = 8708,
["NotExists"] = 8708,
["nexists"] = 8708,
["empty"] = 8709,
["emptyset"] = 8709,
["emptyv"] = 8709,
["varnothing"] = 8709,
["nabla"] = 8711,
["Del"] = 8711,
["isin"] = 8712,
["isinv"] = 8712,
["Element"] = 8712,
["in"] = 8712,
["notin"] = 8713,
["NotElement"] = 8713,
["notinva"] = 8713,
["niv"] = 8715,
["ReverseElement"] = 8715,
["ni"] = 8715,
["SuchThat"] = 8715,
["notni"] = 8716,
```

```
["notniva"] = 8716,
["NotReverseElement"] = 8716,
["prod"] = 8719,
["Product"] = 8719,
["coprod"] = 8720,
["Coproduct"] = 8720,
["sum"] = 8721,
["Sum"] = 8721,
["minus"] = 8722,
["mnplus"] = 8723,
["mp"] = 8723,
["MinusPlus"] = 8723,
["plusdo"] = 8724,
["dotplus"] = 8724,
["setmn"] = 8726,
["setminus"] = 8726,
["Backslash"] = 8726,
["ssetmn"] = 8726,
["smallsetminus"] = 8726,
["lowast"] = 8727,
["compfn"] = 8728,
["SmallCircle"] = 8728,
["radic"] = 8730,
["Sqrt"] = 8730,
["prop"] = 8733,
["propto"] = 8733,
["Proportional"] = 8733,
["vprop"] = 8733,
["varpropto"] = 8733,
["infin"] = 8734,
["angrt"] = 8735,
["ang"] = 8736,
["angle"] = 8736,
["angmsd"] = 8737,
["measuredangle"] = 8737,
["angsph"] = 8738,
["mid"] = 8739,
["VerticalBar"] = 8739,
["smid"] = 8739,
["shortmid"] = 8739,
["nmid"] = 8740,
["NotVerticalBar"] = 8740,
["nsmid"] = 8740,
["nshortmid"] = 8740,
["par"] = 8741,
["parallel"] = 8741,
["DoubleVerticalBar"] = 8741,
```

```
["spar"] = 8741,
["shortparallel"] = 8741,
["npar"] = 8742,
["nparallel"] = 8742,
["NotDoubleVerticalBar"] = 8742,
["nspar"] = 8742,
["nshortparallel"] = 8742,
["and"] = 8743,
["wedge"] = 8743,
["or"] = 8744,
["vee"] = 8744,
["cap"] = 8745,
["cup"] = 8746,
["int"] = 8747,
["Integral"] = 8747,
["Int"] = 8748,
["tint"] = 8749,
["iiint"] = 8749,
["conint"] = 8750,
["oint"] = 8750,
["ContourIntegral"] = 8750,
["Conint"] = 8751,
["DoubleContourIntegral"] = 8751,
["Cconint"] = 8752,
["cwint"] = 8753,
["cwconint"] = 8754,
["ClockwiseContourIntegral"] = 8754,
["awconint"] = 8755,
["CounterClockwiseContourIntegral"] = 8755,
["there4"] = 8756,
["therefore"] = 8756,
["Therefore"] = 8756,
["becaus"] = 8757,
["because"] = 8757,
["Because"] = 8757,
["ratio"] = 8758,
["Colon"] = 8759,
["Proportion"] = 8759,
["minusd"] = 8760,
["dotminus"] = 8760,
["mDDot"] = 8762,
["homtht"] = 8763,
["sim"] = 8764,
["Tilde"] = 8764,
["thksim"] = 8764,
["thicksim"] = 8764,
["bsim"] = 8765,
```

```
["backsim"] = 8765,
["ac"] = 8766,
["mstpos"] = 8766,
["acd"] = 8767,
["wreath"] = 8768,
["VerticalTilde"] = 8768,
["wr"] = 8768,
["nsim"] = 8769,
["NotTilde"] = 8769,
["esim"] = 8770,
["EqualTilde"] = 8770,
["eqsim"] = 8770,
["sime"] = 8771,
["TildeEqual"] = 8771,
["simeq"] = 8771,
["nsime"] = 8772,
["nsimeq"] = 8772,
["NotTildeEqual"] = 8772,
["cong"] = 8773,
["TildeFullEqual"] = 8773,
["simne"] = 8774,
["ncong"] = 8775,
["NotTildeFullEqual"] = 8775,
["asymp"] = 8776,
["ap"] = 8776,
["TildeTilde"] = 8776,
["approx"] = 8776,
["thkap"] = 8776,
["thickapprox"] = 8776,
["nap"] = 8777,
["NotTildeTilde"] = 8777,
["napprox"] = 8777,
["ape"] = 8778,
["approxeq"] = 8778,
["apid"] = 8779,
["bcong"] = 8780,
["backcong"] = 8780,
["asympeq"] = 8781,
["CupCap"] = 8781,
["bump"] = 8782,
["HumpDownHump"] = 8782,
["Bumpeq"] = 8782,
["bumpe"] = 8783,
["HumpEqual"] = 8783,
["bumpeq"] = 8783,
["esdot"] = 8784,
["DotEqual"] = 8784,
```

```
["doteq"] = 8784,
["eDot"] = 8785,
["doteqdot"] = 8785,
["efDot"] = 8786,
["fallingdotseq"] = 8786,
["erDot"] = 8787,
["risingdotseq"] = 8787,
["colone"] = 8788,
["coloneq"] = 8788,
["Assign"] = 8788,
["ecolon"] = 8789,
["eqcolon"] = 8789,
["ecir"] = 8790,
["eqcirc"] = 8790,
["cire"] = 8791,
["circeq"] = 8791,
["wedgeq"] = 8793,
["veeeq"] = 8794,
["trie"] = 8796,
["triangleq"] = 8796,
["equest"] = 8799,
["questeq"] = 8799,
["ne"] = 8800,
["NotEqual"] = 8800,
["equiv"] = 8801,
["Congruent"] = 8801,
["nequiv"] = 8802,
["NotCongruent"] = 8802,
["le"] = 8804,
["leq"] = 8804,
["ge"] = 8805,
["GreaterEqual"] = 8805,
["geq"] = 8805,
["lE"] = 8806,
["LessFullEqual"] = 8806,
["leqq"] = 8806,
["gE"] = 8807,
["GreaterFullEqual"] = 8807,
["geqq"] = 8807,
["lnE"] = 8808,
["lneqq"] = 8808,
["gnE"] = 8809,
["gneqq"] = 8809,
["Lt"] = 8810,
["NestedLessLess"] = 8810,
["ll"] = 8810,
["Gt"] = 8811,
```

```
["NestedGreaterGreater"] = 8811,
["gg"] = 8811,
["twixt"] = 8812,
["between"] = 8812,
["NotCupCap"] = 8813,
["nlt"] = 8814,
["NotLess"] = 8814,
["nless"] = 8814,
["ngt"] = 8815,
["NotGreater"] = 8815,
["ngtr"] = 8815,
["nle"] = 8816,
["NotLessEqual"] = 8816,
["nleq"] = 8816,
["nge"] = 8817,
["NotGreaterEqual"] = 8817,
["ngeq"] = 8817,
["lsim"] = 8818,
["LessTilde"] = 8818,
["lesssim"] = 8818,
["gsim"] = 8819,
["gtrsim"] = 8819,
["GreaterTilde"] = 8819,
["nlsim"] = 8820,
["NotLessTilde"] = 8820,
["ngsim"] = 8821,
["NotGreaterTilde"] = 8821,
["lg"] = 8822,
["lessgtr"] = 8822,
["LessGreater"] = 8822,
["gl"] = 8823,
["gtrless"] = 8823,
["GreaterLess"] = 8823,
["ntlg"] = 8824,
["NotLessGreater"] = 8824,
["ntgl"] = 8825,
["NotGreaterLess"] = 8825,
["pr"] = 8826,
["Precedes"] = 8826,
["prec"] = 8826,
["sc"] = 8827,
["Succeeds"] = 8827,
["succ"] = 8827,
["prcue"] = 8828,
["PrecedesSlantEqual"] = 8828,
["preccurlyeq"] = 8828,
["sccue"] = 8829,
```

```
["SucceedsSlantEqual"] = 8829,
["succcurlyeq"] = 8829,
["prsim"] = 8830,
["precsim"] = 8830,
["PrecedesTilde"] = 8830,
["scsim"] = 8831,
["succsim"] = 8831,
["SucceedsTilde"] = 8831,
["npr"] = 8832,
["nprec"] = 8832,
["NotPrecedes"] = 8832,
["nsc"] = 8833,
["nsucc"] = 8833,
["NotSucceeds"] = 8833,
["sub"] = 8834,
["subset"] = 8834,
["sup"] = 8835,
["supset"] = 8835,
["Superset"] = 8835,
["nsub"] = 8836,
["nsup"] = 8837,
["sube"] = 8838,
["SubsetEqual"] = 8838,
["subseteq"] = 8838,
["supe"] = 8839,
["supseteq"] = 8839,
["SupersetEqual"] = 8839,
["nsube"] = 8840,
["nsubseteq"] = 8840,
["NotSubsetEqual"] = 8840,
["nsupe"] = 8841,
["nsupseteq"] = 8841,
["NotSupersetEqual"] = 8841,
["subne"] = 8842,
["subsetneq"] = 8842,
["supne"] = 8843,
["supsetneq"] = 8843,
["cupdot"] = 8845,
["uplus"] = 8846,
["UnionPlus"] = 8846,
["sqsub"] = 8847,
["SquareSubset"] = 8847,
["sqsubset"] = 8847,
["sqsup"] = 8848,
["SquareSuperset"] = 8848,
["sqsupset"] = 8848,
["sqsube"] = 8849,
```

```
["SquareSubsetEqual"] = 8849,
["sqsubseteq"] = 8849,
["sqsupe"] = 8850,
["SquareSupersetEqual"] = 8850,
["sqsupseteq"] = 8850,
["sqcap"] = 8851,
["SquareIntersection"] = 8851,
["sqcup"] = 8852,
["SquareUnion"] = 8852,
["oplus"] = 8853,
["CirclePlus"] = 8853,
["ominus"] = 8854,
["CircleMinus"] = 8854,
["otimes"] = 8855,
["CircleTimes"] = 8855,
["osol"] = 8856,
["odot"] = 8857,
["CircleDot"] = 8857,
["ocir"] = 8858,
["circledcirc"] = 8858,
["oast"] = 8859,
["circledast"] = 8859,
["odash"] = 8861,
["circleddash"] = 8861,
["plusb"] = 8862,
["boxplus"] = 8862,
["minusb"] = 8863,
["boxminus"] = 8863,
["timesb"] = 8864,
["boxtimes"] = 8864,
["sdotb"] = 8865,
["dotsquare"] = 8865,
["vdash"] = 8866,
["RightTee"] = 8866,
["dashv"] = 8867,
["LeftTee"] = 8867,
["top"] = 8868,
["DownTee"] = 8868,
["bottom"] = 8869,
["bot"] = 8869,
["perp"] = 8869,
["UpTee"] = 8869,
["models"] = 8871,
["vDash"] = 8872,
["DoubleRightTee"] = 8872,
["Vdash"] = 8873,
["Vvdash"] = 8874,
```

```
["VDash"] = 8875,
["nvdash"] = 8876,
["nvDash"] = 8877,
["nVdash"] = 8878,
["nVDash"] = 8879,
["prurel"] = 8880,
["vltri"] = 8882,
["vartriangleleft"] = 8882,
["LeftTriangle"] = 8882,
["vrtri"] = 8883,
["vartriangleright"] = 8883,
["RightTriangle"] = 8883,
["ltrie"] = 8884,
["trianglelefteq"] = 8884,
["LeftTriangleEqual"] = 8884,
["rtrie"] = 8885,
["trianglerighteq"] = 8885,
["RightTriangleEqual"] = 8885,
["origof"] = 8886,
["imof"] = 8887,
["mumap"] = 8888,
["multimap"] = 8888,
["hercon"] = 8889,
["intcal"] = 8890,
["intercal"] = 8890,
["veebar"] = 8891,
["barvee"] = 8893,
["angrtvb"] = 8894,
["lrtri"] = 8895,
["xwedge"] = 8896,
["Wedge"] = 8896,
["bigwedge"] = 8896,
["xvee"] = 8897,
["Vee"] = 8897,
["bigvee"] = 8897,
["xcap"] = 8898,
["Intersection"] = 8898,
["bigcap"] = 8898,
["xcup"] = 8899,
["Union"] = 8899,
["bigcup"] = 8899,
["diam"] = 8900,
["diamond"] = 8900,
["Diamond"] = 8900,
["sdot"] = 8901,
["sstarf"] = 8902,
["Star"] = 8902,
```

```
["divonx"] = 8903,
["divideontimes"] = 8903,
["bowtie"] = 8904,
["ltimes"] = 8905,
["rtimes"] = 8906,
["lthree"] = 8907,
["leftthreetimes"] = 8907,
["rthree"] = 8908,
["rightthreetimes"] = 8908,
["bsime"] = 8909,
["backsimeq"] = 8909,
["cuvee"] = 8910,
["curlyvee"] = 8910,
["cuwed"] = 8911,
["curlywedge"] = 8911,
["Sub"] = 8912,
["Subset"] = 8912,
["Sup"] = 8913,
["Supset"] = 8913,
["Cap"] = 8914,
["Cup"] = 8915,
["fork"] = 8916,
["pitchfork"] = 8916,
["epar"] = 8917,
["ltdot"] = 8918,
["lessdot"] = 8918,
["gtdot"] = 8919,
["gtrdot"] = 8919,
["Ll"] = 8920,
["Gg"] = 8921,
["ggg"] = 8921,
["leg"] = 8922,
["LessEqualGreater"] = 8922,
["lesseqgtr"] = 8922,
["gel"] = 8923,
["gtreqless"] = 8923,
["GreaterEqualLess"] = 8923,
["cuepr"] = 8926,
["curlyeqprec"] = 8926,
["cuesc"] = 8927,
["curlyeqsucc"] = 8927,
["nprcue"] = 8928,
["NotPrecedesSlantEqual"] = 8928,
["nsccue"] = 8929,
["NotSucceedsSlantEqual"] = 8929,
["nsqsube"] = 8930,
["NotSquareSubsetEqual"] = 8930,
```

```
["nsqsupe"] = 8931,
["NotSquareSupersetEqual"] = 8931,
["lnsim"] = 8934,
["gnsim"] = 8935,
["prnsim"] = 8936,
["precnsim"] = 8936,
["scnsim"] = 8937,
["succnsim"] = 8937,
["nltri"] = 8938,
["ntriangleleft"] = 8938,
["NotLeftTriangle"] = 8938,
["nrtri"] = 8939,
["ntriangleright"] = 8939,
["NotRightTriangle"] = 8939,
["nltrie"] = 8940,
["ntrianglelefteq"] = 8940,
["NotLeftTriangleEqual"] = 8940,
["nrtrie"] = 8941,
["ntrianglerighteq"] = 8941,
["NotRightTriangleEqual"] = 8941,
["vellip"] = 8942,
["ctdot"] = 8943,
["utdot"] = 8944,
["dtdot"] = 8945,
["disin"] = 8946,
["isinsv"] = 8947,
["isins"] = 8948,
["isindot"] = 8949,
["notinvc"] = 8950,
["notinvb"] = 8951,
["isinE"] = 8953,
["nisd"] = 8954,
["xnis"] = 8955,
["nis"] = 8956,
["notnivc"] = 8957,
["notnivb"] = 8958,
["barwed"] = 8965,
["barwedge"] = 8965,
["Barwed"] = 8966,
["doublebarwedge"] = 8966,
["lceil"] = 8968,
["LeftCeiling"] = 8968,
["rceil"] = 8969,
["RightCeiling"] = 8969,
["lfloor"] = 8970,
["LeftFloor"] = 8970,
["rfloor"] = 8971,
```

```
["RightFloor"] = 8971,
["drcrop"] = 8972,
["dlcrop"] = 8973,
["urcrop"] = 8974,
["ulcrop"] = 8975,
["bnot"] = 8976,
["profline"] = 8978,
["profsurf"] = 8979,
["telrec"] = 8981,
["target"] = 8982,
["ulcorn"] = 8988,
["ulcorner"] = 8988,
["urcorn"] = 8989,
["urcorner"] = 8989,
["dlcorn"] = 8990,
["llcorner"] = 8990,
["drcorn"] = 8991,
["lrcorner"] = 8991,
["frown"] = 8994,
["sfrown"] = 8994,
["smile"] = 8995,
["ssmile"] = 8995,
["cylcty"] = 9005,
["profalar"] = 9006,
["topbot"] = 9014,
["ovbar"] = 9021,
["solbar"] = 9023,
["angzarr"] = 9084,
["lmoust"] = 9136,
["lmoustache"] = 9136,
["rmoust"] = 9137,
["rmoustache"] = 9137,
["tbrk"] = 9140,
["OverBracket"] = 9140,
["bbrk"] = 9141,
["UnderBracket"] = 9141,
["bbrktbrk"] = 9142,
["OverParenthesis"] = 9180,
["UnderParenthesis"] = 9181,
["OverBrace"] = 9182,
["UnderBrace"] = 9183,
["trpezium"] = 9186,
["elinters"] = 9191,
["blank"] = 9251,
["oS"] = 9416,
["circledS"] = 9416,
["boxh"] = 9472,
```

```
  ["HorizontalLine"] = 9472,
  ["boxv"] = 9474,
  ["boxdr"] = 9484,
  ["boxdl"] = 9488,
  ["boxur"] = 9492,
  ["boxul"] = 9496,
  ["boxvr"] = 9500,
  ["boxvl"] = 9508,
  ["boxhd"] = 9516,
  ["boxhu"] = 9524,
  ["boxvh"] = 9532,
  ["boxH"] = 9552,
  ["boxV"] = 9553,
  ["boxdR"] = 9554,
  ["boxDr"] = 9555,
  ["boxDR"] = 9556,
  ["boxdL"] = 9557,
  ["boxDl"] = 9558,
  ["boxDL"] = 9559,
  ["boxuR"] = 9560,
  ["boxUr"] = 9561,
  ["boxUR"] = 9562,
  ["boxuL"] = 9563,
  ["boxUl"] = 9564,
  ["boxUL"] = 9565,
  ["boxvR"] = 9566,
  ["boxVr"] = 9567,
  ["boxVR"] = 9568,
  ["boxvL"] = 9569,
  ["boxVl"] = 9570,
  ["boxVL"] = 9571,
  ["boxHd"] = 9572,
  ["boxhD"] = 9573,
  ["boxHD"] = 9574,
  ["boxHu"] = 9575,
  ["boxhU"] = 9576,
  ["boxHU"] = 9577,
  ["boxvH"] = 9578,
  ["boxVh"] = 9579,
  ["boxVH"] = 9580,
  ["uhblk"] = 9600,
  ["lhblk"] = 9604,
  ["block"] = 9608,
  ["blk14"] = 9617,
  ["blk12"] = 9618,
  ["blk34"] = 9619,
  ["squ"] = 9633,
```

```
["square"] = 9633,
["Square"] = 9633,
["squf"] = 9642,
["squarf"] = 9642,
["blacksquare"] = 9642,
["FilledVerySmallSquare"] = 9642,
["EmptyVerySmallSquare"] = 9643,
["rect"] = 9645,
["marker"] = 9646,
["fltns"] = 9649,
["xutri"] = 9651,
["bigtriangleup"] = 9651,
["utrif"] = 9652,
["blacktriangle"] = 9652,
["utri"] = 9653,
["triangle"] = 9653,
["rtrif"] = 9656,
["blacktriangleright"] = 9656,
["rtri"] = 9657,
["triangleright"] = 9657,
["xdtri"] = 9661,
["bigtriangledown"] = 9661,
["dtrif"] = 9662,
["blacktriangledown"] = 9662,
["dtri"] = 9663,
["triangledown"] = 9663,
["ltrif"] = 9666,
["blacktriangleleft"] = 9666,
["ltri"] = 9667,
["triangleleft"] = 9667,
["loz"] = 9674,
["lozenge"] = 9674,
["cir"] = 9675,
["tridot"] = 9708,
["xcirc"] = 9711,
["bigcirc"] = 9711,
["ultri"] = 9720,
["urtri"] = 9721,
["lltri"] = 9722,
["EmptySmallSquare"] = 9723,
["FilledSmallSquare"] = 9724,
["starf"] = 9733,
["bigstar"] = 9733,
["star"] = 9734,
["phone"] = 9742,
["female"] = 9792,
["male"] = 9794,
```

```
2341  ["spades"] = 9824,
2342  ["spadesuit"] = 9824,
2343  ["clubs"] = 9827,
2344  ["clubsuit"] = 9827,
2345  ["hearts"] = 9829,
2346  ["heartsuit"] = 9829,
2347  ["diams"] = 9830,
2348  ["diamondsuit"] = 9830,
2349  ["sung"] = 9834,
2350  ["flat"] = 9837,
2351  ["natur"] = 9838,
2352  ["natural"] = 9838,
2353  ["sharp"] = 9839,
2354  ["check"] = 10003,
2355  ["checkmark"] = 10003,
2356  ["cross"] = 10007,
2357  ["malt"] = 10016,
2358  ["maltese"] = 10016,
2359  ["sext"] = 10038,
2360  ["VerticalSeparator"] = 10072,
2361  ["lbbrk"] = 10098,
2362  ["rbbrk"] = 10099,
2363  ["lobrk"] = 10214,
2364  ["LeftDoubleBracket"] = 10214,
2365  ["robrk"] = 10215,
2366  ["RightDoubleBracket"] = 10215,
2367  ["lang"] = 10216,
2368  ["LeftAngleBracket"] = 10216,
2369  ["langle"] = 10216,
2370  ["rang"] = 10217,
2371  ["RightAngleBracket"] = 10217,
2372  ["rangle"] = 10217,
2373  ["Lang"] = 10218,
2374  ["Rang"] = 10219,
2375  ["loang"] = 10220,
2376  ["roang"] = 10221,
2377  ["xlarr"] = 10229,
2378  ["longleftarrow"] = 10229,
2379  ["LongLeftArrow"] = 10229,
2380  ["xrarr"] = 10230,
2381  ["longrightarrow"] = 10230,
2382  ["LongRightArrow"] = 10230,
2383  ["xharr"] = 10231,
2384  ["longleftrightarrow"] = 10231,
2385  ["LongLeftRightArrow"] = 10231,
2386  ["xlArr"] = 10232,
2387  ["Longleftarrow"] = 10232,
```

```
["DoubleLongLeftArrow"] = 10232,
["xrArr"] = 10233,
["Longrightarrow"] = 10233,
["DoubleLongRightArrow"] = 10233,
["xhArr"] = 10234,
["Longleftrightarrow"] = 10234,
["DoubleLongLeftRightArrow"] = 10234,
["xmap"] = 10236,
["longmapsto"] = 10236,
["dzigrarr"] = 10239,
["nvlArr"] = 10498,
["nvrArr"] = 10499,
["nvHarr"] = 10500,
["Map"] = 10501,
["lbarr"] = 10508,
["rbarr"] = 10509,
["bkarow"] = 10509,
["lBarr"] = 10510,
["rBarr"] = 10511,
["dbkarow"] = 10511,
["RBarr"] = 10512,
["drbkarow"] = 10512,
["DDotrahd"] = 10513,
["UpArrowBar"] = 10514,
["DownArrowBar"] = 10515,
["Rarrtl"] = 10518,
["latail"] = 10521,
["ratail"] = 10522,
["lAtail"] = 10523,
["rAtail"] = 10524,
["larrfs"] = 10525,
["rarrfs"] = 10526,
["larrbfs"] = 10527,
["rarrbfs"] = 10528,
["nwarhk"] = 10531,
["nearhk"] = 10532,
["searhk"] = 10533,
["hksearow"] = 10533,
["swarhk"] = 10534,
["hkswarow"] = 10534,
["nwnear"] = 10535,
["nesear"] = 10536,
["toea"] = 10536,
["seswar"] = 10537,
["tosa"] = 10537,
["swnwar"] = 10538,
["rarrc"] = 10547,
```

```
["cudarrr"] = 10549,
["ldca"] = 10550,
["rdca"] = 10551,
["cudarrl"] = 10552,
["larrpl"] = 10553,
["curarrm"] = 10556,
["cularrp"] = 10557,
["rarrpl"] = 10565,
["harrcir"] = 10568,
["Uarrocir"] = 10569,
["lurdshar"] = 10570,
["ldrushar"] = 10571,
["LeftRightVector"] = 10574,
["RightUpDownVector"] = 10575,
["DownLeftRightVector"] = 10576,
["LeftUpDownVector"] = 10577,
["LeftVectorBar"] = 10578,
["RightVectorBar"] = 10579,
["RightUpVectorBar"] = 10580,
["RightDownVectorBar"] = 10581,
["DownLeftVectorBar"] = 10582,
["DownRightVectorBar"] = 10583,
["LeftUpVectorBar"] = 10584,
["LeftDownVectorBar"] = 10585,
["LeftTeeVector"] = 10586,
["RightTeeVector"] = 10587,
["RightUpTeeVector"] = 10588,
["RightDownTeeVector"] = 10589,
["DownLeftTeeVector"] = 10590,
["DownRightTeeVector"] = 10591,
["LeftUpTeeVector"] = 10592,
["LeftDownTeeVector"] = 10593,
["lHar"] = 10594,
["uHar"] = 10595,
["rHar"] = 10596,
["dHar"] = 10597,
["luruhar"] = 10598,
["ldrdhar"] = 10599,
["ruluhar"] = 10600,
["rdldhar"] = 10601,
["lharul"] = 10602,
["llhard"] = 10603,
["rharul"] = 10604,
["lrhard"] = 10605,
["udhar"] = 10606,
["UpEquilibrium"] = 10606,
["duhar"] = 10607,
```

```
["ReverseUpEquilibrium"] = 10607,
["RoundImplies"] = 10608,
["erarr"] = 10609,
["simrarr"] = 10610,
["larrsim"] = 10611,
["rarrsim"] = 10612,
["rarrap"] = 10613,
["ltlarr"] = 10614,
["gtrarr"] = 10616,
["subrarr"] = 10617,
["suplarr"] = 10619,
["lfisht"] = 10620,
["rfisht"] = 10621,
["ufisht"] = 10622,
["dfisht"] = 10623,
["lopar"] = 10629,
["ropar"] = 10630,
["lbrke"] = 10635,
["rbrke"] = 10636,
["lbrkslu"] = 10637,
["rbrksld"] = 10638,
["lbrksld"] = 10639,
["rbrkslu"] = 10640,
["langd"] = 10641,
["rangd"] = 10642,
["lparlt"] = 10643,
["rpargt"] = 10644,
["gtlPar"] = 10645,
["ltrPar"] = 10646,
["vzigzag"] = 10650,
["vangrt"] = 10652,
["angrtvbd"] = 10653,
["ange"] = 10660,
["range"] = 10661,
["dwangle"] = 10662,
["uwangle"] = 10663,
["angmsdaa"] = 10664,
["angmsdab"] = 10665,
["angmsdac"] = 10666,
["angmsdad"] = 10667,
["angmsdae"] = 10668,
["angmsdaf"] = 10669,
["angmsdag"] = 10670,
["angmsdah"] = 10671,
["bemptyv"] = 10672,
["demptyv"] = 10673,
["cemptyv"] = 10674,
```

```
2529  ["raemptyv"] = 10675,
2530  ["laemptyv"] = 10676,
2531  ["ohbar"] = 10677,
2532  ["omid"] = 10678,
2533  ["opar"] = 10679,
2534  ["operp"] = 10681,
2535  ["olcross"] = 10683,
2536  ["odsold"] = 10684,
2537  ["olcir"] = 10686,
2538  ["ofcir"] = 10687,
2539  ["olt"] = 10688,
2540  ["ogt"] = 10689,
2541  ["cirscir"] = 10690,
2542  ["cirE"] = 10691,
2543  ["solb"] = 10692,
2544  ["bsolb"] = 10693,
2545  ["boxbox"] = 10697,
2546  ["trisb"] = 10701,
2547  ["rtriltri"] = 10702,
2548  ["LeftTriangleBar"] = 10703,
2549  ["RightTriangleBar"] = 10704,
2550  ["race"] = 10714,
2551  ["iinfin"] = 10716,
2552  ["infintie"] = 10717,
2553  ["nvinfin"] = 10718,
2554  ["eparsl"] = 10723,
2555  ["smeparsl"] = 10724,
2556  ["eqvparsl"] = 10725,
2557  ["lozf"] = 10731,
2558  ["blacklozenge"] = 10731,
2559  ["RuleDelayed"] = 10740,
2560  ["dsol"] = 10742,
2561  ["xodot"] = 10752,
2562  ["bigodot"] = 10752,
2563  ["xoplus"] = 10753,
2564  ["bigoplus"] = 10753,
2565  ["xotime"] = 10754,
2566  ["bigotimes"] = 10754,
2567  ["xuplus"] = 10756,
2568  ["biguplus"] = 10756,
2569  ["xsqcup"] = 10758,
2570  ["bigsqcup"] = 10758,
2571  ["qint"] = 10764,
2572  ["iiiint"] = 10764,
2573  ["fpartint"] = 10765,
2574  ["cirfnint"] = 10768,
2575  ["awint"] = 10769,
```

```
["rppolint"] = 10770,
["scpolint"] = 10771,
["npolint"] = 10772,
["pointint"] = 10773,
["quatint"] = 10774,
["intlarhk"] = 10775,
["pluscir"] = 10786,
["plusacir"] = 10787,
["simplus"] = 10788,
["plusdu"] = 10789,
["plussim"] = 10790,
["plustwo"] = 10791,
["mcomma"] = 10793,
["minusdu"] = 10794,
["loplus"] = 10797,
["roplus"] = 10798,
["Cross"] = 10799,
["timesd"] = 10800,
["timesbar"] = 10801,
["smashp"] = 10803,
["lotimes"] = 10804,
["rotimes"] = 10805,
["otimesas"] = 10806,
["Otimes"] = 10807,
["odiv"] = 10808,
["triplus"] = 10809,
["triminus"] = 10810,
["tritime"] = 10811,
["iprod"] = 10812,
["intprod"] = 10812,
["amalg"] = 10815,
["capdot"] = 10816,
["ncup"] = 10818,
["ncap"] = 10819,
["capand"] = 10820,
["cupor"] = 10821,
["cupcap"] = 10822,
["capcup"] = 10823,
["cupbrcap"] = 10824,
["capbrcup"] = 10825,
["cupcup"] = 10826,
["capcap"] = 10827,
["ccups"] = 10828,
["ccaps"] = 10829,
["ccupssm"] = 10832,
["And"] = 10835,
["Or"] = 10836,
```

```
  ["andand"] = 10837,
  ["oror"] = 10838,
  ["orslope"] = 10839,
  ["andslope"] = 10840,
  ["andv"] = 10842,
  ["orv"] = 10843,
  ["andd"] = 10844,
  ["ord"] = 10845,
  ["wedbar"] = 10847,
  ["sdote"] = 10854,
  ["simdot"] = 10858,
  ["congdot"] = 10861,
  ["easter"] = 10862,
  ["apacir"] = 10863,
  ["apE"] = 10864,
  ["eplus"] = 10865,
  ["pluse"] = 10866,
  ["Esim"] = 10867,
  ["Colone"] = 10868,
  ["Equal"] = 10869,
  ["eDDot"] = 10871,
  ["ddotseq"] = 10871,
  ["equivDD"] = 10872,
  ["ltcir"] = 10873,
  ["gtcir"] = 10874,
  ["ltquest"] = 10875,
  ["gtquest"] = 10876,
  ["les"] = 10877,
  ["LessSlantEqual"] = 10877,
  ["leqslant"] = 10877,
  ["ges"] = 10878,
  ["GreaterSlantEqual"] = 10878,
  ["geqslant"] = 10878,
  ["lesdot"] = 10879,
  ["gesdot"] = 10880,
  ["lesdoto"] = 10881,
  ["gesdoto"] = 10882,
  ["lesdotor"] = 10883,
  ["gesdotol"] = 10884,
  ["lap"] = 10885,
  ["lessapprox"] = 10885,
  ["gap"] = 10886,
  ["gtrapprox"] = 10886,
  ["lne"] = 10887,
  ["lneq"] = 10887,
  ["gne"] = 10888,
  ["gneq"] = 10888,
```

```
["lnap"] = 10889,
["lnapprox"] = 10889,
["gnap"] = 10890,
["gnapprox"] = 10890,
["lEg"] = 10891,
["lesseqqgtr"] = 10891,
["gEl"] = 10892,
["gtreqqless"] = 10892,
["lsime"] = 10893,
["gsime"] = 10894,
["lsimg"] = 10895,
["gsiml"] = 10896,
["lgE"] = 10897,
["glE"] = 10898,
["lesges"] = 10899,
["gesles"] = 10900,
["els"] = 10901,
["eqslantless"] = 10901,
["egs"] = 10902,
["eqslantgtr"] = 10902,
["elsdot"] = 10903,
["egsdot"] = 10904,
["el"] = 10905,
["eg"] = 10906,
["siml"] = 10909,
["simg"] = 10910,
["simlE"] = 10911,
["simgE"] = 10912,
["LessLess"] = 10913,
["GreaterGreater"] = 10914,
["glj"] = 10916,
["gla"] = 10917,
["ltcc"] = 10918,
["gtcc"] = 10919,
["lescc"] = 10920,
["gescc"] = 10921,
["smt"] = 10922,
["lat"] = 10923,
["smte"] = 10924,
["late"] = 10925,
["bumpE"] = 10926,
["pre"] = 10927,
["preceq"] = 10927,
["PrecedesEqual"] = 10927,
["sce"] = 10928,
["succeq"] = 10928,
["SucceedsEqual"] = 10928,
```

```
["prE"] = 10931,
["scE"] = 10932,
["prnE"] = 10933,
["precneqq"] = 10933,
["scnE"] = 10934,
["succneqq"] = 10934,
["prap"] = 10935,
["precapprox"] = 10935,
["scap"] = 10936,
["succapprox"] = 10936,
["prnap"] = 10937,
["precnapprox"] = 10937,
["scnap"] = 10938,
["succnapprox"] = 10938,
["Pr"] = 10939,
["Sc"] = 10940,
["subdot"] = 10941,
["supdot"] = 10942,
["subplus"] = 10943,
["supplus"] = 10944,
["submult"] = 10945,
["supmult"] = 10946,
["subedot"] = 10947,
["supedot"] = 10948,
["subE"] = 10949,
["subseteqq"] = 10949,
["supE"] = 10950,
["supseteqq"] = 10950,
["subsim"] = 10951,
["supsim"] = 10952,
["subnE"] = 10955,
["subsetneqq"] = 10955,
["supnE"] = 10956,
["supsetneqq"] = 10956,
["csub"] = 10959,
["csup"] = 10960,
["csube"] = 10961,
["csupe"] = 10962,
["subsup"] = 10963,
["supsub"] = 10964,
["subsub"] = 10965,
["supsup"] = 10966,
["suphsub"] = 10967,
["supdsub"] = 10968,
["forkv"] = 10969,
["topfork"] = 10970,
["mlcp"] = 10971,
```

```
2764  ["Dashv"] = 10980,
2765  ["DoubleLeftTee"] = 10980,
2766  ["Vdashl"] = 10982,
2767  ["Barv"] = 10983,
2768  ["vBar"] = 10984,
2769  ["vBarv"] = 10985,
2770  ["Vbar"] = 10987,
2771  ["Not"] = 10988,
2772  ["bNot"] = 10989,
2773  ["rnmid"] = 10990,
2774  ["cirmid"] = 10991,
2775  ["midcir"] = 10992,
2776  ["topcir"] = 10993,
2777  ["nhpar"] = 10994,
2778  ["parsim"] = 10995,
2779  ["parsl"] = 11005,
2780  ["fflig"] = 64256,
2781  ["filig"] = 64257,
2782  ["fllig"] = 64258,
2783  ["ffilig"] = 64259,
2784  ["ffllig"] = 64260,
2785  ["Ascr"] = 119964,
2786  ["Cscr"] = 119966,
2787  ["Dscr"] = 119967,
2788  ["Gscr"] = 119970,
2789  ["Jscr"] = 119973,
2790  ["Kscr"] = 119974,
2791  ["Nscr"] = 119977,
2792  ["Oscr"] = 119978,
2793  ["Pscr"] = 119979,
2794  ["Qscr"] = 119980,
2795  ["Sscr"] = 119982,
2796  ["Tscr"] = 119983,
2797  ["Uscr"] = 119984,
2798  ["Vscr"] = 119985,
2799  ["Wscr"] = 119986,
2800  ["Xscr"] = 119987,
2801  ["Yscr"] = 119988,
2802  ["Zscr"] = 119989,
2803  ["ascr"] = 119990,
2804  ["bscr"] = 119991,
2805  ["cscr"] = 119992,
2806  ["dscr"] = 119993,
2807  ["fscr"] = 119995,
2808  ["hscr"] = 119997,
2809  ["iscr"] = 119998,
2810  ["jscr"] = 119999,
```

```
  ["kscr"] = 120000,
  ["lscr"] = 120001,
  ["mscr"] = 120002,
  ["nscr"] = 120003,
  ["pscr"] = 120005,
  ["qscr"] = 120006,
  ["rscr"] = 120007,
  ["sscr"] = 120008,
  ["tscr"] = 120009,
  ["uscr"] = 120010,
  ["vscr"] = 120011,
  ["wscr"] = 120012,
  ["xscr"] = 120013,
  ["yscr"] = 120014,
  ["zscr"] = 120015,
  ["Afr"] = 120068,
  ["Bfr"] = 120069,
  ["Dfr"] = 120071,
  ["Efr"] = 120072,
  ["Ffr"] = 120073,
  ["Gfr"] = 120074,
  ["Jfr"] = 120077,
  ["Kfr"] = 120078,
  ["Lfr"] = 120079,
  ["Mfr"] = 120080,
  ["Nfr"] = 120081,
  ["Ofr"] = 120082,
  ["Pfr"] = 120083,
  ["Qfr"] = 120084,
  ["Sfr"] = 120086,
  ["Tfr"] = 120087,
  ["Ufr"] = 120088,
  ["Vfr"] = 120089,
  ["Wfr"] = 120090,
  ["Xfr"] = 120091,
  ["Yfr"] = 120092,
  ["afr"] = 120094,
  ["bfr"] = 120095,
  ["cfr"] = 120096,
  ["dfr"] = 120097,
  ["efr"] = 120098,
  ["ffr"] = 120099,
  ["gfr"] = 120100,
  ["hfr"] = 120101,
  ["ifr"] = 120102,
  ["jfr"] = 120103,
  ["kfr"] = 120104,
```

```
  ["lfr"] = 120105,
  ["mfr"] = 120106,
  ["nfr"] = 120107,
  ["ofr"] = 120108,
  ["pfr"] = 120109,
  ["qfr"] = 120110,
  ["rfr"] = 120111,
  ["sfr"] = 120112,
  ["tfr"] = 120113,
  ["ufr"] = 120114,
  ["vfr"] = 120115,
  ["wfr"] = 120116,
  ["xfr"] = 120117,
  ["yfr"] = 120118,
  ["zfr"] = 120119,
  ["Aopf"] = 120120,
  ["Bopf"] = 120121,
  ["Dopf"] = 120123,
  ["Eopf"] = 120124,
  ["Fopf"] = 120125,
  ["Gopf"] = 120126,
  ["Iopf"] = 120128,
  ["Jopf"] = 120129,
  ["Kopf"] = 120130,
  ["Lopf"] = 120131,
  ["Mopf"] = 120132,
  ["Oopf"] = 120134,
  ["Sopf"] = 120138,
  ["Topf"] = 120139,
  ["Uopf"] = 120140,
  ["Vopf"] = 120141,
  ["Wopf"] = 120142,
  ["Xopf"] = 120143,
  ["Yopf"] = 120144,
  ["aopf"] = 120146,
  ["bopf"] = 120147,
  ["copf"] = 120148,
  ["dopf"] = 120149,
  ["eopf"] = 120150,
  ["fopf"] = 120151,
  ["gopf"] = 120152,
  ["hopf"] = 120153,
  ["iopf"] = 120154,
  ["jopf"] = 120155,
  ["kopf"] = 120156,
  ["lopf"] = 120157,
  ["mopf"] = 120158,
```

```
  ["nopf"] = 120159,
  ["oopf"] = 120160,
  ["popf"] = 120161,
  ["qopf"] = 120162,
  ["ropf"] = 120163,
  ["sopf"] = 120164,
  ["topf"] = 120165,
  ["uopf"] = 120166,
  ["vopf"] = 120167,
  ["wopf"] = 120168,
  ["xopf"] = 120169,
  ["yopf"] = 120170,
  ["zopf"] = 120171,
}
```

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

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

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

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

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

```
function entities.char_entity(s)
  local n = character_entities[s]
  if n == nil then
    return "&" .. s .. ";"
  end
  return unicode.utf8.char(n)
end
```

3.1.3 Plain TEX Writer

This section documents the `writer` object, which implements the routines for producing the TEX output. The object is an amalgamate of the generic, TEX, LATEX writer objects that were located in the `lunamark/writer/generic.lua`, `lunamark/writer/tex.lua`, and `lunamark/writer/latex.lua` files in the Lunamark 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.

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

The `writer.new` method creates and returns a new TEX 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*⟩. All member variables are immutable unless explicitly stated otherwise.

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

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

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

Parse the `slice` option and define `writer->slice_begin` `writer->slice_end`, and `writer->is_writing`. The `writer->is_writing` member variable is mutable.

```
  local slice_specifiers = {}
  for specifier in options.slice:gmatch("[^%s]+") do
    table.insert(slice_specifiers, specifier)
  end

  if #slice_specifiers == 2 then
    self.slice_begin, self.slice_end = table.unpack(slice_specifiers)
    local slice_begin_type = self.slice_begin:sub(1, 1)
    if slice_begin_type ~= "^" and slice_begin_type ~= "$" then
      self.slice_begin = "^" .. self.slice_begin
    end
    local slice_end_type = self.slice_end:sub(1, 1)
    if slice_end_type ~= "^" and slice_end_type ~= "$" then
      self.slice_end = "$" .. self.slice_end
    end
  elseif #slice_specifiers == 1 then
    self.slice_begin = "^" .. slice_specifiers[1]
    self.slice_end = "$" .. slice_specifiers[1]
  end

  if self.slice_begin == "^" and self.slice_end ~= "^" then
    self.is_writing = true
  else
    self.is_writing = false
  end
```

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

```
2963  self.suffix = ".tex"
```

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

```
2964  self.space = " "
```

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

```
2965  self.nbsp = "\\markdownRendererNbsp{}"
```

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

```
2966  function self.plain(s)
2967    return s
2968  end
```

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

```
2969  function self.paragraph(s)
2970    if not self.is_writing then return "" end
2971    return s
2972  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.

```
2973  function self.pack(name)
2974    return [[\input ]] .. name .. [[\relax{}]]
2975  end
```

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

```
2976  function self.interblocksep()
2977    if not self.is_writing then return "" end
2978    return "\\markdownRendererInterblockSeparator\n{}"
2979  end
```

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

```
2980  self.eof = [[\relax]]
```

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

```
2981  self.linebreak = "\\markdownRendererLineBreak\n{}"
```

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

```
2982  self.ellipsis = "\\markdownRendererEllipsis{}"
```

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

```
2983  function self.hrule()
2984    if not self.is_writing then return "" end
2985    return "\\markdownRendererHorizontalRule{}"
2986  end
```

Define a table `escaped_chars` containing the mapping from special plain TEX characters (including the active pipe character (|) of ConTEXt) to their escaped variants. Define tables `escaped_uri_chars`, `escaped_citation_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.

```
local escaped_chars = {
   ["{"] = "\\markdownRendererLeftBrace{}",
   ["}"] = "\\markdownRendererRightBrace{}",
   ["$"] = "\\markdownRendererDollarSign{}",
   ["%"] = "\\markdownRendererPercentSign{}",
   ["&"] = "\\markdownRendererAmpersand{}",
   ["_"] = "\\markdownRendererUnderscore{}",
   ["#"] = "\\markdownRendererHash{}",
   ["^"] = "\\markdownRendererCircumflex{}",
   ["\\"] = "\\markdownRendererBackslash{}",
   ["~"] = "\\markdownRendererTilde{}",
   ["|"] = "\\markdownRendererPipe{}",
 }
 local escaped_uri_chars = {
   ["{"] = "\\markdownRendererLeftBrace{}",
   ["}"] = "\\markdownRendererRightBrace{}",
   ["%"] = "\\markdownRendererPercentSign{}",
   ["\\"] = "\\markdownRendererBackslash{}",
 }
 local escaped_citation_chars = {
   ["{"] = "\\markdownRendererLeftBrace{}",
   ["}"] = "\\markdownRendererRightBrace{}",
   ["%"] = "\\markdownRendererPercentSign{}",
   ["#"] = "\\markdownRendererHash{}",
   ["\\"] = "\\markdownRendererBackslash{}",
 }
 local escaped_minimal_strings = {
   ["^^"] = "\\markdownRendererCircumflex\\markdownRendererCircumflex ",
 }
```

Use the `escaped_chars`, `escaped_uri_chars`, `escaped_citation_chars`, and `escaped_minimal_strings` tables to create the `escape`, `escape_citation`, and `escape_uri` escaper functions.

```
local escape = util.escaper(escaped_chars)
local escape_citation = util.escaper(escaped_citation_chars,
  escaped_minimal_strings)
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, `writer->citation` as a function that will transform an input citation name `c` 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 enabled, use

identity functions. Otherwise, use the `escape`, `escape_citation`, and `escape_uri` functions.

```
3020  if options.hybrid then
3021    self.string = function(s) return s end
3022    self.citation = function(c) return c end
3023    self.uri = function(u) return u end
3024  else
3025    self.string = escape
3026    self.citation = escape_citation
3027    self.uri = escape_uri
3028  end
```

Define `writer->escape` as a function that will transform an input plain text span to the output format. Unlike the `writer->string` function, `writer->escape` always uses the `escape` function, even when the `hybrid` option is enabled.

```
3029  self.escape = escape
```

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

```
3030  function self.code(s)
3031    return {"\\markdownRendererCodeSpan{",self.escape(s),"}"}
3032  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.

```
3033  function self.link(lab,src,tit)
3034    return {"\\markdownRendererLink{",lab,"}",
3035                            "{",self.escape(src),"}",
3036                            "{",self.uri(src),"}",
3037                            "{",self.string(tit or ""),"}"}
3038  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.

```
3039  function self.table(rows, caption)
3040    if not self.is_writing then return "" end
3041    local buffer = {"\\markdownRendererTable{",
3042      caption or "", "}{", #rows - 1, "}{", #rows[1], "}"}
3043    local temp = rows[2] -- put alignments on the first row
3044    rows[2] = rows[1]
3045    rows[1] = temp
3046    for i, row in ipairs(rows) do
3047      table.insert(buffer, "{")
3048      for _, column in ipairs(row) do
3049        if i > 1 then -- do not use braces for alignments
```

```
          table.insert(buffer, "{")
        end
        table.insert(buffer, column)
        if i > 1 then
          table.insert(buffer, "}")
        end
      end
      table.insert(buffer, "}")
    end
    return buffer
  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.

```
  function self.image(lab,src,tit)
    return {"\\markdownRendererImage{",lab,"}",
                            "{",self.string(src),"}",
                            "{",self.uri(src),"}",
                            "{",self.string(tit or ""),"}"}
  end
```

The `languages_json` table maps programming language filename extensions to fence infostrings. All `options.contentBlocksLanguageMap` files located by the KPathSea library are loaded into a chain of tables. `languages_json` corresponds to the first table and is chained with the rest via Lua metatables.

```
  local languages_json = (function()
    local ran_ok, kpse = pcall(require, "kpse")
    if ran_ok then
      kpse.set_program_name("luatex")
```

If the KPathSea library is unavailable, perhaps because we are using LuaMetaTEX, we will only locate the `options.contentBlocksLanguageMap` in the current working directory:

```
    else
      kpse = {lookup=function(filename, options) return filename end}
    end
    local base, prev, curr
    for _, filename in ipairs{kpse.lookup(options.contentBlocksLanguageMap,
                                          { all=true })} do
      local file = io.open(filename, "r")
      if not file then goto continue end
      json = file:read("*all"):gsub('("[^\n]-"):','[%1]=')
      curr = (function()
        local _ENV={ json=json, load=load } -- run in sandbox
        return load("return "..json)()
      end)()
```

```
      if type(curr) == "table" then
        if base == nil then
          base = curr
        else
          setmetatable(prev, { __index = curr })
        end
        prev = curr
      end
      ::continue::
    end
    return base or {}
  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.

```
  function self.contentblock(src,suf,type,tit)
    if not self.is_writing then return "" end
    src = src.."."..suf
    suf = suf:lower()
    if type == "onlineimage" then
      return {"\\markdownRendererContentBlockOnlineImage{",suf,"}",
                              "{",self.string(src),"}",
                              "{",self.uri(src),"}",
                              "{",self.string(tit or ""),"}"}
    elseif languages_json[suf] then
      return {"\\markdownRendererContentBlockCode{",suf,"}",
                              "{",self.string(languages_json[suf]),"}",
                              "{",self.string(src),"}",
                              "{",self.uri(src),"}",
                              "{",self.string(tit or ""),"}"}
    else
      return {"\\markdownRendererContentBlock{",suf,"}",
                              "{",self.string(src),"}",
                              "{",self.uri(src),"}",
                              "{",self.string(tit or ""),"}"}
    end
  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.

```
  local function ulitem(s)
    return {"\\markdownRendererUlItem ",s,
            "\\markdownRendererUlItemEnd "}
  end
```

```
3122
3123  function self.bulletlist(items,tight)
3124    if not self.is_writing then return "" end
3125    local buffer = {}
3126    for _,item in ipairs(items) do
3127      buffer[#buffer + 1] = ulitem(item)
3128    end
3129    local contents = util.intersperse(buffer,"\n")
3130    if tight and options.tightLists then
3131      return {"\\markdownRendererUlBeginTight\n",contents,
3132        "\n\\markdownRendererUlEndTight "}
3133    else
3134      return {"\\markdownRendererUlBegin\n",contents,
3135        "\n\\markdownRendererUlEnd "}
3136    end
3137  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.

```
3138  local function olitem(s,num)
3139    if num ~= nil then
3140      return {"\\markdownRendererOlItemWithNumber{",num,"}",s,
3141              "\\markdownRendererOlItemEnd "}
3142    else
3143      return {"\\markdownRendererOlItem ",s,
3144              "\\markdownRendererOlItemEnd "}
3145    end
3146  end
3147
3148  function self.orderedlist(items,tight,startnum)
3149    if not self.is_writing then return "" end
3150    local buffer = {}
3151    local num = startnum
3152    for _,item in ipairs(items) do
3153      buffer[#buffer + 1] = olitem(item,num)
3154      if num ~= nil then
3155        num = num + 1
3156      end
3157    end
3158    local contents = util.intersperse(buffer,"\n")
3159    if tight and options.tightLists then
3160      return {"\\markdownRendererOlBeginTight\n",contents,
3161        "\n\\markdownRendererOlEndTight "}
3162    else
3163      return {"\\markdownRendererOlBegin\n",contents,
```

```
        "\n\\markdownRendererOlEnd "}
    end
  end
```

Define `writer->inline_html_comment` as a function that will transform the contents of an inline HTML comment, to the output format, where `contents` are the contents of the HTML comment.

```
  function self.inline_html_comment(contents)
    return {"\\markdownRendererInlineHtmlComment{",contents,"}"}
  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.

```
  local function dlitem(term, defs)
    local retVal = {"\\markdownRendererDlItem{",term,"}"}
    for _, def in ipairs(defs) do
      retVal[#retVal+1] = {"\\markdownRendererDlDefinitionBegin ",def,
                           "\\markdownRendererDlDefinitionEnd "}
    end
    retVal[#retVal+1] = "\\markdownRendererDlItemEnd "
    return retVal
  end

  function self.definitionlist(items,tight)
    if not self.is_writing then return "" end
    local buffer = {}
    for _,item in ipairs(items) do
      buffer[#buffer + 1] = dlitem(item.term, item.definitions)
    end
    if tight and options.tightLists then
      return {"\\markdownRendererDlBeginTight\n", buffer,
        "\n\\markdownRendererDlEndTight"}
    else
      return {"\\markdownRendererDlBegin\n", buffer,
        "\n\\markdownRendererDlEnd"}
    end
  end
```

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

```
  function self.emphasis(s)
    return {"\\markdownRendererEmphasis{",s,"}"}
  end
```

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

```
3197   function self.strong(s)
3198     return {"\\markdownRendererStrongEmphasis{",s,"}"}
3199   end
```

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

```
3200   function self.blockquote(s)
3201     if #util.rope_to_string(s) == 0 then return "" end
3202     return {"\\markdownRendererBlockQuoteBegin\n",s,
3203       "\n\\markdownRendererBlockQuoteEnd "}
3204   end
```

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

```
3205   function self.verbatim(s)
3206     if not self.is_writing then return "" end
3207     s = string.gsub(s, '[\r\n%s]*$', '')
3208     local name = util.cache(options.cacheDir, s, nil, nil, ".verbatim")
3209     return {"\\markdownRendererInputVerbatim{",name,"}"}
3210   end
```

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

```
3211   function self.fencedCode(i, s)
3212     if not self.is_writing then return "" end
3213     s = string.gsub(s, '[\r\n%s]*$', '')
3214     local name = util.cache(options.cacheDir, s, nil, nil, ".verbatim")
3215     return {"\\markdownRendererInputFencedCode{",name,"}{",i,"}"}
3216   end
```

Define `writer->active_headings` as a stack of identifiers of the headings that are currently active. The `writer->active_headings` member variable is mutable.

```
3217   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.

```
3218   function self.heading(s,level,attributes)
3219     local active_headings = self.active_headings
3220     local slice_begin_type = self.slice_begin:sub(1, 1)
3221     local slice_begin_identifier = self.slice_begin:sub(2) or ""
3222     local slice_end_type = self.slice_end:sub(1, 1)
3223     local slice_end_identifier = self.slice_end:sub(2) or ""
3224
3225     while #active_headings < level do
3226       -- push empty identifiers for implied sections
3227       table.insert(active_headings, {})
3228     end
3229
```

```
  while #active_headings >= level do
    -- pop identifiers for sections that have ended
    local active_identifiers = active_headings[#active_headings]
    if active_identifiers[slice_begin_identifier] ~= nil
        and slice_begin_type == "$" then
      self.is_writing = true
    end
    if active_identifiers[slice_end_identifier] ~= nil
        and slice_end_type == "$" then
      self.is_writing = false
    end
    table.remove(active_headings, #active_headings)
  end

  -- push identifiers for the new section
  attributes = attributes or {}
  local identifiers = {}
  for index = 1, #attributes do
    attribute = attributes[index]
    identifiers[attribute:sub(2)] = true
  end
  if identifiers[slice_begin_identifier] ~= nil
      and slice_begin_type == "^" then
    self.is_writing = true
  end
  if identifiers[slice_end_identifier] ~= nil
      and slice_end_type == "^" then
    self.is_writing = false
  end
  table.insert(active_headings, identifiers)

  if not self.is_writing then return "" end

  local cmd
  level = level + options.shiftHeadings
  if level <= 1 then
    cmd = "\\markdownRendererHeadingOne"
  elseif level == 2 then
    cmd = "\\markdownRendererHeadingTwo"
  elseif level == 3 then
    cmd = "\\markdownRendererHeadingThree"
  elseif level == 4 then
    cmd = "\\markdownRendererHeadingFour"
  elseif level == 5 then
    cmd = "\\markdownRendererHeadingFive"
  elseif level >= 6 then
    cmd = "\\markdownRendererHeadingSix"
```

```
3277      else
3278        cmd = ""
3279      end
3280      return {cmd,"{",s,"}"}
3281    end
```

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

```
3282    function self.note(s)
3283      return {"\\markdownRendererFootnote{",s,"}"}
3284    end
```

Define `writer->citations` as a function that will transform an input array of citations `cites` to the output format. If `text_cites` is enabled, 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.

```
3285    function self.citations(text_cites, cites)
3286      local buffer = {"\\markdownRenderer", text_cites and "TextCite" or "Cite",
3287        "{", #cites, "}"}
3288      for _,cite in ipairs(cites) do
3289        buffer[#buffer+1] = {cite.suppress_author and "-" or "+", "{",
3290          cite.prenote or "", "}{", cite.postnote or "", "}{", cite.name, "}"}
3291      end
3292      return buffer
3293    end
```

Define `writer->get_state` as a function that returns the current state of the writer, where the state of a writer are its mutable member variables.

```
3294    function self.get_state()
3295      return {
3296        is_writing=self.is_writing,
3297        active_headings={table.unpack(self.active_headings)},
3298      }
3299    end
```

Define `writer->set_state` as a function that restores the input state `s` and returns the previous state of the writer.

```
  function self.set_state(s)
    previous_state = self.get_state()
    for key, value in pairs(state) do
      self[key] = value
    end
    return previous_state
  end
```

Define `writer->defer_call` as a function that will encapsulate the input function `f`, so that `f` is called with the state of the writer at the time of calling `writer->defer_call`.

```
  function self.defer_call(f)
    state = self.get_state()
    return function(...)
      state = self.set_state(state)
      local return_value = f(...)
      self.set_state(state)
      return return_value
    end
  end

  return self
end
```

3.1.4 Parsers

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

```
local parsers                 = {}
```

3.1.4.1 Basic Parsers

```
parsers.percent               = P("%")
parsers.at                    = P("@")
parsers.comma                 = P(",")
parsers.asterisk              = P("*")
parsers.dash                  = P("-")
parsers.plus                  = P("+")
parsers.underscore            = P("_")
parsers.period                = P(".")
parsers.hash                  = P("#")
parsers.ampersand             = P("&")
parsers.backtick              = P("`")
parsers.less                  = P("<")
parsers.more                  = P(">")
parsers.space                 = P(" ")
parsers.squote                = P("'")
```

```
parsers.dquote                = P('"')
parsers.lparent               = P("(")
parsers.rparent               = P(")")
parsers.lbracket              = P("[")
parsers.rbracket              = P("]")
parsers.lbrace                = P("{")
parsers.rbrace                = P("}")
parsers.circumflex            = P("^")
parsers.slash                 = P("/")
parsers.equal                 = P("=")
parsers.colon                 = P(":")
parsers.semicolon             = P(";")
parsers.exclamation           = P("!")
parsers.pipe                  = P("|")
parsers.tilde                 = P("~")
parsers.backslash             = P("\\")
parsers.tab                   = P("\t")
parsers.newline               = P("\n")
parsers.tightblocksep         = P("\001")

parsers.digit                 = R("09")
parsers.hexdigit              = R("09","af","AF")
parsers.letter                = R("AZ","az")
parsers.alphanumeric          = R("AZ","az","09")
parsers.keyword               = parsers.letter
                              * parsers.alphanumeric^0
parsers.citation_chars        = parsers.alphanumeric
                              + S("#$%&-+<>~/_")
parsers.internal_punctuation  = S(":;,.?")

parsers.doubleasterisks       = P("**")
parsers.doubleunderscores     = P("__")
parsers.fourspaces            = P("    ")

parsers.any                   = P(1)
parsers.fail                  = parsers.any - 1

parsers.escapable             = S("\\`*_{}[]()+_.!<>#-~:^@;")
parsers.anyescaped            = parsers.backslash / "" * parsers.escapable
                              + parsers.any

parsers.spacechar             = S("\t ")
parsers.spacing               = S(" \n\r\t")
parsers.nonspacechar          = parsers.any - parsers.spacing
parsers.optionalspace         = parsers.spacechar^0

parsers.specialchar           = S("*_`&[]<!\\.@-^")
```

```
3382
3383 parsers.normalchar             = parsers.any - (parsers.specialchar
3384                                               + parsers.spacing
3385                                               + parsers.tightblocksep)
3386 parsers.eof                    = -parsers.any
3387 parsers.nonindentspace         = parsers.space^-3 * - parsers.spacechar
3388 parsers.indent                 = parsers.space^-3 * parsers.tab
3389                                + parsers.fourspaces / ""
3390 parsers.linechar               = P(1 - parsers.newline)
3391
3392 parsers.blankline              = parsers.optionalspace
3393                                * parsers.newline / "\n"
3394 parsers.blanklines             = parsers.blankline^0
3395 parsers.skipblanklines         = (parsers.optionalspace * parsers.newline)^0
3396 parsers.indentedline           = parsers.indent    /""
3397                                * C(parsers.linechar^1 * parsers.newline^-
    1)
3398 parsers.optionallyindentedline = parsers.indent^-1 /""
3399                                * C(parsers.linechar^1 * parsers.newline^-
    1)
3400 parsers.sp                     = parsers.spacing^0
3401 parsers.spnl                   = parsers.optionalspace
3402                                * (parsers.newline * parsers.optionalspace)^-
    1
3403 parsers.line                   = parsers.linechar^0 * parsers.newline
3404 parsers.nonemptyline           = parsers.line - parsers.blankline
3405
3406 parsers.commented_line         = Cs(((parsers.linechar  -- initial
3407                                     - parsers.backslash
3408                                     - parsers.percent)^1
3409                                   + (parsers.backslash  -- even backslash
3410                                     * parsers.backslash)^1
3411                                   + (parsers.backslash  -- odd backslash
3412                                     * parsers.backslash)^0
3413                                   * (parsers.backslash / ""
3414                                     * parsers.percent
3415                                     + parsers.backslash
3416                                     * (parsers.linechar -- initial
3417                                       + parsers.newline
3418                                       - parsers.backslash
3419                                       - parsers.percent))
3420                                )^0)
3421                                * ((parsers.percent  -- comment
3422                                   * parsers.line
3423                                   * #parsers.blankline) -- blank line
3424                                   / "\n"
3425                                  + parsers.percent  -- comment
```

```
                                    * parsers.line
                                    * parsers.optionalspace  -- leading tabs and spaces
                                    + C(parsers.newline))

parsers.chunk                  = parsers.line * (parsers.optionallyindentedline
                                                 - parsers.blankline)^0

parsers.css_identifier_char    = R("AZ", "az", "09") + S("-_")
parsers.css_identifier         = (parsers.hash + parsers.period)
                               * (((parsers.css_identifier_char
                                  - parsers.dash - parsers.digit)
                                 * parsers.css_identifier_char^1)
                                + (parsers.dash
                                  * (parsers.css_identifier_char
                                    - parsers.digit)
                                  * parsers.css_identifier_char^0))
parsers.attribute_name_char    = parsers.any - parsers.space
                               - parsers.squote - parsers.dquote
                               - parsers.more - parsers.slash
                               - parsers.equal
parsers.attribute_value_char   = parsers.any - parsers.dquote
                               - parsers.more

-- block followed by 0 or more optionally
-- indented blocks with first line indented.
parsers.indented_blocks = function(bl)
  return Cs( bl
         * (parsers.blankline^1 * parsers.indent * -parsers.blankline * bl)^0
         * (parsers.blankline^1 + parsers.eof) )
end
```

3.1.4.2 Parsers Used for Markdown Lists

```
parsers.bulletchar = C(parsers.plus + parsers.asterisk + parsers.dash)

parsers.bullet = ( parsers.bulletchar * #parsers.spacing
                                      * (parsers.tab + parsers.space^-3)
                 + parsers.space * parsers.bulletchar * #parsers.spacing
                                 * (parsers.tab + parsers.space^-2)
                 + parsers.space * parsers.space * parsers.bulletchar
                                 * #parsers.spacing
                                 * (parsers.tab + parsers.space^-1)
                 + parsers.space * parsers.space * parsers.space
                                 * parsers.bulletchar * #parsers.spacing
                 )
```

3.1.4.3 Parsers Used for Markdown Code Spans

```
parsers.openticks   = Cg(parsers.backtick^1, "ticks")

local function captures_equal_length(s,i,a,b)
  return #a == #b and i
end

parsers.closeticks  = parsers.space^-1
                    * Cmt(C(parsers.backtick^1)
                        * Cb("ticks"), captures_equal_length)

parsers.intickschar = (parsers.any - S(" \n\r`"))
                    + (parsers.newline * -parsers.blankline)
                    + (parsers.space - parsers.closeticks)
                    + (parsers.backtick^1 - parsers.closeticks)

parsers.inticks     = parsers.openticks * parsers.space^-1
                    * C(parsers.intickschar^0) * parsers.closeticks
```

3.1.4.4 Parsers Used for Fenced Code Blocks

```
local function captures_geq_length(s,i,a,b)
  return #a >= #b and i
end

parsers.infostring     = (parsers.linechar - (parsers.backtick
                       + parsers.space^1 * (parsers.newline + parsers.eof)))^0

local fenceindent
parsers.fencehead    = function(char)
  return               C(parsers.nonindentspace) / function(s) fenceindent = #s end
                     * Cg(char^3, "fencelength")
                     * parsers.optionalspace * C(parsers.infostring)
                     * parsers.optionalspace * (parsers.newline + parsers.eof)
end

parsers.fencetail    = function(char)
  return               parsers.nonindentspace
                     * Cmt(C(char^3) * Cb("fencelength"), captures_geq_length)
                     * parsers.optionalspace * (parsers.newline + parsers.eof)
                     + parsers.eof
end

parsers.fencedline   = function(char)
  return               C(parsers.line - parsers.fencetail(char))
                     / function(s)
                         i = 1
                         remaining = fenceindent
```

```
                          while true do
                            c = s:sub(i, i)
                            if c == " " and remaining > 0 then
                              remaining = remaining - 1
                              i = i + 1
                            elseif c == "\t" and remaining > 3 then
                              remaining = remaining - 4
                              i = i + 1
                            else
                              break
                            end
                          end
                          return s:sub(i)
                        end
end
```

3.1.4.5 Parsers Used for Markdown Tags and Links

```
parsers.leader      = parsers.space^-3

-- content in balanced brackets, parentheses, or quotes:
parsers.bracketed   = P{ parsers.lbracket
                         * ((parsers.anyescaped - (parsers.lbracket
                                                 + parsers.rbracket
                                                 + parsers.blankline^2)
                            ) + V(1))^0
                         * parsers.rbracket }

parsers.inparens    = P{ parsers.lparent
                         * ((parsers.anyescaped - (parsers.lparent
                                                 + parsers.rparent
                                                 + parsers.blankline^2)
                            ) + V(1))^0
                         * parsers.rparent }

parsers.squoted     = P{ parsers.squote * parsers.alphanumeric
                         * ((parsers.anyescaped - (parsers.squote
                                                 + parsers.blankline^2)
                            ) + V(1))^0
                         * parsers.squote }

parsers.dquoted     = P{ parsers.dquote * parsers.alphanumeric
                         * ((parsers.anyescaped - (parsers.dquote
                                                 + parsers.blankline^2)
                            ) + V(1))^0
                         * parsers.dquote }

```

```
-- bracketed tag for markdown links, allowing nested brackets:
parsers.tag         = parsers.lbracket
                    * Cs((parsers.alphanumeric^1
                         + parsers.bracketed
                         + parsers.inticks
                         + (parsers.anyescaped
                           - (parsers.rbracket + parsers.blankline^2)))^0)
                    * parsers.rbracket

-- url for markdown links, allowing nested brackets:
parsers.url         = parsers.less * Cs((parsers.anyescaped
                                       - parsers.more)^0)
                                   * parsers.more
                    + Cs((parsers.inparens + (parsers.anyescaped
                                            - parsers.spacing
                                            - parsers.rparent))^1)

-- quoted text, possibly with nested quotes:
parsers.title_s     = parsers.squote * Cs(((parsers.anyescaped-parsers.squote)
                                         + parsers.squoted)^0)
                                     * parsers.squote

parsers.title_d     = parsers.dquote * Cs(((parsers.anyescaped-parsers.dquote)
                                         + parsers.dquoted)^0)
                                     * parsers.dquote

parsers.title_p     = parsers.lparent
                    * Cs((parsers.inparens + (parsers.anyescaped-parsers.rparent))^0)
                    * parsers.rparent

parsers.title       = parsers.title_d + parsers.title_s + parsers.title_p

parsers.optionaltitle
                    = parsers.spnl * parsers.title * parsers.spacechar^0
                    + Cc("")
```

3.1.4.6 Parsers Used for iA Writer Content Blocks

```
parsers.contentblock_tail
                    = parsers.optionaltitle
                    * (parsers.newline + parsers.eof)

-- case insensitive online image suffix:
parsers.onlineimagesuffix
                    = (function(...)
                        local parser = nil
                        for _,suffix in ipairs({...}) do
```

```
                            local pattern=nil
                            for i=1,#suffix do
                              local char=suffix:sub(i,i)
                              char = S(char:lower()..char:upper())
                              if pattern == nil then
                                pattern = char
                              else
                                pattern = pattern * char
                              end
                            end
                            if parser == nil then
                              parser = pattern
                            else
                              parser = parser + pattern
                            end
                          end
                          return parser
                        end)("png", "jpg", "jpeg", "gif", "tif", "tiff")

-- online image url for iA Writer content blocks with mandatory suffix,
-- allowing nested brackets:
parsers.onlineimageurl
                      = (parsers.less
                        * Cs((parsers.anyescaped
                             - parsers.more
                             - #(parsers.period
                                * parsers.onlineimagesuffix
                                * parsers.more
                                * parsers.contentblock_tail))^0)
                        * parsers.period
                        * Cs(parsers.onlineimagesuffix)
                        * parsers.more
                        + (Cs((parsers.inparens
                              + (parsers.anyescaped
                                - parsers.spacing
                                - parsers.rparent
                                - #(parsers.period
                                   * parsers.onlineimagesuffix
                                   * parsers.contentblock_tail)))^0)
                          * parsers.period
                          * Cs(parsers.onlineimagesuffix))
                        ) * Cc("onlineimage")

-- filename for iA Writer content blocks with mandatory suffix:
parsers.localfilepath
                      = parsers.slash
                      * Cs((parsers.anyescaped
```

```
                          - parsers.tab
                          - parsers.newline
                          - #(parsers.period
                             * parsers.alphanumeric^1
                             * parsers.contentblock_tail))^1)
                     * parsers.period
                     * Cs(parsers.alphanumeric^1)
                     * Cc("localfile")
```

3.1.4.7 Parsers Used for Citations

```
parsers.citation_name = Cs(parsers.dash^-1) * parsers.at
                      * Cs(parsers.citation_chars
                          * (((parsers.citation_chars + parsers.internal_punctuation
                              - parsers.comma - parsers.semicolon)
                             * -#((parsers.internal_punctuation - parsers.comma
                                  - parsers.semicolon)^0
                                 * -(parsers.citation_chars + parsers.internal_punctuati
                                    - parsers.comma - parsers.semicolon)))^0
                            * parsers.citation_chars)^-1)

parsers.citation_body_prenote
                     = Cs((parsers.alphanumeric^1
                          + parsers.bracketed
                          + parsers.inticks
                          + (parsers.anyescaped
                            - (parsers.rbracket + parsers.blankline^2))
                          - (parsers.spnl * parsers.dash^-1 * parsers.at))^0)

parsers.citation_body_postnote
                     = Cs((parsers.alphanumeric^1
                          + parsers.bracketed
                          + parsers.inticks
                          + (parsers.anyescaped
                            - (parsers.rbracket + parsers.semicolon
                              + parsers.blankline^2))
                          - (parsers.spnl * parsers.rbracket))^0)

parsers.citation_body_chunk
                     = parsers.citation_body_prenote
                     * parsers.spnl * parsers.citation_name
                     * (parsers.internal_punctuation - parsers.semicolon)^-
1
                     * parsers.spnl * parsers.citation_body_postnote

parsers.citation_body
                     = parsers.citation_body_chunk
```

```
                    * (parsers.semicolon * parsers.spnl
                      * parsers.citation_body_chunk)^0

parsers.citation_headless_body_postnote
                    = Cs((parsers.alphanumeric^1
                         + parsers.bracketed
                         + parsers.inticks
                         + (parsers.anyescaped
                           - (parsers.rbracket + parsers.at
                             + parsers.semicolon + parsers.blankline^2))
                         - (parsers.spnl * parsers.rbracket))^0)

parsers.citation_headless_body
                    = parsers.citation_headless_body_postnote
                    * (parsers.sp * parsers.semicolon * parsers.spnl
                      * parsers.citation_body_chunk)^0
```

3.1.4.8 Parsers Used for Footnotes

```
local function strip_first_char(s)
  return s:sub(2)
end

parsers.RawNoteRef = #(parsers.lbracket * parsers.circumflex)
                   * parsers.tag / strip_first_char
```

3.1.4.9 Parsers Used for Tables

```
local function make_pipe_table_rectangular(rows)
  local num_columns = #rows[2]
  local rectangular_rows = {}
  for i = 1, #rows do
    local row = rows[i]
    local rectangular_row = {}
    for j = 1, num_columns do
      rectangular_row[j] = row[j] or ""
    end
    table.insert(rectangular_rows, rectangular_row)
  end
  return rectangular_rows
end

local function pipe_table_row(allow_empty_first_column
                              , nonempty_column
                              , column_separator
                              , column)
  local row_beginning
  if allow_empty_first_column then
```

```
    row_beginning = -- empty first column
                    #(parsers.spacechar^4
                     * column_separator)
                  * parsers.optionalspace
                  * column
                  * parsers.optionalspace
                  -- non-empty first column
                  + parsers.nonindentspace
                  * nonempty_column^-1
                  * parsers.optionalspace
  else
    row_beginning = parsers.nonindentspace
                  * nonempty_column^-1
                  * parsers.optionalspace
  end

  return Ct(row_beginning
            * (-- single column with no leading pipes
               #(column_separator
                 * parsers.optionalspace
                 * parsers.newline)
              * column_separator
              * parsers.optionalspace
              -- single column with leading pipes or
              -- more than a single column
              + (column_separator
                * parsers.optionalspace
                * column
                * parsers.optionalspace)^1
              * (column_separator
                * parsers.optionalspace)^-1))
end

parsers.table_hline_separator = parsers.pipe + parsers.plus
parsers.table_hline_column = (parsers.dash
                              - #(parsers.dash
                                 * (parsers.spacechar
                                   + parsers.table_hline_separator
                                   + parsers.newline)))^1
                           * (parsers.colon * Cc("r")
                             + parsers.dash * Cc("d"))
                           + parsers.colon
                           * (parsers.dash
                             - #(parsers.dash
                                * (parsers.spacechar
                                  + parsers.table_hline_separator
                                  + parsers.newline)))^1
```

```
                            * (parsers.colon * Cc("c")
                              + parsers.dash * Cc("l"))
parsers.table_hline = pipe_table_row(false
                                     , parsers.table_hline_column
                                     , parsers.table_hline_separator
                                     , parsers.table_hline_column)
parsers.table_caption_beginning = parsers.skipblanklines
                                * parsers.nonindentspace
                                * (P("Table")^-1 * parsers.colon)
                                * parsers.optionalspace
```

3.1.4.10 Parsers Used for HTML

```
-- case-insensitive match (we assume s is lowercase). must be single byte encoding
parsers.keyword_exact = function(s)
  local parser = P(0)
  for i=1,#s do
    local c = s:sub(i,i)
    local m = c .. upper(c)
    parser = parser * S(m)
  end
  return parser
end

parsers.block_keyword =
    parsers.keyword_exact("address") + parsers.keyword_exact("blockquote") +
    parsers.keyword_exact("center") + parsers.keyword_exact("del") +
    parsers.keyword_exact("dir") + parsers.keyword_exact("div") +
    parsers.keyword_exact("p") + parsers.keyword_exact("pre") +
    parsers.keyword_exact("li") + parsers.keyword_exact("ol") +
    parsers.keyword_exact("ul") + parsers.keyword_exact("dl") +
    parsers.keyword_exact("dd") + parsers.keyword_exact("form") +
    parsers.keyword_exact("fieldset") + parsers.keyword_exact("isindex") +
    parsers.keyword_exact("ins") + parsers.keyword_exact("menu") +
    parsers.keyword_exact("noframes") + parsers.keyword_exact("frameset") +
    parsers.keyword_exact("h1") + parsers.keyword_exact("h2") +
    parsers.keyword_exact("h3") + parsers.keyword_exact("h4") +
    parsers.keyword_exact("h5") + parsers.keyword_exact("h6") +
    parsers.keyword_exact("hr") + parsers.keyword_exact("script") +
    parsers.keyword_exact("noscript") + parsers.keyword_exact("table") +
    parsers.keyword_exact("tbody") + parsers.keyword_exact("tfoot") +
    parsers.keyword_exact("thead") + parsers.keyword_exact("th") +
    parsers.keyword_exact("td") + parsers.keyword_exact("tr")

-- There is no reason to support bad html, so we expect quoted attributes
parsers.htmlattributevalue
                          = parsers.squote * (parsers.any - (parsers.blankline
```

```
                                                          + parsers.squote))^0
                                        * parsers.squote
                          + parsers.dquote * (parsers.any - (parsers.blankline
                                                          + parsers.dquote))^0
                                        * parsers.dquote

parsers.htmlattribute     = parsers.spacing^1
                          * (parsers.alphanumeric + S("_-"))^1
                          * parsers.sp * parsers.equal * parsers.sp
                          * parsers.htmlattributevalue

parsers.htmlcomment       = P("<!--")
                          * parsers.optionalspace
                          * Cs((parsers.any - parsers.optionalspace * P("-->"))^0)
                          * parsers.optionalspace
                          * P("-->")

parsers.htmlinstruction   = P("<?") * (parsers.any - P("?>"))^0 * P("?>")

parsers.openelt_any = parsers.less * parsers.keyword * parsers.htmlattribute^0
                    * parsers.sp * parsers.more

parsers.openelt_exact = function(s)
  return parsers.less * parsers.sp * parsers.keyword_exact(s)
       * parsers.htmlattribute^0 * parsers.sp * parsers.more
end

parsers.openelt_block = parsers.sp * parsers.block_keyword
                      * parsers.htmlattribute^0 * parsers.sp * parsers.more

parsers.closeelt_any = parsers.less * parsers.sp * parsers.slash
                     * parsers.keyword * parsers.sp * parsers.more

parsers.closeelt_exact = function(s)
  return parsers.less * parsers.sp * parsers.slash * parsers.keyword_exact(s)
       * parsers.sp * parsers.more
end

parsers.emptyelt_any = parsers.less * parsers.sp * parsers.keyword
                     * parsers.htmlattribute^0 * parsers.sp * parsers.slash
                     * parsers.more

parsers.emptyelt_block = parsers.less * parsers.sp * parsers.block_keyword
                       * parsers.htmlattribute^0 * parsers.sp * parsers.slash
                       * parsers.more

parsers.displaytext = (parsers.any - parsers.less)^1
```

```
3870
3871 -- return content between two matched HTML tags
3872 parsers.in_matched = function(s)
3873   return { parsers.openelt_exact(s)
3874          * (V(1) + parsers.displaytext
3875            + (parsers.less - parsers.closeelt_exact(s)))^0
3876          * parsers.closeelt_exact(s) }
3877 end
3878
3879 local function parse_matched_tags(s,pos)
3880   local t = string.lower(lpeg.match(C(parsers.keyword),s,pos))
3881   return lpeg.match(parsers.in_matched(t),s,pos-1)
3882 end
3883
3884 parsers.in_matched_block_tags = parsers.less
3885                               * Cmt(#parsers.openelt_block, parse_matched_tags)
3886
3887 parsers.displayhtml = parsers.htmlcomment / ""
3888                     + parsers.emptyelt_block
3889                     + parsers.openelt_exact("hr")
3890                     + parsers.in_matched_block_tags
3891                     + parsers.htmlinstruction
```

3.1.4.11 Parsers Used for HTML Entities

```
3892 parsers.hexentity = parsers.ampersand * parsers.hash * S("Xx")
3893                   * C(parsers.hexdigit^1) * parsers.semicolon
3894 parsers.decentity = parsers.ampersand * parsers.hash
3895                   * C(parsers.digit^1) * parsers.semicolon
3896 parsers.tagentity = parsers.ampersand * C(parsers.alphanumeric^1)
3897                   * parsers.semicolon
```

3.1.4.12 Helpers for References

```
3898 -- parse a reference definition:  [foo]: /bar "title"
3899 parsers.define_reference_parser = parsers.leader * parsers.tag * parsers.colon
3900                                  * parsers.spacechar^0 * parsers.url
3901                                  * parsers.optionaltitle * parsers.blankline^1
```

3.1.4.13 Inline Elements

```
3902 parsers.Inline         = V("Inline")
3903 parsers.IndentedInline = V("IndentedInline")
3904
3905 -- parse many p between starter and ender
3906 parsers.between = function(p, starter, ender)
3907   local ender2 = B(parsers.nonspacechar) * ender
3908   return (starter * #parsers.nonspacechar * Ct(p * (p - ender2)^0) * ender2)
```

```
3909 end
3910
3911 parsers.urlchar      = parsers.anyescaped - parsers.newline - parsers.more
```

3.1.4.14 Block Elements

```
3912 parsers.Block        = V("Block")
3913
3914 parsers.OnlineImageURL
3915                      = parsers.leader
3916                      * parsers.onlineimageurl
3917                      * parsers.optionaltitle
3918
3919 parsers.LocalFilePath
3920                      = parsers.leader
3921                      * parsers.localfilepath
3922                      * parsers.optionaltitle
3923
3924 parsers.TildeFencedCode
3925                      = parsers.fencehead(parsers.tilde)
3926                      * Cs(parsers.fencedline(parsers.tilde)^0)
3927                      * parsers.fencetail(parsers.tilde)
3928
3929 parsers.BacktickFencedCode
3930                      = parsers.fencehead(parsers.backtick)
3931                      * Cs(parsers.fencedline(parsers.backtick)^0)
3932                      * parsers.fencetail(parsers.backtick)
3933
3934 parsers.lineof = function(c)
3935     return (parsers.leader * (P(c) * parsers.optionalspace)^3
3936             * (parsers.newline * parsers.blankline^1
3937               + parsers.newline^-1 * parsers.eof))
3938 end
```

3.1.4.15 Lists

```
3939 parsers.defstartchar = S("~:")
3940 parsers.defstart     = ( parsers.defstartchar * #parsers.spacing
3941                                               * (parsers.tab + parsers.space^-
    3)
3942                      + parsers.space * parsers.defstartchar * #parsers.spacing
3943                                      * (parsers.tab + parsers.space^-2)
3944                      + parsers.space * parsers.space * parsers.defstartchar
3945                                      * #parsers.spacing
3946                                      * (parsers.tab + parsers.space^-1)
3947                      + parsers.space * parsers.space * parsers.space
3948                                      * parsers.defstartchar * #parsers.spacing
3949                      )
```

```
3950
3951 parsers.dlchunk = Cs(parsers.line * (parsers.indentedline - parsers.blankline)^0)
```

3.1.4.16 Headings

```
3952 parsers.heading_attribute = C(parsers.css_identifier)
3953                           + C((parsers.attribute_name_char
3954                              - parsers.rbrace)^1
3955                              * parsers.equal
3956                              * (parsers.attribute_value_char
3957                                - parsers.rbrace)^1)
3958 parsers.HeadingAttributes = parsers.lbrace
3959                           * parsers.heading_attribute
3960                           * (parsers.spacechar^1
3961                             * parsers.heading_attribute)^0
3962                           * parsers.rbrace
3963
3964 -- parse Atx heading start and return level
3965 parsers.HeadingStart = #parsers.hash * C(parsers.hash^-6)
3966                      * -parsers.hash / length
3967
3968 -- parse setext header ending and return level
3969 parsers.HeadingLevel = parsers.equal^1 * Cc(1) + parsers.dash^1 * Cc(2)
3970
3971 local function strip_atx_end(s)
3972   return s:gsub("[#%s]*\n$","")
3973 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*⟩.

```
3974 M.reader = {}
```

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

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

```
  setmetatable(options, { __index = function (_, key)
    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.

```
  local function normalize_tag(tag)
    return string.lower(
      gsub(util.rope_to_string(tag), "[ \n\r\t]+", " "))
  end
```

Define `iterlines` as a function that iterates over the lines of the input string `s`, transforms them using an input function `f`, and reassembles them into a new string, which it returns.

```
  local function iterlines(s, f)
    rope = lpeg.match(Ct((parsers.line / f)^1), s)
    return util.rope_to_string(rope)
  end
```

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

```
  local expandtabs
  if options.preserveTabs then
    expandtabs = function(s) return s end
  else
    expandtabs = function(s)
                   if s:find("\t") then
                     return iterlines(s, util.expand_tabs_in_line)
                   else
                     return s
                   end
                 end
  end
```

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

```
  local larsers    = {}
```

3.1.5.2 Top-Level Parser Functions

```
  local function create_parser(name, grammar, toplevel)
    return function(str)
```

If the parser is top-level and the `stripIndent` Lua option is enabled, we will first expand tabs in the input string `str` into spaces and then we will count the minimum indent across all lines, skipping blank lines. Next, we will remove the minimum indent from all lines.

```
4003        if toplevel and options.stripIndent then
4004            local min_prefix_length, min_prefix = nil, ''
4005            str = iterlines(str, function(line)
4006                if lpeg.match(parsers.nonemptyline, line) == nil then
4007                    return line
4008                end
4009                line = util.expand_tabs_in_line(line)
4010                prefix = lpeg.match(C(parsers.optionalspace), line)
4011                local prefix_length = #prefix
4012                local is_shorter = min_prefix_length == nil
4013                is_shorter = is_shorter or prefix_length < min_prefix_length
4014                if is_shorter then
4015                    min_prefix_length, min_prefix = prefix_length, prefix
4016                end
4017                return line
4018            end)
4019            str = str:gsub('^' .. min_prefix, '')
4020        end
```

If the parser is top-level and the `texComments` Lua option is enabled, we will strip all plain TEX comments from the input string `str` together with the trailing newline characters.

```
4021        if toplevel and options.texComments then
4022            str = lpeg.match(Ct(parsers.commented_line^1), str)
4023            str = util.rope_to_string(str)
4024            print(str)
4025        end
4026        local res = lpeg.match(grammar(), str)
4027        if res == nil then
4028          error(format("%s failed on:\n%s", name, str:sub(1,20)))
4029        else
4030          return res
4031        end
4032      end
4033    end
4034
4035    local parse_blocks
4036      = create_parser("parse_blocks",
4037                      function()
4038                        return larsers.blocks
4039                      end, false)
4040
4041    local parse_blocks_toplevel
```

```
  = create_parser("parse_blocks_toplevel",
                  function()
                    return larsers.blocks_toplevel
                  end, true)

local parse_inlines
  = create_parser("parse_inlines",
                  function()
                    return larsers.inlines
                  end, false)

local parse_inlines_no_link
  = create_parser("parse_inlines_no_link",
                  function()
                    return larsers.inlines_no_link
                  end, false)

local parse_inlines_no_inline_note
  = create_parser("parse_inlines_no_inline_note",
                  function()
                    return larsers.inlines_no_inline_note
                  end, false)

local parse_inlines_no_html
  = create_parser("parse_inlines_no_html",
                  function()
                    return larsers.inlines_no_html
                  end, false)

local parse_inlines_nbsp
  = create_parser("parse_inlines_nbsp",
                  function()
                    return larsers.inlines_nbsp
                  end, false)
```

3.1.5.3 Parsers Used for Markdown Lists (local)

```
if options.hashEnumerators then
  larsers.dig = parsers.digit + parsers.hash
else
  larsers.dig = parsers.digit
end

larsers.enumerator = C(larsers.dig^3 * parsers.period) * #parsers.spacing
                   + C(larsers.dig^2 * parsers.period) * #parsers.spacing
                                     * (parsers.tab + parsers.space^1)
                   + C(larsers.dig * parsers.period) * #parsers.spacing
```

```
                                    * (parsers.tab + parsers.space^-2)
                     + parsers.space * C(larsers.dig^2 * parsers.period)
                                    * #parsers.spacing
                     + parsers.space * C(larsers.dig * parsers.period)
                                    * #parsers.spacing
                                    * (parsers.tab + parsers.space^-1)
                     + parsers.space * parsers.space * C(larsers.dig^1
                                    * parsers.period) * #parsers.spacing
```

3.1.5.4 Parsers Used for Blockquotes (local)

```
  -- strip off leading > and indents, and run through blocks
  larsers.blockquote_body = ((parsers.leader * parsers.more * parsers.space^-
1)/""
                             * parsers.linechar^0 * parsers.newline)^1
                           * (-(parsers.leader * parsers.more
                               + parsers.blankline) * parsers.linechar^1
                             * parsers.newline)^0

  if not options.breakableBlockquotes then
    larsers.blockquote_body = larsers.blockquote_body
                           * (parsers.blankline^0 / "")
  end
```

3.1.5.5 Parsers Used for Citations (local)

```
  larsers.citations = function(text_cites, raw_cites)
      local function normalize(str)
          if str == "" then
              str = nil
          else
              str = (options.citationNbsps and parse_inlines_nbsp or
                parse_inlines)(str)
          end
          return str
      end

      local cites = {}
      for i = 1,#raw_cites,4 do
          cites[#cites+1] = {
              prenote = normalize(raw_cites[i]),
              suppress_author = raw_cites[i+1] == "-",
              name = writer.citation(raw_cites[i+2]),
              postnote = normalize(raw_cites[i+3]),
          }
      end
      return writer.citations(text_cites, cites)
  end
```

3.1.5.6 Parsers Used for Footnotes (local)

```
4127  local rawnotes = {}
4128
4129  -- like indirect_link
4130  local function lookup_note(ref)
4131    return writer.defer_call(function()
4132      local found = rawnotes[normalize_tag(ref)]
4133      if found then
4134        return writer.note(parse_blocks_toplevel(found))
4135      else
4136        return {"[", parse_inlines("^" .. ref), "]"}
4137      end
4138    end)
4139  end
4140
4141  local function register_note(ref,rawnote)
4142    rawnotes[normalize_tag(ref)] = rawnote
4143    return ""
4144  end
4145
4146  larsers.NoteRef    = parsers.RawNoteRef / lookup_note
4147
4148
4149  larsers.NoteBlock  = parsers.leader * parsers.RawNoteRef * parsers.colon
4150                     * parsers.spnl * parsers.indented_blocks(parsers.chunk)
4151                     / register_note
4152
4153  larsers.InlineNote = parsers.circumflex
4154                     * (parsers.tag / parse_inlines_no_inline_note) -- no notes inside n
4155                     / writer.note
```

3.1.5.7 Parsers Used for Tables (local)

```
4156 larsers.table_row = pipe_table_row(true
4157                                    , (C((parsers.linechar - parsers.pipe)^1)
4158                                      / parse_inlines)
4159                                    , parsers.pipe
4160                                    , (C((parsers.linechar - parsers.pipe)^0)
4161                                      / parse_inlines))
4162
4163 if options.tableCaptions then
4164   larsers.table_caption = #parsers.table_caption_beginning
4165                         * parsers.table_caption_beginning
4166                         * Ct(parsers.IndentedInline^1)
4167                         * parsers.newline
4168 else
4169   larsers.table_caption = parsers.fail
```

```
4170 end
4171
4172 larsers.PipeTable = Ct(larsers.table_row * parsers.newline
4173                     * parsers.table_hline
4174                     * (parsers.newline * larsers.table_row)^0)
4175                   / make_pipe_table_rectangular
4176                   * larsers.table_caption^-1
4177                   / writer.table
```

3.1.5.8 Helpers for Links and References (local)

```
4178   -- List of references defined in the document
4179   local references
4180
4181   -- add a reference to the list
4182   local function register_link(tag,url,title)
4183       references[normalize_tag(tag)] = { url = url, title = title }
4184       return ""
4185   end
4186
4187   -- lookup link reference and return either
4188   -- the link or nil and fallback text.
4189   local function lookup_reference(label,sps,tag)
4190       local tagpart
4191       if not tag then
4192           tag = label
4193           tagpart = ""
4194       elseif tag == "" then
4195           tag = label
4196           tagpart = "[]"
4197       else
4198           tagpart = {"[", parse_inlines(tag), "]"}
4199       end
4200       if sps then
4201         tagpart = {sps, tagpart}
4202       end
4203       local r = references[normalize_tag(tag)]
4204       if r then
4205         return r
4206       else
4207         return nil, {"[", parse_inlines(label), "]", tagpart}
4208       end
4209   end
4210
4211   -- lookup link reference and return a link, if the reference is found,
4212   -- or a bracketed label otherwise.
4213   local function indirect_link(label,sps,tag)
```

```
    return writer.defer_call(function()
      local r,fallback = lookup_reference(label,sps,tag)
      if r then
        return writer.link(parse_inlines_no_link(label), r.url, r.title)
      else
        return fallback
      end
    end)
  end

  -- lookup image reference and return an image, if the reference is found,
  -- or a bracketed label otherwise.
  local function indirect_image(label,sps,tag)
    return writer.defer_call(function()
      local r,fallback = lookup_reference(label,sps,tag)
      if r then
        return writer.image(writer.string(label), r.url, r.title)
      else
        return {"!", fallback}
      end
    end)
  end
```

3.1.5.9 Inline Elements (local)

```
  larsers.Str      = (parsers.normalchar * (parsers.normalchar + parsers.at)^0)
                   / writer.string

  larsers.Symbol   = (parsers.specialchar - parsers.tightblocksep)
                   / writer.string

  larsers.Ellipsis = P("...") / writer.ellipsis

  larsers.Smart    = larsers.Ellipsis

  larsers.Code     = parsers.inticks / writer.code

  if options.blankBeforeBlockquote then
    larsers.bqstart = parsers.fail
  else
    larsers.bqstart = parsers.more
  end

  if options.blankBeforeHeading then
    larsers.headerstart = parsers.fail
  else
    larsers.headerstart = parsers.hash
```

```
                       + (parsers.line * (parsers.equal^1 + parsers.dash^1)
                       * parsers.optionalspace * parsers.newline)
end

if not options.fencedCode or options.blankBeforeCodeFence then
  larsers.fencestart = parsers.fail
else
  larsers.fencestart = parsers.fencehead(parsers.backtick)
                     + parsers.fencehead(parsers.tilde)
end

larsers.Endline   = parsers.newline * -( -- newline, but not before...
                      parsers.blankline -- paragraph break
                    + parsers.tightblocksep  -- nested list
                    + parsers.eof       -- end of document
                    + larsers.bqstart
                    + larsers.headerstart
                    + larsers.fencestart
                  ) * parsers.spacechar^0 / writer.space

larsers.OptionalIndent
                   = parsers.spacechar^1 / writer.space

larsers.Space      = parsers.spacechar^2 * larsers.Endline / writer.linebreak
                   + parsers.spacechar^1 * larsers.Endline^-1 * parsers.eof / ""
                   + parsers.spacechar^1 * larsers.Endline^-1
                                         * parsers.optionalspace / writer.space

larsers.NonbreakingEndline
                  = parsers.newline * -( -- newline, but not before...
                      parsers.blankline -- paragraph break
                    + parsers.tightblocksep  -- nested list
                    + parsers.eof       -- end of document
                    + larsers.bqstart
                    + larsers.headerstart
                    + larsers.fencestart
                  ) * parsers.spacechar^0 / writer.nbsp

larsers.NonbreakingSpace
                = parsers.spacechar^2 * larsers.Endline / writer.linebreak
                + parsers.spacechar^1 * larsers.Endline^-1 * parsers.eof / ""
                + parsers.spacechar^1 * larsers.Endline^-1
                                      * parsers.optionalspace / writer.nbsp

if options.underscores then
  larsers.Strong = ( parsers.between(parsers.Inline, parsers.doubleasterisks,
                                     parsers.doubleasterisks)
```

```
                    + parsers.between(parsers.Inline, parsers.doubleunderscores,
                                      parsers.doubleunderscores)
                    ) / writer.strong

  larsers.Emph   = ( parsers.between(parsers.Inline, parsers.asterisk,
                                      parsers.asterisk)
                    + parsers.between(parsers.Inline, parsers.underscore,
                                      parsers.underscore)
                    ) / writer.emphasis
else
  larsers.Strong = ( parsers.between(parsers.Inline, parsers.doubleasterisks,
                                      parsers.doubleasterisks)
                    ) / writer.strong

  larsers.Emph   = ( parsers.between(parsers.Inline, parsers.asterisk,
                                      parsers.asterisk)
                    ) / writer.emphasis
end

larsers.AutoLinkUrl    = parsers.less
                        * C(parsers.alphanumeric^1 * P("://") * parsers.urlchar^1)
                        * parsers.more
                        / function(url)
                            return writer.link(writer.escape(url), url)
                          end

larsers.AutoLinkEmail = parsers.less
                       * C((parsers.alphanumeric + S("-._+"))^1
                       * P("@") * parsers.urlchar^1)
                       * parsers.more
                       / function(email)
                           return writer.link(writer.escape(email),
                                              "mailto:"..email)
                         end

larsers.DirectLink    = (parsers.tag / parse_inlines_no_link)  -- no links inside linl
                       * parsers.spnl
                       * parsers.lparent
                       * (parsers.url + Cc(""))  -- link can be empty [foo]()
                       * parsers.optionaltitle
                       * parsers.rparent
                       / writer.link

larsers.IndirectLink  = parsers.tag * (C(parsers.spnl) * parsers.tag)^-
1
                       / indirect_link

```

```
4351  -- parse a link or image (direct or indirect)
4352  larsers.Link          = larsers.DirectLink + larsers.IndirectLink
4353
4354  larsers.DirectImage   = parsers.exclamation
4355                        * (parsers.tag / parse_inlines)
4356                        * parsers.spnl
4357                        * parsers.lparent
4358                        * (parsers.url + Cc(""))  -- link can be empty [foo]()
4359                        * parsers.optionaltitle
4360                        * parsers.rparent
4361                        / writer.image
4362
4363  larsers.IndirectImage = parsers.exclamation * parsers.tag
4364                        * (C(parsers.spnl) * parsers.tag)^-1 / indirect_image
4365
4366  larsers.Image         = larsers.DirectImage + larsers.IndirectImage
4367
4368  larsers.TextCitations = Ct((parsers.spnl
4369                        * Cc("")
4370                        * parsers.citation_name
4371                        * ((parsers.spnl
4372                            * parsers.lbracket
4373                            * parsers.citation_headless_body
4374                            * parsers.rbracket) + Cc("")))^1)
4375                        / function(raw_cites)
4376                            return larsers.citations(true, raw_cites)
4377                          end
4378
4379  larsers.ParenthesizedCitations
4380                        = Ct((parsers.spnl
4381                        * parsers.lbracket
4382                        * parsers.citation_body
4383                        * parsers.rbracket)^1)
4384                        / function(raw_cites)
4385                            return larsers.citations(false, raw_cites)
4386                          end
4387
4388  larsers.Citations     = larsers.TextCitations + larsers.ParenthesizedCitations
4389
4390  -- avoid parsing long strings of * or _ as emph/strong
4391  larsers.UlOrStarLine  = parsers.asterisk^4 + parsers.underscore^4
4392                        / writer.string
4393
4394  larsers.EscapedChar   = parsers.backslash * C(parsers.escapable) / writer.string
4395
4396  larsers.InlineHtml    = parsers.emptyelt_any
4397                        + (parsers.htmlcomment / parse_inlines_no_html)
```

```
                        / writer.inline_html_comment
                        + parsers.htmlinstruction
                        + parsers.openelt_any
                        + parsers.closeelt_any

  larsers.HtmlEntity    = parsers.hexentity / entities.hex_entity  / writer.string
                        + parsers.decentity / entities.dec_entity  / writer.string
                        + parsers.tagentity / entities.char_entity / writer.string
```

3.1.5.10 Block Elements (local)

```
  larsers.ContentBlock = parsers.leader
                       * (parsers.localfilepath + parsers.onlineimageurl)
                       * parsers.contentblock_tail
                       / writer.contentblock

  larsers.DisplayHtml  = parsers.displayhtml

  larsers.Verbatim     = Cs( (parsers.blanklines
                           * ((parsers.indentedline - parsers.blankline))^1)^1
                           ) / expandtabs / writer.verbatim

  larsers.FencedCode   = (parsers.TildeFencedCode
                         + parsers.BacktickFencedCode)
                       / function(infostring, code)
                           return writer.fencedCode(writer.string(infostring),
                                                    expandtabs(code))
                         end

  larsers.Blockquote   = Cs(larsers.blockquote_body^1)
                       / parse_blocks_toplevel / writer.blockquote

  larsers.HorizontalRule = ( parsers.lineof(parsers.asterisk)
                           + parsers.lineof(parsers.dash)
                           + parsers.lineof(parsers.underscore)
                           ) / writer.hrule

  larsers.Reference    = parsers.define_reference_parser / register_link

  larsers.Paragraph    = parsers.nonindentspace * Ct(parsers.Inline^1)
                       * parsers.newline
                       * ( parsers.blankline^1
                         + #parsers.hash
                         + #(parsers.leader * parsers.more * parsers.space^-
1)
                         )
                       / writer.paragraph
```

```
4441
4442  larsers.ToplevelParagraph
4443                        = parsers.nonindentspace * Ct(parsers.Inline^1)
4444                        * ( parsers.newline
4445                        * ( parsers.blankline^1
4446                          + #parsers.hash
4447                          + #(parsers.leader * parsers.more * parsers.space^-
    1)
4448                          + parsers.eof
4449                          )
4450                        + parsers.eof )
4451                        / writer.paragraph
4452
4453  larsers.Plain         = parsers.nonindentspace * Ct(parsers.Inline^1)
4454                        / writer.plain
```

3.1.5.11 Lists (local)

```
4455  larsers.starter = parsers.bullet + larsers.enumerator
4456
4457  -- we use \001 as a separator between a tight list item and a
4458  -- nested list under it.
4459  larsers.NestedList            = Cs((parsers.optionallyindentedline
4460                                      - larsers.starter)^1)
4461                                / function(a) return "\001"..a end
4462
4463  larsers.ListBlockLine         = parsers.optionallyindentedline
4464                                - parsers.blankline - (parsers.indent^-1
4465                                                     * larsers.starter)
4466
4467  larsers.ListBlock             = parsers.line * larsers.ListBlockLine^0
4468
4469  larsers.ListContinuationBlock = parsers.blanklines * (parsers.indent / "")
4470                                * larsers.ListBlock
4471
4472  larsers.TightListItem = function(starter)
4473      return -larsers.HorizontalRule
4474             * (Cs(starter / "" * larsers.ListBlock * larsers.NestedList^-
    1)
4475               / parse_blocks)
4476             * -(parsers.blanklines * parsers.indent)
4477  end
4478
4479  larsers.LooseListItem = function(starter)
4480      return -larsers.HorizontalRule
4481             * Cs( starter / "" * larsers.ListBlock * Cc("\n")
4482               * (larsers.NestedList + larsers.ListContinuationBlock^0)
```

```
              * (parsers.blanklines / "\n\n")
              ) / parse_blocks
end

larsers.BulletList = ( Ct(larsers.TightListItem(parsers.bullet)^1) * Cc(true)
                     * parsers.skipblanklines * -parsers.bullet
                     + Ct(larsers.LooseListItem(parsers.bullet)^1) * Cc(false)
                     * parsers.skipblanklines )
                   / writer.bulletlist

local function ordered_list(items,tight,startNumber)
  if options.startNumber then
    startNumber = tonumber(startNumber) or 1  -- fallback for '#'
    if startNumber ~= nil then
      startNumber = math.floor(startNumber)
    end
  else
    startNumber = nil
  end
  return writer.orderedlist(items,tight,startNumber)
end

larsers.OrderedList = Cg(larsers.enumerator, "listtype") *
                    ( Ct(larsers.TightListItem(Cb("listtype"))
                        * larsers.TightListItem(larsers.enumerator)^0)
                    * Cc(true) * parsers.skipblanklines * -larsers.enumerator
                    + Ct(larsers.LooseListItem(Cb("listtype"))
                        * larsers.LooseListItem(larsers.enumerator)^0)
                    * Cc(false) * parsers.skipblanklines
                    ) * Cb("listtype") / ordered_list

local function definition_list_item(term, defs, tight)
  return { term = parse_inlines(term), definitions = defs }
end

larsers.DefinitionListItemLoose = C(parsers.line) * parsers.skipblanklines
                                * Ct((parsers.defstart
                                     * parsers.indented_blocks(parsers.dlchunk)
                                     / parse_blocks_toplevel)^1)
                                * Cc(false) / definition_list_item

larsers.DefinitionListItemTight = C(parsers.line)
                                * Ct((parsers.defstart * parsers.dlchunk
                                     / parse_blocks)^1)
                                * Cc(true) / definition_list_item

larsers.DefinitionList = ( Ct(larsers.DefinitionListItemLoose^1) * Cc(false)
```

```
                          + Ct(larsers.DefinitionListItemTight^1)
                          * (parsers.skipblanklines
                            * -larsers.DefinitionListItemLoose * Cc(true))
                          ) / writer.definitionlist
```

3.1.5.12 Blank (local)

```
larsers.Blank        = parsers.blankline / ""
                     + larsers.NoteBlock
                     + larsers.Reference
                     + (parsers.tightblocksep / "\n")
```

3.1.5.13 Headings (local)

```
-- parse atx header
if options.headerAttributes then
  larsers.AtxHeading = Cg(parsers.HeadingStart,"level")
                     * parsers.optionalspace
                     * (C(((parsers.linechar
                           - ((parsers.hash^1
                              * parsers.optionalspace
                              * parsers.HeadingAttributes^-1
                              + parsers.HeadingAttributes)
                             * parsers.optionalspace
                             * parsers.newline))
                          * (parsers.linechar
                            - parsers.hash
                            - parsers.lbrace)^0)^1)
                         / parse_inlines)
                     * Cg(Ct(parsers.newline
                           + (parsers.hash^1
                             * parsers.optionalspace
                             * parsers.HeadingAttributes^-1
                             + parsers.HeadingAttributes)
                           * parsers.optionalspace
                           * parsers.newline), "attributes")
                     * Cb("level")
                     * Cb("attributes")
                     / writer.heading

  larsers.SetextHeading = #(parsers.line * S("=-"))
                        * (C(((parsers.linechar
                              - (parsers.HeadingAttributes
                                * parsers.optionalspace
                                * parsers.newline))
                             * (parsers.linechar
                               - parsers.lbrace)^0)^1)
                            / parse_inlines)
```

```
                          * Cg(Ct(parsers.newline
                                + (parsers.HeadingAttributes
                                  * parsers.optionalspace
                                  * parsers.newline)), "attributes")
                          * parsers.HeadingLevel
                          * Cb("attributes")
                          * parsers.optionalspace
                          * parsers.newline
                          / writer.heading
  else
    larsers.AtxHeading = Cg(parsers.HeadingStart,"level")
                         * parsers.optionalspace
                         * (C(parsers.line) / strip_atx_end / parse_inlines)
                         * Cb("level")
                         / writer.heading

    larsers.SetextHeading = #(parsers.line * S("=-"))
                          * Ct(parsers.linechar^1 / parse_inlines)
                          * parsers.newline
                          * parsers.HeadingLevel
                          * parsers.optionalspace
                          * parsers.newline
                          / writer.heading
  end

  larsers.Heading = larsers.AtxHeading + larsers.SetextHeading
```

3.1.5.14 Syntax Specification

```
  local syntax =
    { "Blocks",

      Blocks                = larsers.Blank^0 * parsers.Block^-1
                            * (larsers.Blank^0 / writer.interblocksep
                              * parsers.Block)^0
                            * larsers.Blank^0 * parsers.eof,

      Blank                 = larsers.Blank,

      Block                 = V("ContentBlock")
                            + V("Blockquote")
                            + V("PipeTable")
                            + V("Verbatim")
                            + V("FencedCode")
                            + V("HorizontalRule")
                            + V("BulletList")
                            + V("OrderedList")
```

```
                        + V("Heading")
                        + V("DefinitionList")
                        + V("DisplayHtml")
                        + V("Paragraph")
                        + V("Plain"),

    ContentBlock        = larsers.ContentBlock,
    Blockquote          = larsers.Blockquote,
    Verbatim            = larsers.Verbatim,
    FencedCode          = larsers.FencedCode,
    HorizontalRule      = larsers.HorizontalRule,
    BulletList          = larsers.BulletList,
    OrderedList         = larsers.OrderedList,
    Heading             = larsers.Heading,
    DefinitionList      = larsers.DefinitionList,
    DisplayHtml         = larsers.DisplayHtml,
    Paragraph           = larsers.Paragraph,
    PipeTable           = larsers.PipeTable,
    Plain               = larsers.Plain,

    Inline              = V("Str")
                        + V("Space")
                        + V("Endline")
                        + V("UlOrStarLine")
                        + V("Strong")
                        + V("Emph")
                        + V("InlineNote")
                        + V("NoteRef")
                        + V("Citations")
                        + V("Link")
                        + V("Image")
                        + V("Code")
                        + V("AutoLinkUrl")
                        + V("AutoLinkEmail")
                        + V("InlineHtml")
                        + V("HtmlEntity")
                        + V("EscapedChar")
                        + V("Smart")
                        + V("Symbol"),

    IndentedInline      = V("Str")
                        + V("OptionalIndent")
                        + V("Endline")
                        + V("UlOrStarLine")
                        + V("Strong")
                        + V("Emph")
                        + V("InlineNote")
```

```
                              + V("NoteRef")
                              + V("Citations")
                              + V("Link")
                              + V("Image")
                              + V("Code")
                              + V("AutoLinkUrl")
                              + V("AutoLinkEmail")
                              + V("InlineHtml")
                              + V("HtmlEntity")
                              + V("EscapedChar")
                              + V("Smart")
                              + V("Symbol"),

    Str                     = larsers.Str,
    Space                   = larsers.Space,
    OptionalIndent          = larsers.OptionalIndent,
    Endline                 = larsers.Endline,
    UlOrStarLine            = larsers.UlOrStarLine,
    Strong                  = larsers.Strong,
    Emph                    = larsers.Emph,
    InlineNote              = larsers.InlineNote,
    NoteRef                 = larsers.NoteRef,
    Citations               = larsers.Citations,
    Link                    = larsers.Link,
    Image                   = larsers.Image,
    Code                    = larsers.Code,
    AutoLinkUrl             = larsers.AutoLinkUrl,
    AutoLinkEmail           = larsers.AutoLinkEmail,
    InlineHtml              = larsers.InlineHtml,
    HtmlEntity              = larsers.HtmlEntity,
    EscapedChar             = larsers.EscapedChar,
    Smart                   = larsers.Smart,
    Symbol                  = larsers.Symbol,
  }

if not options.citations then
  syntax.Citations = parsers.fail
end

if not options.contentBlocks then
  syntax.ContentBlock = parsers.fail
end

if not options.codeSpans then
  syntax.Code = parsers.fail
end

```

```
4710  if not options.definitionLists then
4711    syntax.DefinitionList = parsers.fail
4712  end
4713
4714  if not options.fencedCode then
4715    syntax.FencedCode = parsers.fail
4716  end
4717
4718  if not options.footnotes then
4719    syntax.NoteRef = parsers.fail
4720  end
4721
4722  if not options.html then
4723    syntax.DisplayHtml = parsers.fail
4724    syntax.InlineHtml = parsers.fail
4725    syntax.HtmlEntity  = parsers.fail
4726  end
4727
4728  if not options.inlineFootnotes then
4729    syntax.InlineNote = parsers.fail
4730  end
4731
4732  if not options.smartEllipses then
4733    syntax.Smart = parsers.fail
4734  end
4735
4736  if options.preserveTabs then
4737    options.stripIndent = false
4738  end
4739
4740  if not options.pipeTables then
4741    syntax.PipeTable = parsers.fail
4742  end
4743
4744  local blocks_toplevel_t = util.table_copy(syntax)
4745  blocks_toplevel_t.Paragraph = larsers.ToplevelParagraph
4746  larsers.blocks_toplevel = Ct(blocks_toplevel_t)
4747
4748  larsers.blocks = Ct(syntax)
4749
4750  local inlines_t = util.table_copy(syntax)
4751  inlines_t[1] = "Inlines"
4752  inlines_t.Inlines = parsers.Inline^0 * (parsers.spacing^0 * parsers.eof / "")
4753  larsers.inlines = Ct(inlines_t)
4754
4755  local inlines_no_link_t = util.table_copy(inlines_t)
4756  inlines_no_link_t.Link = parsers.fail
```

```
4757  larsers.inlines_no_link = Ct(inlines_no_link_t)
4758
4759  local inlines_no_inline_note_t = util.table_copy(inlines_t)
4760  inlines_no_inline_note_t.InlineNote = parsers.fail
4761  larsers.inlines_no_inline_note = Ct(inlines_no_inline_note_t)
4762
4763  local inlines_no_html_t = util.table_copy(inlines_t)
4764  inlines_no_html_t.DisplayHtml = parsers.fail
4765  inlines_no_html_t.InlineHtml = parsers.fail
4766  inlines_no_html_t.HtmlEntity = parsers.fail
4767  larsers.inlines_no_html = Ct(inlines_no_html_t)
4768
4769  local inlines_nbsp_t = util.table_copy(inlines_t)
4770  inlines_nbsp_t.Endline = larsers.NonbreakingEndline
4771  inlines_nbsp_t.Space = larsers.NonbreakingSpace
4772  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 TEX output and returns it. Note that the converter assumes that the input has UNIX line endings.

```
4773  function self.convert(input)
4774    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.

```
4775    local opt_string = {}
4776    for k,_ in pairs(defaultOptions) do
4777      local v = options[k]
4778      if k ~= "cacheDir" then
4779        opt_string[#opt_string+1] = k .. "=" .. tostring(v)
4780      end
4781    end
4782    table.sort(opt_string)
4783    local salt = table.concat(opt_string, ",") .. "," .. metadata.version
```

Produce the cache file and transform its filename to plain TEX output via the `writer->pack` method.

```
4784    local name = util.cache(options.cacheDir, input, salt, function(input)
4785        return util.rope_to_string(parse_blocks_toplevel(input)) .. writer.eof
4786      end, ".md" .. writer.suffix)
4787    local output = writer.pack(name)
```

If the `frozenCache` option is enabled, populate the frozen cache in the file `frozenCacheFileName` with an entry for markdown document number `frozenCacheCounter`.

```
4788    if options.finalizeCache then
```

```
      local file, mode
      if options.frozenCacheCounter > 0 then
        mode = "a"
      else
        mode = "w"
      end
      file = assert(io.open(options.frozenCacheFileName, mode),
        [[could not open file "]] .. options.frozenCacheFileName
        .. [[" for writing]])
      assert(file:write([[\expandafter\global\expandafter\def\csname ]]
        .. [[markdownFrozenCache]] .. options.frozenCacheCounter
        .. [[\endcsname{]] .. output .. [[}]] .. "\n"))
      assert(file:close())
    end
    return output
  end
  return self
end
```

3.1.6 Conversion from Markdown to Plain TEX

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`.

```
function M.new(options)
  local writer = M.writer.new(options)
  local reader = M.reader.new(writer, options)
  return reader.convert
end

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.

```

local input
if input_filename then
  local input_file = assert(io.open(input_filename, "r"),
    [[could not open file "]] .. input_filename .. [[" for reading]])
  input = assert(input_file:read("*a"))
  assert(input_file:close())
else
  input = assert(io.read("*a"))
end
```

```
4824
```

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

```
4825 local lfs = require("lfs")
4826 if options.cacheDir and not lfs.isdir(options.cacheDir) then
4827   assert(lfs.mkdir(options["cacheDir"]))
4828 end
4829
4830 local ran_ok, kpse = pcall(require, "kpse")
4831 if ran_ok then kpse.set_program_name("luatex") end
4832 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.

```
4833 if metadata.version ~= md.metadata.version then
4834   warn("markdown-cli.lua " .. metadata.version .. " used with " ..
4835        "markdown.lua " .. md.metadata.version .. ".")
4836 end
4837 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.

```
4838 local output = convert(input:gsub("\r\n?", "\n") .. "\n")
4839
4840 if output_filename then
4841   local output_file = assert(io.open(output_filename, "w"),
4842     [[could not open file "]] .. output_filename .. [[" for writing]])
4843   assert(output_file:write(output))
4844   assert(output_file:close())
4845 else
4846   assert(io.write(output))
4847 end
```

3.2 Plain TEX Implementation

The plain TEX implementation provides macros for the interfacing between TEX 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 TEX exposed by the plain TEX interface (see Section 2.2).

3.2.1 Logging Facilities

```
4848 \ifx\markdownInfo\undefined
4849   \def\markdownInfo#1{%
4850     \immediate\write-1{(l.\the\inputlineno) markdown.tex info: #1.}}%
4851 \fi
4852 \ifx\markdownWarning\undefined
```

```
  \def\markdownWarning#1{%
    \immediate\write16{(l.\the\inputlineno) markdown.tex warning: #1}}%
\fi
\ifx\markdownError\undefined
  \def\markdownError#1#2{%
    \errhelp{#2.}%
    \errmessage{(l.\the\inputlineno) markdown.tex error: #1}}%
\fi
```

3.2.2 Finalizing and Freezing the Cache

When the `\markdownOptionFinalizeCache` option is enabled, then the `\markdownFrozenCacheCounter` counter is used to enumerate the markdown documents using the Lua interface `frozenCacheCounter` option.

When the `\markdownOptionFrozenCache` option is enabled, then the `\markdownFrozenCacheCounter` counter is used to render markdown documents from the frozen cache without invoking Lua.

```
\newcount\markdownFrozenCacheCounter
```

3.2.3 Token Renderer Prototypes

The following definitions should be considered placeholder.

```
\def\markdownRendererInterblockSeparatorPrototype{\par}%
\def\markdownRendererLineBreakPrototype{\hfil\break}%
\let\markdownRendererEllipsisPrototype\dots
\def\markdownRendererNbspPrototype{~}%
\def\markdownRendererLeftBracePrototype{\char`\{}%
\def\markdownRendererRightBracePrototype{\char`\}}%
\def\markdownRendererDollarSignPrototype{\char`$}%
\def\markdownRendererPercentSignPrototype{\char`\%}%
\def\markdownRendererAmpersandPrototype{\&}%
\def\markdownRendererUnderscorePrototype{\char`_}%
\def\markdownRendererHashPrototype{\char`\#}%
\def\markdownRendererCircumflexPrototype{\char`^}%
\def\markdownRendererBackslashPrototype{\char`\\}%
\def\markdownRendererTildePrototype{\char`~}%
\def\markdownRendererPipePrototype{|}%
\def\markdownRendererCodeSpanPrototype#1{{\tt#1}}%
\def\markdownRendererLinkPrototype#1#2#3#4{#2}%
\def\markdownRendererContentBlockPrototype#1#2#3#4{%
  \markdownInput{#3}}%
\def\markdownRendererContentBlockOnlineImagePrototype{%
  \markdownRendererImage}%
\def\markdownRendererContentBlockCodePrototype#1#2#3#4#5{%
  \markdownRendererInputFencedCode{#3}{#2}}%
\def\markdownRendererImagePrototype#1#2#3#4{#2}%
\def\markdownRendererUlBeginPrototype{}%
```

```
\def\markdownRendererUlBeginTightPrototype{}%
\def\markdownRendererUlItemPrototype{}%
\def\markdownRendererUlItemEndPrototype{}%
\def\markdownRendererUlEndPrototype{}%
\def\markdownRendererUlEndTightPrototype{}%
\def\markdownRendererOlBeginPrototype{}%
\def\markdownRendererOlBeginTightPrototype{}%
\def\markdownRendererOlItemPrototype{}%
\def\markdownRendererOlItemWithNumberPrototype#1{}%
\def\markdownRendererOlItemEndPrototype{}%
\def\markdownRendererOlEndPrototype{}%
\def\markdownRendererOlEndTightPrototype{}%
\def\markdownRendererDlBeginPrototype{}%
\def\markdownRendererDlBeginTightPrototype{}%
\def\markdownRendererDlItemPrototype#1{#1}%
\def\markdownRendererDlItemEndPrototype{}%
\def\markdownRendererDlDefinitionBeginPrototype{}%
\def\markdownRendererDlDefinitionEndPrototype{\par}%
\def\markdownRendererDlEndPrototype{}%
\def\markdownRendererDlEndTightPrototype{}%
\def\markdownRendererEmphasisPrototype#1{{\it#1}}%
\def\markdownRendererStrongEmphasisPrototype#1{{\bf#1}}%
\def\markdownRendererBlockQuoteBeginPrototype{\par\begingroup\it}%
\def\markdownRendererBlockQuoteEndPrototype{\endgroup\par}%
\def\markdownRendererInputVerbatimPrototype#1{%
  \par{\tt\input#1\relax{}}\par}%
\def\markdownRendererInputFencedCodePrototype#1#2{%
  \markdownRendererInputVerbatimPrototype{#1}}%
\def\markdownRendererHeadingOnePrototype#1{#1}%
\def\markdownRendererHeadingTwoPrototype#1{#1}%
\def\markdownRendererHeadingThreePrototype#1{#1}%
\def\markdownRendererHeadingFourPrototype#1{#1}%
\def\markdownRendererHeadingFivePrototype#1{#1}%
\def\markdownRendererHeadingSixPrototype#1{#1}%
\def\markdownRendererHorizontalRulePrototype{}%
\def\markdownRendererFootnotePrototype#1{#1}%
\def\markdownRendererCitePrototype#1{}%
\def\markdownRendererTextCitePrototype#1{}%
```

3.2.4 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).

```
\def\markdownLuaOptions{{%
\ifx\markdownOptionBlankBeforeBlockquote\undefined\else
  blankBeforeBlockquote = \markdownOptionBlankBeforeBlockquote,
\fi
```

```
\ifx\markdownOptionBlankBeforeCodeFence\undefined\else
  blankBeforeCodeFence = \markdownOptionBlankBeforeCodeFence,
\fi
\ifx\markdownOptionBlankBeforeHeading\undefined\else
  blankBeforeHeading = \markdownOptionBlankBeforeHeading,
\fi
\ifx\markdownOptionBreakableBlockquotes\undefined\else
  breakableBlockquotes = \markdownOptionBreakableBlockquotes,
\fi
  cacheDir = "\markdownOptionCacheDir",
\ifx\markdownOptionCitations\undefined\else
  citations = \markdownOptionCitations,
\fi
\ifx\markdownOptionCitationNbsps\undefined\else
  citationNbsps = \markdownOptionCitationNbsps,
\fi
\ifx\markdownOptionCodeSpans\undefined\else
  codeSpans = \markdownOptionCodeSpans,
\fi
\ifx\markdownOptionContentBlocks\undefined\else
  contentBlocks = \markdownOptionContentBlocks,
\fi
\ifx\markdownOptionContentBlocksLanguageMap\undefined\else
  contentBlocksLanguageMap =
    "\markdownOptionContentBlocksLanguageMap",
\fi
\ifx\markdownOptionDefinitionLists\undefined\else
  definitionLists = \markdownOptionDefinitionLists,
\fi
\ifx\markdownOptionFinalizeCache\undefined\else
  finalizeCache = \markdownOptionFinalizeCache,
\fi
  frozenCacheFileName = "\markdownOptionFrozenCacheFileName",
  frozenCacheCounter = \the\markdownFrozenCacheCounter,
\ifx\markdownOptionFootnotes\undefined\else
  footnotes = \markdownOptionFootnotes,
\fi
\ifx\markdownOptionFencedCode\undefined\else
  fencedCode = \markdownOptionFencedCode,
\fi
\ifx\markdownOptionHashEnumerators\undefined\else
  hashEnumerators = \markdownOptionHashEnumerators,
\fi
\ifx\markdownOptionHeaderAttributes\undefined\else
  headerAttributes = \markdownOptionHeaderAttributes,
\fi
\ifx\markdownOptionHtml\undefined\else
```

```
  html = \markdownOptionHtml,
\fi
\ifx\markdownOptionHybrid\undefined\else
  hybrid = \markdownOptionHybrid,
\fi
\ifx\markdownOptionInlineFootnotes\undefined\else
  inlineFootnotes = \markdownOptionInlineFootnotes,
\fi
\ifx\markdownOptionPipeTables\undefined\else
  pipeTables = \markdownOptionPipeTables,
\fi
\ifx\markdownOptionPreserveTabs\undefined\else
  preserveTabs = \markdownOptionPreserveTabs,
\fi
\ifx\markdownOptionShiftHeadings\undefined\else
  shiftHeadings = "\markdownOptionShiftHeadings",
\fi
\ifx\markdownOptionSlice\undefined\else
  slice = "\markdownOptionSlice",
\fi
\ifx\markdownOptionSmartEllipses\undefined\else
  smartEllipses = \markdownOptionSmartEllipses,
\fi
\ifx\markdownOptionStartNumber\undefined\else
  startNumber = \markdownOptionStartNumber,
\fi
\ifx\markdownOptionStripIndent\undefined\else
  stripIndent = \markdownOptionStripIndent,
\fi
\ifx\markdownOptionTableCaptions\undefined\else
  tableCaptions = \markdownOptionTableCaptions,
\fi
\ifx\markdownOptionTeXComments\undefined\else
  texComments = \markdownOptionTeXComments,
\fi
\ifx\markdownOptionTightLists\undefined\else
  tightLists = \markdownOptionTightLists,
\fi
\ifx\markdownOptionUnderscores\undefined\else
  underscores = \markdownOptionUnderscores,
\fi}
}%
```

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

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

First, ensure that the \markdownOptionCacheDir directory exists.

```
  local lfs = require("lfs")
  local cacheDir = "\markdownOptionCacheDir"
  if not lfs.isdir(cacheDir) then
    assert(lfs.mkdir(cacheDir))
  end
```

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

```
  local md = require("markdown")
  local convert = md.new(\markdownLuaOptions)
}%
```

3.2.5 Buffering Markdown Input

The \markdownIfOption{⟨*name*⟩}{⟨*iftrue*⟩}{⟨*iffalse*⟩} macro is provided for testing, whether the value of \markdownOption⟨*name*⟩ is true. If the value is true, then ⟨*iftrue*⟩ is expanded, otherwise ⟨*iffalse*⟩ is expanded.

```
\def\markdownIfOption#1#2#3{%
  \begingroup
  \def\next{true}%
  \expandafter\ifx\csname markdownOption#1\endcsname\next
    \endgroup#2\else\endgroup#3\fi}%
```

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.

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

The \markdownReadAndConvertTab macro contains the tab character literal.

```
\begingroup
  \catcode`\^^I=12%
  \gdef\markdownReadAndConvertTab{^^I}%
\endgroup
```

The \markdownReadAndConvert macro is largely a rewrite of the LATEX 2ε \filecontents macro to plain TEX.

```
\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 enabled.

```
  \catcode`\^^M=13%
  \catcode`\^^I=13%
```

```
\catcode`|=0%
\catcode`\\=12%
|catcode`@=14%
|catcode`|%=12@
|gdef|markdownReadAndConvert#1#2{@
  |begingroup@
```

If we are not reading markdown documents from the frozen cache, open the `\markdownOptionInputTempFileName` file for writing.

```
    |markdownIfOption{FrozenCache}{}{@
      |immediate|openout|markdownOutputFileStream@
        |markdownOptionInputTempFileName|relax@
      |markdownInfo{Buffering markdown input into the temporary @
        input file "|markdownOptionInputTempFileName" and scanning @
        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.

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

The `\markdownReadAndConvertStripPercentSigns` macro will process the individual lines of output, stipping away leading percent signs (`%`) when `\markdownOptionStripPercentSigns` is enabled. 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.

```
    |def|markdownReadAndConvertStripPercentSign##1{@
      |markdownIfOption{StripPercentSigns}{@
        |if##1%@
          |expandafter|expandafter|expandafter@
            |markdownReadAndConvertProcessLine@
        |else@
          |expandafter|expandafter|expandafter@
            |markdownReadAndConvertProcessLine@
            |expandafter|expandafter|expandafter##1@
        |fi@
      }{@
        |expandafter@
          |markdownReadAndConvertProcessLine@
          |expandafter##1@
      }@
    }@
```

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.

```
5073      |def|markdownReadAndConvertProcessLine##1#1##2#1##3|relax{@
```

If we are not reading markdown documents from the frozen cache and the ending token sequence does not appear in the line, store the line in the \markdownOptionInputTempFileName file. If we are reading markdown documents from the frozen cache and the ending token sequence does not appear in the line, gobble the line.

```
5074        |ifx|relax##3|relax@
5075          |markdownIfOption{FrozenCache}{}{@
5076            |immediate|write|markdownOutputFileStream{##1}@
5077          }@
5078        |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.

```
5079          |def^^M{@
5080            |markdownInfo{The ending token sequence was found}@
5081            |markdownIfOption{FrozenCache}{}{@
5082              |immediate|closeout|markdownOutputFileStream@
5083            }@
5084            |endgroup@
5085            |markdownInput{@
5086              |markdownOptionOutputDir@
5087              /|markdownOptionInputTempFileName@
5088            }@
5089            #2}@
5090        |fi@
```

Repeat with the next line.

```
5091        ^^M}@
```

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

```
5092      |catcode`|^^I=13@
5093      |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.

```
5094      |catcode`|^^M=13@
5095      |def^^M##1^^M{@
```

```
      |def^^M####1^^M{@
        |markdownReadAndConvertStripPercentSign####1#1#1|relax}@
      ^^M}@
    ^^M}@
```

Reset the character categories back to the former state.

```
|endgroup
```

3.2.6 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 3.2.7 are meant to be indistinguishable to the remaining code.

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

```
\ifnum\markdownMode<2\relax
\ifnum\markdownMode=0\relax
  \markdownInfo{Using mode 0: Shell escape via write18}%
\else
  \markdownInfo{Using mode 1: Shell escape via os.execute}%
\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` (LuaTEX, PdfTEX) or the `\shellescape` (XeTEX) 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.

```
\ifx\pdfshellescape\undefined
  \ifx\shellescape\undefined
    \ifnum\markdownMode=0\relax
      \def\markdownExecuteShellEscape{1}%
    \else
      \def\markdownExecuteShellEscape{%
        \directlua{tex.sprint(status.shell_escape or "1")}}%
    \fi
  \else
    \let\markdownExecuteShellEscape\shellescape
  \fi
\else
  \let\markdownExecuteShellEscape\pdfshellescape
\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.

```
\ifnum\markdownMode=0\relax
  \def\markdownExecuteDirect#1{\immediate\write18{#1}}%
\else
  \def\markdownExecuteDirect#1{%
    \directlua{os.execute("\luaescapestring{#1}")}}%
\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.

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

```
\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.

```
  \catcode`|=0%
  \catcode`\\=12%
  |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.

```
    |immediate|openout|markdownOutputFileStream=%
      |markdownOptionHelperScriptFileName
    |markdownInfo{Writing a helper Lua script to the file
      "|markdownOptionHelperScriptFileName"}%
    |immediate|write|markdownOutputFileStream{%
      local ran_ok, error = pcall(function()
        local ran_ok, kpse = pcall(require, "kpse")
        if ran_ok then kpse.set_program_name("luatex") end
        #1
      end)
```

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

```
      if not ran_ok then
        local file = io.open("%
          |markdownOptionOutputDir
          /|markdownOptionErrorTempFileName", "w")
        if file then
          file:write(error .. "\n")
          file:close()
        end
        print('\\markdownError{An error was encountered while executing
               Lua code}{For further clues, examine the file
               "|markdownOptionOutputDir
               /|markdownOptionErrorTempFileName"}')
      end}%
    |immediate|closeout|markdownOutputFileStream
```

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

```
    |markdownInfo{Executing a helper Lua script from the file
      "|markdownOptionHelperScriptFileName" and storing the result in the
      file "|markdownOptionOutputTempFileName"}%
    |markdownExecute{texlua "|markdownOptionOutputDir
      /|markdownOptionHelperScriptFileName" > %
      "|markdownOptionOutputDir
      /|markdownOptionOutputTempFileName"}%
```

\input the generated \markdownOptionOutputTempFileName file.

```
    |input|markdownOptionOutputTempFileName|relax}%
|endgroup
```

3.2.7 Direct Lua Access

The following TEX code is intended for TEX engines that provide direct access to Lua (LuaTEX). The macro \markdownLuaExecute defined here and in Section 3.2.6 are meant to be indistinguishable to the remaining code. This corresponds to the \markdownMode value of 2.

```
\else
\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 3.2.6,

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

3.2.8 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.

```
\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.

```
  \catcode`|=0%
  \catcode`\\=12%
  |gdef|markdownInput#1{%
```

Change the category code of the percent sign (%) to other, so that a user of the hybrid Lua option or a malevolent actor can't produce TeX comments in the plain TeX output of the Markdown package.

```
    |begingroup
    |catcode`|%=12
```

If we are reading from the frozen cache, input it, expand the corresponding \markdownFrozenCache⟨*number*⟩ macro, and increment \markdownFrozenCacheCounter.

```
    |markdownIfOption{FrozenCache}{%
      |ifnum|markdownFrozenCacheCounter=0|relax
        |markdownInfo{Reading frozen cache from
          "|markdownOptionFrozenCacheFileName"}%
        |input|markdownOptionFrozenCacheFileName|relax
      |fi
      |markdownInfo{Including markdown document number
        "|the|markdownFrozenCacheCounter" from frozen cache}%
      |csname markdownFrozenCache|the|markdownFrozenCacheCounter|endcsname
      |global|advance|markdownFrozenCacheCounter by 1|relax
    }{%
      |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 LATEXMk to track changes to the markdown document.

```
      |openin|markdownInputFileStream#1
      |closein|markdownInputFileStream
      |markdownLuaExecute{%
        |markdownPrepare
        local file = assert(io.open("#1", "r"),
          [[could not open file "#1" for reading]])
        local input = assert(file:read("*a"))
        assert(file:close())
```

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.

```
5202          print(convert(input:gsub("\r\n?", "\n") .. "\n"))}%
```

If we are finalizing the frozen cache, increment `\markdownFrozenCacheCounter`.

```
5203        |markdownIfOption{FinalizeCache}{%
5204          |global|advance|markdownFrozenCacheCounter by 1|relax
5205        }%
5206      }%
5207      |endgroup
5208    }%
5209  |endgroup
```

3.3 LaTeX Implementation

The LaTeX implemenation makes use of the fact that, apart from some subtle differences, LaTeX implements the majority of the plain TeX format [10, Section 9]. As a consequence, we can directly reuse the existing plain TeX implementation.

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

```
5210 \newcommand\markdownInfo[1]{\PackageInfo{markdown}{#1}}%
5211 \newcommand\markdownWarning[1]{\PackageWarning{markdown}{#1}}%
5212 \newcommand\markdownError[2]{\PackageError{markdown}{#1}{#2.}}%
5213 \input markdown/markdown
5214 \def\markdownVersionSpace{ }%
5215 \ProvidesPackage{markdown}[\markdownLastModified\markdownVersionSpace v%
5216   \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.

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).

```
5217 \let\markdownInputPlainTeX\markdownInput
5218 \renewcommand\markdownInput[2][]{%
5219   \begingroup
5220     \markdownSetup{#1}%
5221     \markdownInputPlainTeX{#2}%
5222   \endgroup}%
```

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

```
\renewenvironment{markdown}{%
  \markdownReadAndConvert@markdown{}}{%
  \markdownEnd}%
\renewenvironment{markdown*}[1]{%
  \markdownSetup{#1}%
  \markdownReadAndConvert@markdown*}{%
  \markdownEnd}%
\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*.

```
  \catcode`\|=0\catcode`\<=1\catcode`\>=2%
  \catcode`\\=12|catcode`|{=12|catcode`|}=12%
  |gdef|markdownReadAndConvert@markdown#1<%
    |markdownReadAndConvert<\end{markdown#1}>%
                           <|end<markdown#1>>>%
|endgroup
```

3.3.3 Options

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

```
\DeclareOption*{%
  \expandafter\markdownSetup\expandafter{\CurrentOption}}%
\ProcessOptions\relax
```

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

```
\define@key{markdownOptions}{renderers}{%
  \setkeys{markdownRenderers}{#1}%
  \def\KV@prefix{KV@markdownOptions@}}%
\define@key{markdownOptions}{rendererPrototypes}{%
  \setkeys{markdownRendererPrototypes}{#1}%
  \def\KV@prefix{KV@markdownOptions@}}%
```

3.3.3.1 LaTeX Themes

This section implements example themes provided with the Markdown package.

The `witiko/dot` theme enables the `fencedCode` Lua option:

```
\markdownSetup{fencedCode}%
```

We store the previous definition of the fenced code token renderer prototype:

```
\let\markdown@witiko@dot@oldRendererInputFencedCodePrototype
  \markdownRendererInputFencedCodePrototype
```

If the infostring starts with `dot …`, we redefine the fenced code block token renderer prototype, so that it typesets the code block via Graphviz tools if and only if the `\markdownOptionFrozenCache` plain TEX option is disabled and the code block has not been previously typeset:

```
\RequirePackage{ifthen}
\renewcommand\markdownRendererInputFencedCode[2]{%
  \def\next##1 ##2\relax{%
    \ifthenelse{\equal{##1}{dot}}{%
      \markdownIfOption{FrozenCache}{}{%
        \immediate\write18{%
          if ! test -e #1.pdf.source || ! diff #1 #1.pdf.source;
          then
            dot -Tpdf -o #1.pdf #1;
            cp #1 #1.pdf.source;
          fi}}%
```

We include the typeset image using the image token renderer:

```
      \markdownRendererImage{Graphviz image}{#1.pdf}{#1.pdf}{##2}%
```

If the infostring does not start with `dot …`, we use the previous definition of the fenced code token renderer prototype:

```
    }{%
      \markdown@witiko@dot@oldRendererInputFencedCodePrototype{#1}{#2}%
    }%
  }%
  \next#2 \relax}%
```

The `witiko/graphicx/http` theme stores the previous definition of the image token renderer prototype:

```
\let\markdown@witiko@graphicx@http@oldRendererImagePrototype
  \markdownRendererImagePrototype
```

We define the `\markdown@witiko@graphicx@http@counter` counter to enumerate the images for caching and the `\markdown@witiko@graphicx@http@filename` command, which will store the pathname of the file containing the pathname of the downloaded image file.

```
\newcount\markdown@witiko@graphicx@http@counter
\markdown@witiko@graphicx@http@counter=0
\newcommand\markdown@witiko@graphicx@http@filename{%
  \markdownOptionCacheDir/witiko_graphicx_http%
  .\the\markdown@witiko@graphicx@http@counter}%
```

We define the `\markdown@witiko@graphicx@http@download` command, which will receive two arguments that correspond to the URL of the online image and to the pathname, where the online image should be downloaded. The command will produce a shell command that tries to downloads the online image to the pathname.

```
\newcommand\markdown@witiko@graphicx@http@download[2]{%
  wget -O #2 #1 || curl --location -o #2 #1 || rm -f #2}
```

We locally swap the category code of the percentage sign with the line feed control character, so that we can use percentage signs in the shell code:

```
\begingroup
\catcode`\%=12
\catcode`\^^A=14
```

We redefine the image token renderer prototype, so that it tries to download an online image.

```
\global\def\markdownRendererImagePrototype#1#2#3#4{^^A
  \begingroup
    \edef\filename{\markdown@witiko@graphicx@http@filename}^^A
```

The image will be downloaded only if the image URL has the http or https protocols and the \markdownOptionFrozenCache plain TEX option is disabled:

```
    \markdownIfOption{FrozenCache}{}{^^A
      \immediate\write18{^^A
        if printf '%s' "#3" | grep -q -E '^https?:';
        then
```

The image will be downloaded to the pathname \markdownOptionCacheDir/⟨*the MD5 digest of the image URL*⟩.⟨*the suffix of the image URL*⟩:

```
          OUTPUT_PREFIX="\markdownOptionCacheDir";
          OUTPUT_BODY="$(printf '%s' '#3' | md5sum | cut -d' ' -f1)";
          OUTPUT_SUFFIX="$(printf '%s' '#3' | sed 's/.*[.]//')";
          OUTPUT="$OUTPUT_PREFIX/$OUTPUT_BODY.$OUTPUT_SUFFIX";
```

The image will be downloaded only if it has not already been downloaded:

```
          if ! [ -e "$OUTPUT" ];
          then
            \markdown@witiko@graphicx@http@download{'#3'}{"$OUTPUT"};
            printf '%s' "$OUTPUT" > "\filename";
          fi;
```

If the image does not have the http or https protocols or the image has already been downloaded, the URL will be stored as-is:

```
        else
          printf '%s' '#3' > "\filename";
        fi}}^^A
```

We load the pathname of the downloaded image and we typeset the image using the previous definition of the image renderer prototype:

```
    \CatchFileDef{\filename}{\filename}{}^^A
    \markdown@witiko@graphicx@http@oldRendererImagePrototype^^A
      {#1}{#2}{\filename}{#4}^^A
  \endgroup
  \global\advance\markdown@witiko@graphicx@http@counter by 1\relax}^^A
\endgroup
```

The witiko/tilde theme redefines the tilde token renderer prototype, so that it expands to a non-breaking space:

```
\renewcommand\markdownRendererTildePrototype{~}%
```

3.3.4 Token Renderer Prototypes

The following configuration should be considered placeholder. If the plain package option has been enabled (see Section 2.3.2.1), none of it will take effect.

```
\ifmarkdownLaTeXPlain\else
```

If the \markdownOptionTightLists macro expands to false, do not load the paralist package. This is necessary for LaTeX 2ε 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.

```
\RequirePackage{ifthen}
\@ifundefined{markdownOptionTightLists}{%
  \@ifclassloaded{beamer}{}{%
    \RequirePackage{paralist}}%
}{%
  \ifthenelse{\equal{\markdownOptionTightLists}{false}}{}{%
    \RequirePackage{paralist}}%
}%
```

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.

```
\@ifpackageloaded{paralist}{
  \markdownSetup{rendererPrototypes={
    ulBeginTight = {\begin{compactitem}},
    ulEndTight = {\end{compactitem}},
    olBeginTight = {\begin{compactenum}},
    olEndTight = {\end{compactenum}},
    dlBeginTight = {\begin{compactdesc}},
    dlEndTight = {\end{compactdesc}}}}
}{
  \markdownSetup{rendererPrototypes={
    ulBeginTight = {\markdownRendererUlBegin},
    ulEndTight = {\markdownRendererUlEnd},
    olBeginTight = {\markdownRendererOlBegin},
    olEndTight = {\markdownRendererOlEnd},
    dlBeginTight = {\markdownRendererDlBegin},
    dlEndTight = {\markdownRendererDlEnd}}}}
\RequirePackage{csvsimple}
\RequirePackage{fancyvrb}
\RequirePackage{graphicx}
```

```
\markdownSetup{rendererPrototypes={
  lineBreak = {\\},
  leftBrace = {\textbraceleft},
  rightBrace = {\textbraceright},
  dollarSign = {\textdollar},
  underscore = {\textunderscore},
  circumflex = {\textasciicircum},
  backslash = {\textbackslash},
  tilde = {\textasciitilde},
  pipe = {\textbar},
  codeSpan = {\texttt{#1}},
  contentBlock = {%
    \ifthenelse{\equal{#1}{csv}}{%
      \begin{table}%
        \begin{center}%
          \csvautotabular{#3}%
        \end{center}
        \ifx\empty#4\empty\else
          \caption{#4}%
        \fi
      \end{table}}{%
      \markdownInput{#3}}},
  image = {%
    \begin{figure}%
      \begin{center}%
        \includegraphics{#3}%
      \end{center}%
      \ifx\empty#4\empty\else
        \caption{#4}%
      \fi
      \label{fig:#1}%
    \end{figure}},
  ulBegin = {\begin{itemize}},
  ulItem = {\item{}},
  ulEnd = {\end{itemize}},
  olBegin = {\begin{enumerate}},
  olItem = {\item{}},
  olItemWithNumber = {\item[#1.]},
  olEnd = {\end{enumerate}},
  dlBegin = {\begin{description}},
  dlItem = {\item[#1]},
  dlEnd = {\end{description}},
  emphasis = {\emph{#1}},
  blockQuoteBegin = {\begin{quotation}},
  blockQuoteEnd = {\end{quotation}},
  inputVerbatim = {\VerbatimInput{#1}},
  inputFencedCode = {%
```

```
    \ifx\relax#2\relax
      \VerbatimInput{#1}%
    \else
      \@ifundefined{minted@code}{%
        \@ifundefined{lst@version}{%
          \markdownRendererInputFencedCode{#1}{}%
```

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

```
        }{%
          \lstinputlisting[language=#2]{#1}%
        }%
```

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

```
      }{%
        \inputminted{#2}{#1}%
      }%
    \fi},
  horizontalRule = {\noindent\rule[0.5ex]{\linewidth}{1pt}},
  footnote = {\footnote{#1}}}}
```

Support the nesting of strong emphasis.

```
\def\markdownLATEXStrongEmphasis#1{%
  \IfSubStr\f@series{b}{\textnormal{#1}}{\textbf{#1}}}
\markdownSetup{rendererPrototypes={strongEmphasis={%
  \protect\markdownLATEXStrongEmphasis{#1}}}}
```

Support LaTeX document classes that do not provide chapters.

```
\@ifundefined{chapter}{%
  \markdownSetup{rendererPrototypes = {
    headingOne = {\section{#1}},
    headingTwo = {\subsection{#1}},
    headingThree = {\subsubsection{#1}},
    headingFour = {\paragraph{#1}\leavevmode},
    headingFive = {\subparagraph{#1}\leavevmode}}}
}{%
  \markdownSetup{rendererPrototypes = {
    headingOne = {\chapter{#1}},
    headingTwo = {\section{#1}},
    headingThree = {\subsection{#1}},
    headingFour = {\subsubsection{#1}},
    headingFive = {\paragraph{#1}\leavevmode},
    headingSix = {\subparagraph{#1}\leavevmode}}}
}%
```

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

```
\newcount\markdownLaTeXCitationsCounter

% Basic implementation
\RequirePackage{gobble}
\def\markdownLaTeXBasicCitations#1#2#3#4#5#6{%
  \advance\markdownLaTeXCitationsCounter by 1\relax
  \ifx\relax#4\relax
    \ifx\relax#5\relax
      \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
        \cite{#1#2#6}%  Without prenotes and postnotes, just accumulate cites
        \expandafter\expandafter\expandafter
        \expandafter\expandafter\expandafter\expandafter
        \@gobblethree
      \fi
    \else%  Before a postnote (#5), dump the accumulator
      \ifx\relax#1\relax\else
        \cite{#1}%
      \fi
      \cite[#5]{#6}%
      \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
      \else
        \expandafter\expandafter\expandafter
        \expandafter\expandafter\expandafter\expandafter
        \expandafter\expandafter\expandafter
        \expandafter\expandafter\expandafter\expandafter
        \markdownLaTeXBasicCitations
      \fi
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter{%
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter}%
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter{%
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter}%
      \expandafter\expandafter\expandafter
      \@gobblethree
    \fi
  \else%  Before a prenote (#4), dump the accumulator
    \ifx\relax#1\relax\else
      \cite{#1}%
    \fi
    \ifnum\markdownLaTeXCitationsCounter>1\relax
      \space  % Insert a space before the prenote in later citations
    \fi
    #4~\expandafter\cite\ifx\relax#5\relax{#6}\else[#5]{#6}\fi
    \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
```

```
    \else
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter
      \markdownLaTeXBasicCitations
    \fi
    \expandafter\expandafter\expandafter{%
    \expandafter\expandafter\expandafter}%
    \expandafter\expandafter\expandafter{%
    \expandafter\expandafter\expandafter}%
    \expandafter
    \@gobblethree
  \fi\markdownLaTeXBasicCitations{#1#2#6},}
\let\markdownLaTeXBasicTextCitations\markdownLaTeXBasicCitations

% Natbib implementation
\def\markdownLaTeXNatbibCitations#1#2#3#4#5{%
  \advance\markdownLaTeXCitationsCounter by 1\relax
  \ifx\relax#3\relax
    \ifx\relax#4\relax
      \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
        \citep{#1,#5}%  Without prenotes and postnotes, just accumulate cites
        \expandafter\expandafter\expandafter
        \expandafter\expandafter\expandafter\expandafter
        \@gobbletwo
      \fi
    \else%  Before a postnote (#4), dump the accumulator
      \ifx\relax#1\relax\else
        \citep{#1}%
      \fi
      \citep[][#4]{#5}%
      \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
      \else
        \expandafter\expandafter\expandafter
        \expandafter\expandafter\expandafter\expandafter
        \expandafter\expandafter\expandafter
        \expandafter\expandafter\expandafter\expandafter
        \markdownLaTeXNatbibCitations
      \fi
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter{%
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter}%
      \expandafter\expandafter\expandafter
      \@gobbletwo
    \fi
  \else%  Before a prenote (#3), dump the accumulator
    \ifx\relax#1\relax\relax\else
```

```
      \citep{#1}%
    \fi
    \citep[#3][#4]{#5}%
    \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
    \else
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter
      \markdownLaTeXNatbibCitations
    \fi
    \expandafter\expandafter\expandafter{%
    \expandafter\expandafter\expandafter}%
    \expandafter
    \@gobbletwo
  \fi\markdownLaTeXNatbibCitations{#1,#5}}
\def\markdownLaTeXNatbibTextCitations#1#2#3#4#5{%
  \advance\markdownLaTeXCitationsCounter by 1\relax
  \ifx\relax#3\relax
    \ifx\relax#4\relax
      \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
        \citet{#1,#5}%  Without prenotes and postnotes, just accumulate cites
        \expandafter\expandafter\expandafter
        \expandafter\expandafter\expandafter\expandafter
        \@gobbletwo
      \fi
    \else%  After a prenote or a postnote, dump the accumulator
      \ifx\relax#1\relax\else
        \citet{#1}%
      \fi
      , \citet[#3][#4]{#5}%
      \ifnum\markdownLaTeXCitationsCounter<\markdownLaTeXCitationsTotal\relax
        ,
      \else
        \ifnum\markdownLaTeXCitationsCounter=\markdownLaTeXCitationsTotal\relax
          ,
        \fi
      \fi
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter
      \markdownLaTeXNatbibTextCitations
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter{%
      \expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\expandafter}%
      \expandafter\expandafter\expandafter
      \@gobbletwo
    \fi
  \else%  After a prenote or a postnote, dump the accumulator
```

```
    \ifx\relax#1\relax\relax\else
      \citet{#1}%
    \fi
    , \citet[#3][#4]{#5}%
    \ifnum\markdownLaTeXCitationsCounter<\markdownLaTeXCitationsTotal\relax
      ,
    \else
      \ifnum\markdownLaTeXCitationsCounter=\markdownLaTeXCitationsTotal\relax
        ,
      \fi
    \fi
    \expandafter\expandafter\expandafter
    \markdownLaTeXNatbibTextCitations
    \expandafter\expandafter\expandafter{%
    \expandafter\expandafter\expandafter}%
    \expandafter
    \@gobbletwo
  \fi\markdownLaTeXNatbibTextCitations{#1,#5}}

% BibLaTeX implementation
\def\markdownLaTeXBibLaTeXCitations#1#2#3#4#5{%
  \advance\markdownLaTeXCitationsCounter by 1\relax
  \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
    \autocites#1[#3][#4]{#5}%
    \expandafter\@gobbletwo
  \fi\markdownLaTeXBibLaTeXCitations{#1[#3][#4]{#5}}}
\def\markdownLaTeXBibLaTeXTextCitations#1#2#3#4#5{%
  \advance\markdownLaTeXCitationsCounter by 1\relax
  \ifnum\markdownLaTeXCitationsCounter>\markdownLaTeXCitationsTotal\relax
    \textcites#1[#3][#4]{#5}%
    \expandafter\@gobbletwo
  \fi\markdownLaTeXBibLaTeXTextCitations{#1[#3][#4]{#5}}}

\markdownSetup{rendererPrototypes = {
  cite = {%
    \markdownLaTeXCitationsCounter=1%
    \def\markdownLaTeXCitationsTotal{#1}%
    \@ifundefined{autocites}{%
      \@ifundefined{citep}{%
        \expandafter\expandafter\expandafter
        \markdownLaTeXBasicCitations
        \expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter}%
        \expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter}%
      }{%
        \expandafter\expandafter\expandafter
```

```
        \markdownLaTeXNatbibCitations
        \expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter}%
      }%
    }{%
      \expandafter\expandafter\expandafter
      \markdownLaTeXBibLaTeXCitations
      \expandafter{\expandafter}%
    }},
  textCite = {%
    \markdownLaTeXCitationsCounter=1%
    \def\markdownLaTeXCitationsTotal{#1}%
    \@ifundefined{autocites}{%
      \@ifundefined{citep}{%
        \expandafter\expandafter\expandafter
        \markdownLaTeXBasicTextCitations
        \expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter}%
        \expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter}%
      }{%
        \expandafter\expandafter\expandafter
        \markdownLaTeXNatbibTextCitations
        \expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter}%
      }%
    }{%
      \expandafter\expandafter\expandafter
      \markdownLaTeXBibLaTeXTextCitations
      \expandafter{\expandafter}%
    }}}}
```

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.

```
\RequirePackage{url}
\def\markdownRendererLinkPrototype{%
  \begingroup
  \catcode`\#=12
  \def\next##1##2##3##4{%
    ##1\footnote{%
      \ifx\empty##4\empty\else##4: \fi\texttt<\url{##3}\texttt>}%
    \endgroup}%
  \next}
```

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

```
\newcount\markdownLaTeXRowCounter
```

```
\newcount\markdownLaTeXRowTotal
\newcount\markdownLaTeXColumnCounter
\newcount\markdownLaTeXColumnTotal
\newtoks\markdownLaTeXTable
\newtoks\markdownLaTeXTableAlignment
\newtoks\markdownLaTeXTableEnd
\@ifpackageloaded{booktabs}{
  \let\markdownLaTeXTopRule\toprule
  \let\markdownLaTeXMidRule\midrule
  \let\markdownLaTeXBottomRule\bottomrule
}{
  \let\markdownLaTeXTopRule\hline
  \let\markdownLaTeXMidRule\hline
  \let\markdownLaTeXBottomRule\hline
}
\markdownSetup{rendererPrototypes={
  table = {%
    \markdownLaTeXTable={}%
    \markdownLaTeXTableAlignment={}%
    \markdownLaTeXTableEnd={%
      \markdownLaTeXBottomRule
      \end{tabular}}%
    \ifx\empty#1\empty\else
      \addto@hook\markdownLaTeXTable{%
        \begin{table}
        \centering}%
      \addto@hook\markdownLaTeXTableEnd{%
        \caption{#1}
        \end{table}}%
    \fi
    \addto@hook\markdownLaTeXTable{\begin{tabular}}%
    \markdownLaTeXRowCounter=0%
    \markdownLaTeXRowTotal=#2%
    \markdownLaTeXColumnTotal=#3%
    \markdownLaTeXRenderTableRow
  }
}}
\def\markdownLaTeXRenderTableRow#1{%
  \markdownLaTeXColumnCounter=0%
  \ifnum\markdownLaTeXRowCounter=0\relax
    \markdownLaTeXReadAlignments#1%
    \markdownLaTeXTable=\expandafter\expandafter\expandafter{%
      \expandafter\the\expandafter\markdownLaTeXTable\expandafter{%
        \the\markdownLaTeXTableAlignment}}%
    \addto@hook\markdownLaTeXTable{\markdownLaTeXTopRule}%
  \else
    \markdownLaTeXRenderTableCell#1%
```

```
  \fi
  \ifnum\markdownLaTeXRowCounter=1\relax
    \addto@hook\markdownLaTeXTable\markdownLaTeXMidRule
  \fi
  \advance\markdownLaTeXRowCounter by 1\relax
  \ifnum\markdownLaTeXRowCounter>\markdownLaTeXRowTotal\relax
    \the\markdownLaTeXTable
    \the\markdownLaTeXTableEnd
    \expandafter\@gobble
  \fi\markdownLaTeXRenderTableRow}
\def\markdownLaTeXReadAlignments#1{%
  \advance\markdownLaTeXColumnCounter by 1\relax
  \if#1d%
    \addto@hook\markdownLaTeXTableAlignment{l}%
  \else
    \addto@hook\markdownLaTeXTableAlignment{#1}%
  \fi
  \ifnum\markdownLaTeXColumnCounter<\markdownLaTeXColumnTotal\relax\else
    \expandafter\@gobble
  \fi\markdownLaTeXReadAlignments}
\def\markdownLaTeXRenderTableCell#1{%
  \advance\markdownLaTeXColumnCounter by 1\relax
  \ifnum\markdownLaTeXColumnCounter<\markdownLaTeXColumnTotal\relax
    \addto@hook\markdownLaTeXTable{#1&}%
  \else
    \addto@hook\markdownLaTeXTable{#1\\}%
    \expandafter\@gobble
  \fi\markdownLaTeXRenderTableCell}
\fi
```

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.

```
\newcommand\markdownMakeOther{%
  \count0=128\relax
  \loop
    \catcode\count0=11\relax
    \advance\count0 by 1\relax
  \ifnum\count0<256\repeat}%
```

3.4 ConTEXt Implementation

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

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

```
\def\markdownInfo#1{\writestatus{markdown}{#1.}}%
\def\markdownWarning#1{\writestatus{markdown\space warn}{#1.}}%
\def\dospecials{\do\ \do\\\do\{\do\}\do\$\do\&%
  \do\#\do\^\do\_\do\%\do\~}%
\input markdown/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 LATEX package.

```
\def\markdownMakeOther{%
  \count0=128\relax
  \loop
    \catcode\count0=11\relax
    \advance\count0 by 1\relax
  \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 ConTEXt.

```
  \catcode`|=12}%
```

3.4.1 Typesetting Markdown

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

```
\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*.

```
  \catcode`\|=0%
  \catcode`\\=12%
  |gdef|startmarkdown{%
    |markdownReadAndConvert{\stopmarkdown}%
                           {|stopmarkdown}}%
  |gdef|stopmarkdown{|markdownEnd}%
|endgroup
```

3.4.2 Token Renderer Prototypes

The following configuration should be considered placeholder.

```
\def\markdownRendererLineBreakPrototype{\blank}%
\def\markdownRendererLeftBracePrototype{\textbraceleft}%
\def\markdownRendererRightBracePrototype{\textbraceright}%
\def\markdownRendererDollarSignPrototype{\textdollar}%
\def\markdownRendererPercentSignPrototype{\percent}%
\def\markdownRendererUnderscorePrototype{\textunderscore}%
\def\markdownRendererCircumflexPrototype{\textcircumflex}%
\def\markdownRendererBackslashPrototype{\textbackslash}%
\def\markdownRendererTildePrototype{\textasciitilde}%
\def\markdownRendererPipePrototype{\char`|}%
\def\markdownRendererLinkPrototype#1#2#3#4{%
  \useURL[#1][#3][][#4]#1\footnote[#1]{\ifx\empty#4\empty\else#4:
  \fi\tt<\hyphenatedurl{#3}>}}%
\usemodule[database]
\defineseparatedlist
  [MarkdownConTeXtCSV]
  [separator={,},
   before=\bTABLE,after=\eTABLE,
   first=\bTR,last=\eTR,
   left=\bTD,right=\eTD]
\def\markdownConTeXtCSV{csv}
\def\markdownRendererContentBlockPrototype#1#2#3#4{%
  \def\markdownConTeXtCSV@arg{#1}%
  \ifx\markdownConTeXtCSV@arg\markdownConTeXtCSV
    \placetable[][tab:#1]{#4}{%
      \processseparatedfile[MarkdownConTeXtCSV][#3]}%
  \else
    \markdownInput{#3}%
  \fi}%
\def\markdownRendererImagePrototype#1#2#3#4{%
  \placefigure[][fig:#1]{#4}{\externalfigure[#3]}}%
\def\markdownRendererUlBeginPrototype{\startitemize}%
\def\markdownRendererUlBeginTightPrototype{\startitemize[packed]}%
\def\markdownRendererUlItemPrototype{\item}%
\def\markdownRendererUlEndPrototype{\stopitemize}%
\def\markdownRendererUlEndTightPrototype{\stopitemize}%
\def\markdownRendererOlBeginPrototype{\startitemize[n]}%
\def\markdownRendererOlBeginTightPrototype{\startitemize[packed,n]}%
\def\markdownRendererOlItemPrototype{\item}%
\def\markdownRendererOlItemWithNumberPrototype#1{\sym{#1.}}%
\def\markdownRendererOlEndPrototype{\stopitemize}%
\def\markdownRendererOlEndTightPrototype{\stopitemize}%
\definedescription
  [MarkdownConTeXtDlItemPrototype]
```

```
  [location=hanging,
   margin=standard,
   headstyle=bold]%
\definestartstop
  [MarkdownConTeXtDlPrototype]
  [before=\blank,
   after=\blank]%
\definestartstop
  [MarkdownConTeXtDlTightPrototype]
  [before=\blank\startpacked,
   after=\stoppacked\blank]%
\def\markdownRendererDlBeginPrototype{%
  \startMarkdownConTeXtDlPrototype}%
\def\markdownRendererDlBeginTightPrototype{%
  \startMarkdownConTeXtDlTightPrototype}%
\def\markdownRendererDlItemPrototype#1{%
  \startMarkdownConTeXtDlItemPrototype{#1}}%
\def\markdownRendererDlItemEndPrototype{%
  \stopMarkdownConTeXtDlItemPrototype}%
\def\markdownRendererDlEndPrototype{%
  \stopMarkdownConTeXtDlPrototype}%
\def\markdownRendererDlEndTightPrototype{%
  \stopMarkdownConTeXtDlTightPrototype}%
\def\markdownRendererEmphasisPrototype#1{{\em#1}}%
\def\markdownRendererStrongEmphasisPrototype#1{{\bf#1}}%
\def\markdownRendererBlockQuoteBeginPrototype{\startquotation}%
\def\markdownRendererBlockQuoteEndPrototype{\stopquotation}%
\def\markdownRendererInputVerbatimPrototype#1{\typefile{#1}}%
\def\markdownRendererInputFencedCodePrototype#1#2{%
  \ifx\relax#2\relax
    \typefile{#1}%
  \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.
\definetyping [latex]
\setuptyping  [latex] [option=TEX]

\starttext
  \startmarkdown
~~~ latex
\documentclass{article}
\begin{document}
  Hello world!
```

```
\end{document}
~~~
  \stopmarkdown
\stoptext
```

```
    \typefile[#2][]{#1}%
  \fi}%
\def\markdownRendererHeadingOnePrototype#1{\chapter{#1}}%
\def\markdownRendererHeadingTwoPrototype#1{\section{#1}}%
\def\markdownRendererHeadingThreePrototype#1{\subsection{#1}}%
\def\markdownRendererHeadingFourPrototype#1{\subsubsection{#1}}%
\def\markdownRendererHeadingFivePrototype#1{\subsubsubsection{#1}}%
\def\markdownRendererHeadingSixPrototype#1{\subsubsubsubsection{#1}}%
\def\markdownRendererHorizontalRulePrototype{%
  \blackrule[height=1pt, width=\hsize]}%
\def\markdownRendererFootnotePrototype#1{\footnote{#1}}%
\stopmodule\protect
```

There is a basic implementation of tables.

```
\newcount\markdownConTeXtRowCounter
\newcount\markdownConTeXtRowTotal
\newcount\markdownConTeXtColumnCounter
\newcount\markdownConTeXtColumnTotal
\newtoks\markdownConTeXtTable
\newtoks\markdownConTeXtTableFloat
\def\markdownRendererTablePrototype#1#2#3{%
  \markdownConTeXtTable={}%
  \ifx\empty#1\empty
    \markdownConTeXtTableFloat={%
      \the\markdownConTeXtTable}%
  \else
    \markdownConTeXtTableFloat={%
      \placetable{#1}{\the\markdownConTeXtTable}}%
  \fi
  \begingroup
  \setupTABLE[r][each][topframe=off, bottomframe=off, leftframe=off, rightframe=off]
  \setupTABLE[c][each][topframe=off, bottomframe=off, leftframe=off, rightframe=off]
  \setupTABLE[r][1][topframe=on, bottomframe=on]
  \setupTABLE[r][#1][bottomframe=on]
  \markdownConTeXtRowCounter=0%
  \markdownConTeXtRowTotal=#2%
  \markdownConTeXtColumnTotal=#3%
  \markdownConTeXtRenderTableRow}
\def\markdownConTeXtRenderTableRow#1{%
  \markdownConTeXtColumnCounter=0%
  \ifnum\markdownConTeXtRowCounter=0\relax
    \markdownConTeXtReadAlignments#1%
```

```
    \markdownConTeXtTable={\bTABLE}%
  \else
    \markdownConTeXtTable=\expandafter{%
      \the\markdownConTeXtTable\bTR}%
    \markdownConTeXtRenderTableCell#1%
    \markdownConTeXtTable=\expandafter{%
      \the\markdownConTeXtTable\eTR}%
  \fi
  \advance\markdownConTeXtRowCounter by 1\relax
  \ifnum\markdownConTeXtRowCounter>\markdownConTeXtRowTotal\relax
    \markdownConTeXtTable=\expandafter{%
      \the\markdownConTeXtTable\eTABLE}%
    \the\markdownConTeXtTableFloat
    \endgroup
    \expandafter\gobbleoneargument
  \fi\markdownConTeXtRenderTableRow}
\def\markdownConTeXtReadAlignments#1{%
  \advance\markdownConTeXtColumnCounter by 1\relax
  \if#1d%
    \setupTABLE[c][\the\markdownConTeXtColumnCounter][align=right]
  \fi\if#1l%
    \setupTABLE[c][\the\markdownConTeXtColumnCounter][align=right]
  \fi\if#1c%
    \setupTABLE[c][\the\markdownConTeXtColumnCounter][align=middle]
  \fi\if#1r%
    \setupTABLE[c][\the\markdownConTeXtColumnCounter][align=left]
  \fi
  \ifnum\markdownConTeXtColumnCounter<\markdownConTeXtColumnTotal\relax\else
    \expandafter\gobbleoneargument
  \fi\markdownConTeXtReadAlignments}
\def\markdownConTeXtRenderTableCell#1{%
  \advance\markdownConTeXtColumnCounter by 1\relax
  \markdownConTeXtTable=\expandafter{%
    \the\markdownConTeXtTable\bTD#1\eTD}%
  \ifnum\markdownConTeXtColumnCounter<\markdownConTeXtColumnTotal\relax\else
    \expandafter\gobbleoneargument
  \fi\markdownConTeXtRenderTableCell}
```

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] Till Tantau, Joseph Wright, and Vedran Miletić. *The Beamer class*. Feb. 10, 2021. URL: http://mirrors.ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf (visited on 02/11/2021).

[7] Vít Novotný. *LaTeX 2ε no longer keys packages by pathnames*. Feb. 20, 2021. URL: https://github.com/latex3/latex2e/issues/510 (visited on 02/21/2021).

[8] Geoffrey M. Poore. *The minted Package. Highlighted source code in LaTeX*. July 19, 2017. URL: http://mirrors.ctan.org/macros/latex/contrib/minted/minted.pdf (visited on 09/01/2020).

[9] Roberto Ierusalimschy. *Programming in Lua*. 3rd ed. Rio de Janeiro: PUC-Rio, 2013. xviii, 347. ISBN: 978-85-903798-5-0.

[10] Johannes Braams et al. *The LaTeX 2ε Sources*. Apr. 15, 2017. URL: http://mirrors.ctan.org/macros/latex/base/source2e.pdf (visited on 01/08/2018).