Spine, Doc Reform
-
SiSU Markup

Ralph Amissah

2008-05-22

Spine, Doc Reform - SiSU Markup 1

Ralph Amissah

SiSU Description 2

1. SiSU Description 3

SiSU is an object-centric, lightweight markup based, document structuring, parser, publishing and search tool for document collections. It is command line oriented and generates static content that is currently made searchable at an object level through an SQL database. Markup helps define (delineate) objects (primarily various types of text block) which are tracked in sequence, substantive objects being numbered sequentially by the program for object citation. 4

Summary. An object is a unit of text within a document the most common being a paragraph. Objects include individual headings, paragraphs, tables, grouped text of various types such as code blocks and within poems, verse. Objects have properties and attributes, of particular significance are headings and their levels which provide document structure. A heading is an object with a heirarchical value, that conceptually contains other objects (such as paragraphs and possibly sub-headings etc.). Objects are tracked sequentially as they relate to each other object within a document and substantive objects are numbered sequentially, for citation purposes. Notably footnotes are not objects in themselves, rather belonging to the object from which they are referenced, and following their own numbering sequence. From heading objects (linked) tables of content may be generated, and if additional metadata is provided book type indexes can be generated that link back to the objects to which they relate. 5

Unpacking this a bit further. SiSU as a concept independent of its markup language and the parsers that have been implemented, is based on the following ideas: 6

Object-Centricity. On objects: In SiSU objects are the fundamental unit from which larger constructs within a document and the document itself is built. Breaking the document into objects provides interesting possibilities. 7

Objects are fundamental building blocks: Conceptually within SiSU, objects are the building blocks or individual units of construction of a document. Objects are usually blocks of text, the most common of which is the paragraph, other examples include: individual headings, tables, grouped text of various types which include code blocks and verse within poems, ... and as mentioned an object could also, for example, be an image. Objects can be formatted and placed as needed, providing flexibility and enabling multiple types of representation across disperate formats and text recepticle, examples including html, epub, latex (in the past mind-maps) and sql (populated at an object level, and thereby providing search with that degree of granularity). 8

Sequential. Objects have sequence: That objects have sequence, goes largely without saying, this follows authorship, it is part of the definition of a document and how a document is written to convey meaning. 9

Object Numbers & Citation. Substantive objects are numbered for citation purposes: Most objects within a document are meant by the author to be a substantive part of the document. All such objects are numbered sequentially and can be referenced thereby for citation purposes. Object numbers provide the possibility of citing/locating text precisely across different document formats and different languages (assuming the document has been translated). For search it also makes it 10 possible to identify precisely where search criteria is met within in each document in the form of an index or to view those precise text objects before deciding which documents are of interest. Additionally the use of objects (and that objects are numbered) frees the possibility to represent the document in the manner considered most suitable to a specific document format wilst retaining its structural (and citation) integrity).

Characteristics. Objects have properties and attributes: Objects have properties (and may have attributes). By properties I here refer to the fundamental type of object, be it a heading, a paragraph, table, verse etc. Attributes extend further and may include other things that one might wish to associate with the object (examples not necessarily currently available/ implemented in SiSU might include, formatting whether it is indented, or metadata e.g. the associated language, or programming language for a code block) 11

Document structure. Heading objects hold documents structure: Heading objects hold documents structure through their heading level property. The types of document of interest to SiSU have structure that is captured by the heading level property. Headings are individual objects like any other with the additional properties that (i) they may be regarded as containing the other objects following them sequentially (until the next heading of a similar or higher level), heading objects may include other headings (sub-headings), and (ii) that they have a heirarchy, the root "heading" being the document title. A complication was intruduced to provide greater flexibility across document output formats. Headings have two sets of levels, the level under which substantive text occurs, this would be a chapter or segment level, and above that in the heirarchy if needed are document section separators, book, section, part. 12

Non-objects Most but not all parts of a document are treated as objects. Notably footnotes are not objects in themselves, rather belonging to the object from which they are referenced, and following their own numbering sequence. From heading objects (linked) tables of content may be generated, and if additional metadata is provided book type indexes can be generated that link back to the objects to which they relate. 13

The Document Header. SiSU document have headers which contain document metadata, at a minimum the document title and author. In addition the document header may contain markup instruction (e.g. how to identify headings within the document, in which case those headings need not be found and treated accordingly) 14

SiSU parsers have now been implemented in different programming paradigms and languages a couple of times, the chosen markup has been left unchanged though the document headers have been modified. 15

This is the core of sisu, beyond which there is more but largely in the form of choices based on ... existing output formats and of implementation detail, deciding what attributes of objects, or within objects should be supported, extending markup to allow for the generation of book indexes from if tagging provided. 16

1.1. Older Descriptions 17

Here is a description that has been used for the original sisu (scribe): 18

With minimal preparation of a plain-text (UTF-8) file, using sisu markup syntax in your text editor of choice, SiSU can generate various document formats, most of which share a common object numbering system for locating content, including plain text, HTML, XHTML, XML, EPUB, OpenDocument text (ODF:ODT), LaTeX, PDF files, and populate an SQL database with objects (roughly paragraph-sized chunks) so searches may be performed and matches returned with that degree of granularity. Think of being able to finely match text in documents, using common object numbers, across different output formats (same object identifier for pdf, epub or html) and across languages if you have translations of the same document (same object identifier across languages). For search, your criteria is met by these documents at these locations within each document (equally relevant across different output formats and languages). To be clear (if obvious) page numbers provide none of this functionality. Object numbering is particularly suitable for ”published” works (finalized texts as opposed to works that are frequently changed or updated) for which it provides a fixed means of reference of content. Document outputs can also share provided semantic meta-data. 19

1.2. ... 20

SiSU is less about document layout than it is about finding a way using little markup to construct an abstract representation of a document that makes it possible to produce multiple representations of it which may be rather different from each other and used for different purposes, whether layout and publishing, scrollworthy online viewing/ reading, or content search. To be able to take advantage from its minimal preparation starting point of some of the strengths of rather different established ways of representing documents for different purposes, whether for search (relational database, or indexed flat files generated for that purpose whether of complete documents, or say of files made up of objects), online or other electronic viewing (e.g. html, xml, epub), or paper publication (e.g. pdf via latex)... 21

The solution arrived at is to extract structural information about the document (document sections and headings within the document, available through pattern matching or markup) and tracking objects (which primarily are defined units of text such as paragraphs, headings, tables, verse, etc. but also images) which can be reconstituted as the same documents with relevant object identification numbers so text (objects) can be referenced across different output formats and presentations. 22

SiSU generates tables of content, and through its markup the means for metadata to be provided for the generation of book style indexes for a document (that again due to document object numbers are the same and equally relevant across all document formats). Per document classifying/organizing metadata can also be provided for automated document curation. 23

... there have also been working experiments with sisu markup source, two way conversion/representation of sisu document markup source in mind-mapping (software kdissert was used for its strong focus on producing documents (now apparently called semantik)); also po4a software for translators has been used successfuly in 24 its regular text mode for sisu markup in translation, (which is more an attribute of po4a than of sisu, but) which is of interest due to sisu/spine's object citation numbering being available across translations. Open Document Format text (odf:odt), has been an output, but much more interesting (and requested by potential users of sisu/spine) would be the ability of a word processor to save text/a document in sisu markup, making alternative document processing and presentations with sisu possible.

also worth mention, in the relatively long history of this project, there has been work done on extracting hash representations of each object, that could hypothetically be shared to prove the content of a document without sharing its content, or of identifying which objects change; these hashes can also be used as unique identifiers in a database or as identifying filenames if individual objects are saved. 25

SiSU has evolved, the current implementation focuses on one primary use-case, books and literary writings. However the concept on which it is based has wider application. Here is a prevously posted souvenir from my encounter with an IBM software evaluator in London June 2004 that came about through a chance encounter with an IBM manager at a Linux Expo, who was curious about my interest in Gnu/Linux with my legal background... on hearing that I also wrote software, he suggested, maybe IBM should have a look at it. I was interested, the meeting was set up... with an IBM, Software Innovations evaluator<br>His response after the meeting: 26

"Ralph Good to meet with you today, I was very impressed with your software. 27

[colleague's name (also posted to an IBM colleague)] - in summary - Ralph has built an application that runs on linux and takes ASCII documents and pulls them apart in to the smallest constituent parts, storing them as XML, PDF and HTML, the HTML are hyperlinked up so the document can be browsed in its full form. the format and text data created is stored in a database.<br>This has potential in any place that needs the power of full text search whilst holding the structural concepts of the document i.e. legal, pharma, education, research.. which ones we need to figure out, ..."

Special interest was expressed in the search implications of SiSU. To paraphrase, the company has document management systems dealing with hundreds of thousands of texts, these tell you which documents match your search criteria, but cannot inform you where within a text these matches were found without opening the documents. This is achieved through defining document objects and making them the building block of the document, trackable document objects (that can be placed back in the context of the document or corpus of documents if part of a collection). SiSU's early design was to - abstract documents to their structure, and identified objects, numbered in a citable way (as pointed out document object hashes can be of use for the purpose). 28

1.3. SiSU Spine 29

SiSU Spine is the new generator for documents prepared in sisu markup, written in D as opposed to the original sisu which was first shared in Ruby. 30

Spine code has not as yet been made publicly available. 31

As compared with the original sisu generator sisu spine: 32

- Spine uses the same document markup for the document body, but uses yaml for document headers (which contains document metadata and configuration details), the original sisu has a bespoke markup for headers. 33

- Spine (written in D) is considerably faster at generating native output than sisu (written in Ruby), on last test at least 60 times faster (what took 1 minute takes 1 second; 1 hour a minute :-) (admittedly some time ago, ruby has been getting faster, hopefully this is not over over promising). 34

- Spine produces fewer document outputs types than sisu (html, epub, (odt, latex) and populates sql db for search) 35

- As regards non-native output, so far Spine has greater separation of what it does and largely leaves calling the external program to the user, e.g.: latex output is a native output in the sense that it is generated directly by spine, but the pdfs that can be produced from these are produced through use of an external program xelatex, which produces fine output but is a very much slower process. 36

- (where both produce the same output type, generally) Spine generally produces more up to date output format representations. 37

SiSU Markup 38

2. Introduction to SiSU Markup[1] 39

2.1. Summary 40

This is the D version of the program sisu on which the markup it uses is based. 41

SiSU source documents are plaintext (UTF-8)[2] files 42

All paragraphs are separated by an empty line. 43

Markup is comprised of: markup-summary * 44

at the top of a document, the document header made up of semantic meta-data about the document and if desired additional processing instructions (such an instruction to automatically number headings from a particular level down) 45

followed by the prepared substantive text of which the most important single characteristic is the markup of different heading levels, which define the primary outline of the document structure. Markup of substantive text includes: 46

heading levels defines document structure 47

text basic attributes, italics, bold etc. 48

grouped text (objects), which are to be treated differently, such as code blocks or poems. 49

footnotes/endnotes 50

linked text and images 51

paragraph actions, such as indent, bulleted, numbered-lists, etc. 52

2.2. Markup Rules, document structure and metadata requirements 53

minimal content/structure requirement, minimum being: 54

metadata 55

56

```
title: ”SiSU Spine”
  subtitle: ”Markup”

creator:
  author: ”Amissah, Ralph”
```

levels 57

58

```
A~ (level A [title])

1~ (at least one level 1 [segment/(chapter)])
```

structure rules (document heirarchy, heading levels): 59

there are two sets of heading levels ABCD (title & parts if any) and 123 (segment & 60

[1] From sometime after SiSU 0.58 it should be possible to describe SiSU markup using SiSU, which

subsegments if any)

sisu has the fllowing levels (that may be described as document parts, headings and subheadings): 61

62

```
A~ [title (& author)]
   - document root, required once (== 1)
   - followed by part B~ or level 1~
   - often written in the form:
     A~ @title @creator
     where title and creator are taken from the document header

B~ [part]
   - part B is followed by a part C~ if there is one or level 1~

C~ [subpart]
   - part C is followed by a part D~ if there is one or level 1~

D~ [subsubpart]
   - part D is followed by level 1~

1~ [heading, segment (chapter)]
   - level 1 required at least once (>= 1)
   - is followed by level 2~ or
     by text which can then be followed
     - by more text or by levels 1~ or 2~ (or relevant part)
   - level 1 in html (and epub) is the basis of a document segment and in a book
     would correspond to a chapter

2~ [sub-heading]
   - followed by level 3~ or
   - by text which can then be followed
     by more text or by levels 1~, 2~ or 3~ (or relevant part)

3~ [sub-sub-heading]
   - followed by text which can be followed
     by more text or by levels 1~, 2~ or 3~ (or relevant part)
```

Rules: 63

64

```
- level A~ is mandatory, it is the (document root and) title

- there can only be one document root == level/part A~

- heading levels B,C,D, are optional and there may be several of each
  (where all three are used corresponding to e.g. Book, Part, Section)
  - sublevels that are used must follow each other sequentially
    (alphabetically),

- heading levels A~ B~ C~ D~ are followed by other heading levels rather
  than substantive text
  - which may be the subsequent sequential (alphabetic) heading part level
  - or a heading (segment) level 1~

- there must be at least one heading (segment) level 1~
  (the level on which the text is segmented, in a book would correspond
  to the Chapter level)

- additional heading levels 1~ 2~ 3~ are optional and there may be several
  of each

- heading levels 1~ 2~ 3~ are followed by text (which may be followed by
```

though not an original design goal is useful.

[2] files should be prepared using UTF-8 character encoding

```
the same heading level)
and/or the next lower numeric heading level (followed by text)
or indeed return to the relevant part level
(as a corollary to the rules above substantive text/ content
must be preceded by a level 1~ (2~ or 3~) heading)
```

2.3. Markup Examples 65

2.3.1. Online 66

Markup examples are available in the form of prepared texts that were written under creative commons license that permit re-publication. 67

There is of course this document, which is provided with the program and provides a cursory overview of sisu markup. Running sisu spine against it gives an overview of the output produced by the program. 68

3. Markup of Headers 69

The document header is based on yaml, and is the part of the document preceeding the document root marked by ”A [Document title & author]” 70

The document header contains either: semantic meta-data about the document, or processing instructions. 71

Note: the first line of a document may include information on the markup version used in the form of a comment. Comments within the header section are the hash symbol at the start of a line (and as the first character in a line of text) followed by a space and the comment: 72

73

```
# in the header section of a document, this would be a comment
```

3.1. Sample Header 74

This current document is loaded by a master document that has a header similar to this one: 75

76

```
# SiSU 8.0

title:
  main: ”SiSU”
  subtitle: ”Markup”

creator:
  author: ”Amissah, Ralph”

date:
  created: ”2002-08-28”
  issued: ”2002-08-28”
  available: ”2002-08-28”
  published: ”2008-05-22”
  modified: ”2020-04-11”

rights:
  copyright: ”Copyright (C) Ralph Amissah 2007, 2020”
  license: ”AGPL 3 (part of SiSU Spine documentation)”

classify:
  topic_register: ”electronic documents:SiSU:document:markup;SiSU:document:markup;SiSU:↩
      manual:markup;electronic documents:SiSU:manual:markup”
  subject = ”ebook, epublishing, electronic book, electronic publishing, electronic document↩
      , electronic citation, data structure, citation systems, search”
```

Looking back a bit: 77

78

```
# SiSU master 8.0

title:
  main: ”SiSU”
  subtitle: ”Markup”

creator:
  author: ”Amissah, Ralph”
```

```
date:
  created: ”2002-08-28”
  issued: ”2002-08-28”
  available: ”2002-08-28”
  published: ”2008-05-22”
  modified: ”2020-04-11”

rights:
  copyright: ”Copyright (C) Ralph Amissah 2007, 2020”
  license: ”AGPL 3 (part of SiSU Spine documentation)”

classify:
  topic_register: ”electronic documents:SiSU:document:markup;SiSU:document:markup;SiSU:↩
      manual:markup;electronic documents:SiSU:manual:markup”
  subject: ”ebook, epublishing, electronic book, electronic publishing, electronic document,↩
       electronic citation, data structure, citation systems, search”

make:
  auto_num_top_at_level: ”1”
  substitute: [
    [ ”[$]{2}\\{sisudoc\\}”, ”www.sisudoc.org” ]
  ]
  bold: ”Debian|SiSU”
  italics: ”Linux|GPL|LaTeX|SQL”
  breaks: ”new=:B; break=1”
  home_button_text: ”{SiSU}https://sisudoc.org; {sources / git}https://git.sisudoc.org/↩
      projects/”
  footer: ”{SiSU}https://sisudoc.org; {git}https://git.sisudoc.org/projects”
```

3.2. Available Headers 79

Header tags appear at the beginning of a document and provide meta information on the document (such as the Dublin Core), or information as to how the document as a whole is to be processed. All header instructions take the form headername: or on the next line and indented by two spaces subheadername: All Dublin Core meta tags are available 80

@identifier: information or instructions 81

where the ”identifier” is a tag recognised by the program, and the ”information” or ”instructions” belong to the tag/identifier specified 82

Note: a header where used should only be used once; all headers apart from [title] are optional; the [structure] header is used to describe document structure, and can be useful to know. 83

This is a sample header 84

85

```
# SiSU 8.0
```

86

```
title:
  main: ”SiSU”
  subtitle: ”Markup”
  language: ”English”
```

87

```
creator:
  author: [Lastname, First names]
  illustrator: [Lastname, First names]
  translator: [Lastname, First names]
  prepared_by: [Lastname, First names]
```

88

```
date:
  created: [year or yyyy-mm-dd]
  issued: [year or yyyy-mm-dd]
  available: [year or yyyy-mm-dd]
  published: [year or yyyy-mm-dd]
  modified: [year or yyyy-mm-dd]
  valid: [year or yyyy-mm-dd]
  added_to_site: [year or yyyy-mm-dd]
  translated: [year or yyyy-mm-dd]
```

89

```
rights:
  copyright: "Copyright (C) [Year and Holder]"
  license: "[Use License granted]"
  text: "[Name, Year]"
  translation: "[Name, Year]"
  illustrations: "[Name, Year]"

# check rest
```

90

```
classify:
  topic_register: "electronic documents;SiSU:document:markup;SiSU:document:markup;SiSU:↩
      document:markup;SiSU:manual:markup;electronic documents:SiSU:manual:markup"
  subject: "ebook, epublishing, electronic book, electronic publishing, electronic document,↩
       electronic citation, data structure, citation systems, search"
  keywords: "list"
  loc: "[Library of Congress classification]"
  dewey: "[Dewey classification]"
```

91

```
identifier:
  isbn: "[ISBN]"
  oclc: ""
```

92

```
links: [
  "{SiSU }https://www.sisudoc.org",
  "{ FSF }https://www.fsf.org",
]
```

93

```
make:
  auto_num_top_at_level: "1"
  substitute: [
    [ "[$]{2}\\{sisudoc\\}", "www.sisudoc.org" ]
  ]
  bold: "Debian|SiSU" # [regular expression of words/phrases to be made bold]
  italics: "Linux|GPL|LaTeX|SQL" # [regular expression of words/phrases to italicise]
  breaks: "new=:B; break=1"
  home_button_text: "{SiSU}https://sisudoc.org; {sources / git}https://git.sisudoc.org/↩
      gitweb/"
  footer: "{SiSU}https://sisudoc.org; {git}https://git.sisudoc.org"
  headings: text to match for each level
    (e.g. PART; Chapter; Section; Article; or another: none; BOOK|FIRST|SECOND; none; ↩
        CHAPTER;)
```

4. Markup of Substantive Text 94

4.1. Heading Levels 95

96 Heading levels are :A ,:B ,:C ,1 ,2 ,3 ... :A - :C being part / section headings, followed by other heading levels, and 1 -6 being headings followed by substantive text or sub-headings. :A usually the title :A ? conditional level 1 heading (used where a stand-alone document may be imported into another)

97 **:A [heading text]** Top level heading [this usually has similar content to the title [title]] NOTE: the heading levels described here are in 0.38 notation, see heading

98 **:B [heading text]** Second level heading [this is a heading level divider]

99 **:C [heading text]** Third level heading [this is a heading level divider]

100 **1 [heading text]** Top level heading preceding substantive text of document or sub-heading 2, the heading level that would normally be marked 1. or 2. or 3. etc. in a document, and the level on which sisu by default would break html output into named segments, names are provided automatically if none are given (a number), otherwise takes the form 1 my_filename_for_this_segment

101 **2 [heading text]** Second level heading preceding substantive text of document or sub-heading 3 , the heading level that would normally be marked 1.1 or 1.2 or 1.3 or 2.1 etc. in a document.

102 **3 [heading text]** Third level heading preceding substantive text of document, that would normally be marked 1.1.1 or 1.1.2 or 1.2.1 or 2.1.1 etc. in a document

103

```
1~filename level 1 heading,

% the primary division such as Chapter that is followed by substantive text, and may be ←↩
    further subdivided (this is the level on which by default html segments are made)
```

4.2. Font Attributes 104

markup example: 105

106

```
normal text, *{emphasis}*, !{bold text}!, /{italics}/, _{underscore}_, ”{citation}”,
^{superscript}^, ,{subscript},, +{inserted text}+, -{strikethrough}-, #{monospace}#

normal text

*{emphasis}* [note: can be configured to be represented by bold, italics or underscore]

!{bold text}!

/{italics}/

_{underscore}_

”{citation}”

^{superscript}^
```

```
,{subscript},

+{inserted text}+

-{strikethrough}-

#{monospace}#
```

resulting output: 107

normal text, **emphasis**, **bold text**, *italics*, _underscore_, ”{citation}”, ^{superscript}, $_{subscript}$, inserted text, ~~strikethrough~~, `monospace` 108

normal text 109

emphasis [note: can be configured to be represented by bold, italics or underscore] 110

bold text 111

italics 112

underscore 113

”{citation}” 114

^{superscript} 115

$_{subscript}$ 116

inserted text 117

~~strikethrough~~ 118

`monospace` 119

4.3. Indentation and bullets 120

markup example: 121

122

```
ordinary paragraph

_1 indent paragraph one step

_2 indent paragraph two steps

_9 indent paragraph nine steps
```

resulting output: 123

ordinary paragraph 124

indent paragraph one step 125

indent paragraph two steps 126

indent paragraph nine steps 127

markup example: 128

129

```
_* bullet text

_1* bullet text, first indent

_2* bullet text, two step indent
```

resulting output: 130

bullet text 131

bullet text, first indent 132

bullet text, two step indent 133

Numbered List (not to be confused with headings/titles, (document structure)) 134

markup example: 135

136

```
# numbered list                numbered list 1., 2., 3, etc.

_# numbered list numbered list indented a., b., c., d., etc.
```

4.4. Hanging Indents 137

markup example: 138

139

```
_0_1 first line no indent (no hang),
rest of paragraph indented one step

_1_0 first line indented,
rest of paragraph no indent

in each case level may be 0-9
```

resulting output: 140

first line no indent, rest of paragraph indented one step; first line no indent, rest of paragraph indented one step; first line no indent, rest of paragraph indented one step; first line no indent, rest of paragraph indented one step; first line no indent, rest of paragraph indented one step; first line no indent, rest of paragraph indented one step; first line no indent, rest of paragraph indented one step; first line no indent, rest of paragraph indented one step; first line no indent, rest of paragraph indented one step; 141

A regular paragraph. 142

first line indented, rest of paragraph no indent first line indented, rest of paragraph no indent first line indented, rest of paragraph no indent first line indented, rest of paragraph no indent first line indented, rest of paragraph no indent first line indented, rest of paragraph no indent first line indented, rest of paragraph no indent first line indented, rest of paragraph no indent first line indented, rest of paragraph no indent first line indented, rest of paragraph no indent 143 in each case level may be 0-9 144

live-build A collection of scripts used to build customized Debian Livesystems. *live-build* was formerly known as live-helper, and even earlier known as live-package. 145

live-build 146
A collection of scripts used to build customized Debian Livesystems. *live-build* was formerly known as live-helper, and even earlier known as live-package.

4.5. Footnotes / Endnotes 147

Footnotes and endnotes are marked up at the location where they would be indicated within a text. They are automatically numbered. The output type determines whether footnotes or endnotes will be produced 148

markup example: 149

150

```
~{ a footnote or endnote }~
```

resulting output: 151

[3] 152

markup example: 153

154

```
normal text~{ self contained endnote marker & endnote in one }~ continues
```

resulting output: 155

normal text[4] continues 156

markup example: 157

158

```
normal text ~{* unnumbered asterisk footnote/endnote, insert multiple asterisks if required ↩
    }~ continues

normal text ~{** another unnumbered asterisk footnote/endnote }~ continues
```

resulting output: 159

normal text * unnumbered asterisk footnote/endnote, insert multiple asterisks if required continues 160

normal text ** another unnumbered asterisk footnote/endnote continues 161

markup example: 162

163

[3] a footnote or endnote

[4] self contained endnote marker & endnote in one

```
normal text ~[* editors notes, numbered asterisk footnote/endnote series ]~ continues

normal text ~[+ editors notes, numbered plus symbol footnote/endnote series ]~ continues
```

resulting output: 164

normal text [* editors notes, numbered asterisk footnote/endnote series] continues 165

normal text [+ editors notes, numbered plus symbol footnote/endnote series] continues 166

[discontinued] Alternative binary endnote notation (endnote pair) for footnotes/endnotes: 167

168

```
% note the endnote marker "~^"

normal text~^ continues

^~ endnote text following the paragraph in which the marker occurs
```

standard (inline) and pair (binary) notation could not be mixed in the same document. 169

The reason binary notation was provided as an option was for the conversion of documents to sisu markup. Many documents were prepared in such a way that endnotes had been previously marked up in a binary fashion, and this provided a convenient and faster way to make the document conversion, just reflect those markup practices. The reason it has been dropped is it adds a slowing step to something that needs to be done at most once and it prove to be flakey, unnecessarily so even when kept under version control. It is preferable to do a two step conversion of the previously marked up document to sisu: first to the binary/paired footnote markup, then; convert it to the proper form of inline endnote markup with a dedicated helper conversion program, keeping the resulting properly marked up text. 170

4.6. Links 171

4.6.1. Naked URLs within text, dealing with urls 172

urls found within text are marked up automatically. A url within text is automatically hyperlinked to itself and by default decorated with angled braces, unless they are contained within a code block (in which case they are passed as normal text), or escaped by a preceding underscore (in which case the decoration is omitted). 173

markup example: 174

175

```
normal text https://www.sisudoc.org/ continues
```

resulting output: 176

normal text 「https://www.sisudoc.org/」 continues 177

An escaped url without decoration 178

markup example: 179

180

```
normal text _https://www.sisudoc.org/ continues

deb _https://www.jus.uio.no/sisu/archive unstable main non-free
```

resulting output: 181

normal text _https://www.sisudoc.org/ continues 182

deb _https://www.jus.uio.no/sisu/archive unstable main non-free 183

where a code block is used there is neither decoration nor hyperlinking, code blocks are discussed later in this document 184

resulting output: 185

186

```
deb https://www.jus.uio.no/sisu/archive unstable main non-free
deb-src https://www.jus.uio.no/sisu/archive unstable main non-free
```

4.6.2. Linking Text 187

To link text or an image to a url the markup is as follows 188

markup example: 189

190

```
about { SiSU }https://url.org markup
```

resulting output: 191

about
「SiSU」 markup 192

a couple of test urls 193

「https://example.com/Alice&Bob」 194

195

「programs I use」

A shortcut notation is available so the url link may also be provided automatically as a footnote 196

markup example: 197

198

```
about {~^ SiSU }https://url.org markup
```

resulting output: 199

about 200

「SiSU」[5] markup

Internal document links to a named (anchor) tagged location, including named headings named inline anchor tags an-inline-anchor-tag or an ocn the heading: 201

202

```
1~markup Markup
```

can be linked to as follows: 203

204

```
to find out more see { Markup }#markup
```

to find out more see Markup 205

an inline anchor tag is made with the following markup internal-links 206

207

```
named inline anchor tags *~an-inline-anchor-tag
```

and linked to the same way 208

209

```
the link { an inline anchor tag }#an-inline-anchor-tag
```

the link an inline anchor tag or to another part of the document: markup summary 210

markup example: 211

212

```
about { text links }#link_text
```

resulting output: 213

about text links 214

Shared document collection link 215

markup example: 216

217

```
about { SiSU book markup examples }:SiSU/examples.html
```

resulting output: 218

about { SiSU book markup examples }:SiSU/examples.html 219

4.6.3. Linking Images 220

markup example: 221

222

[5] 「https://www.sisudoc.org/」

```
{ sm_tux.png 64x80 }image

% various url linked images

{sm_tux.png 64x80 "a better way" }https://www.sisudoc.org/

{sm_GnuDebianLinuxRubyBetterWay.png 100x101 "Way Better - with Gnu/Linux, Debian and Ruby" }↩
    https://www.sisudoc.org/

{~^ sm_ruby_logo.png "Ruby" }https://www.ruby-lang.org/en/
```

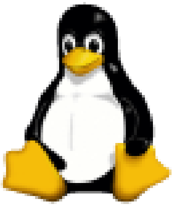
resulting output: 223

sm_tux.png,w64h80 224

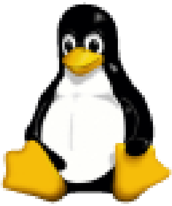
sm_tux.png,w64h80 225

sm_tux.png,w64h80 "test" 226

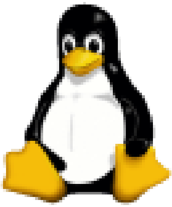
227



228

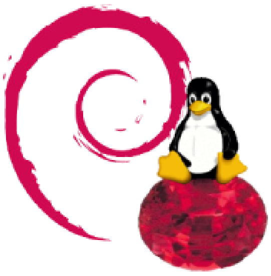


229



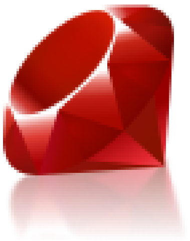
"Gnu/Linux - a better way"

230



"Way Better - with Gnu/Linux, Debian and Ruby"

231



”Ruby”[6]

232

”D for me”

233

By and large I'm trying to minimize mentions of D in C++ contexts because it's as unfair as bringing a machine gun to a knife fight.
So fast without
illing yourself
Native Conf 2012
#nc2012
Efficiency is a
ten letter wo
Later in D-Town
Let's actually do this.
Sorry, you can't get in without a knife.
KNIFE FIGHT CLUB

”D, hey no fair”[7]

linked url footnote shortcut 234

235

```
{~^ [text to link] }https://url.org

% maps to: { [text to link] }https://url.org ~{ https://url.org }~

% which produces hyper-linked text within a document/paragraph, with an endnote providing ↩
    the url for the text location used in the hyperlink
```

236

```
text marker *~name
```

note at a heading level the same is automatically achieved by providing names to headings 1, 2 and 3 i.e. 2 [name] and 3 [name] or in the case of auto-heading numbering, without further intervention. 237

[6] 「https://www.ruby-lang.org/en/」

[7] 「https://github.com/dlang-community/d-mans」

4.6.4. Link shortcut for multiple versions of a sisu document in the same directory tree 238

markup example: 239

240

```
!_ /{”Viral Spiral”}/, David Bollier

{ ”Viral Spiral”, David Bollier [3sS]}viral_spiral.david_bollier.sst
```

***”Viral Spiral”*, David Bollier** 241

{ ”Viral Spiral”, David Bollier [3sS]}viral_spiral.david_bollier.sst 242

4.7. Grouped Text / blocked text 243

There are two markup syntaxes for blocked text, using curly braces or using tics 244

4.7.1. blocked text curly brace syntax 245

at the start of a line on its own use name of block type with an opening curly brace, follow with the content of the block, and close with a closing curly brace and the name of the block type, e.g. 246

247

```
code{

this is a code block

}code
```

248

```
poem{

this here is a poem

}poem
```

4.7.2. blocked text tic syntax 249

250

~~~
``` code
this is a code block
```

``` poem
this here is a poem
```
~~~

start a line with three backtics, a space followed by the name of the name of block type, follow with the content of the block, and close with three back ticks on a line of their own, e.g. 251

4.7.3. Group 252

The ”group” is different from the ”block” mark in that ”group” does not preserve whitespace, the ”block” mark does. The text falling within the block is a single object. 253

basic markup: 254

255

```
group{

  Your grouped text here

}group

A group is treated as an object and given a single object number.
```

resulting group text output: 256

‘Fury said to a mouse, That he met in the house, "Let us both go to law: I will prosecute YOU. –Come, I’ll take no denial; We must have a trial: For really this morning I’ve nothing to do." Said the mouse to the cur, "Such a trial, dear Sir, With no jury or judge, would be wasting our breath."[8] "I’ll be judge, I’ll be jury," Said cunning old Fury: "I’ll try the whole[9] cause, and condemn you to death."’ 257

resulting group text output: 258

The Road Not Taken Related Poem Content Details BY ROBERT FROST Two roads diverged in a yellow wood, And sorry I could not travel both And be one traveler, long I stood And looked down one as far as I could To where it bent in the undergrowth; Then took the other, as just as fair, And having perhaps the better claim, Because it was grassy and wanted wear; Though as for that the passing there Had worn them really about the same, And both that morning equally lay In leaves no step had trodden black. Oh, I kept the first for another day! Yet knowing how way leads on to way, I doubted if I should ever come back. I shall be telling this with a sigh Somewhere ages and ages hence: Two roads diverged in a wood, and I I took the one less traveled by, And that has made all the difference. 259

4.7.4. Block 260

The ”block” is different from the ”group” mark in that the ”block” mark (like the ”poem” mark) preserves whitespace, the ”group” mark does not. The text falling within the ”block” is a single object, which is different from the ”poem” mark where each identified verse is an object. 261

basic markup: 262

263

```
block{

  Your block text here

}block

A block is treated as an object and given a single object number.
```

resulting block text output: 264

[8] endnote test

[9] stress test

‘Fury said to a 265
mouse, That he
met in the
house,
"Let us
both go to
law: I will
prosecute
YOU. –Come,
I’ll take no
denial; We
must have a
trial: For
really this
morning I’ve
nothing
to do."
Said the
mouse to the
cur, "Such
a trial,
dear Sir,
With
no jury
or judge,
would be
wasting
our
breath."[10]
"I’ll be
judge, I’ll
be jury,"
Said
cunning
old Fury:
"I’ll
try the
whole[11]
cause,
and
condemn
you
to
death."’

curly brace delimiter, resulting block text output: 266

The Road Not Taken Related Poem Content Details 267

[10] endnote test

[11] stress test

BY ROBERT FROST

Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;

Then took the other, as just as fair,
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that the passing there
Had worn them really about the same,

And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back.

I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I
I took the one less traveled by,
And that has made all the difference.

4.7.5. Poem 268

The ”poem” mark like the ”block” preserves whitespace. Text followed by two newlines are identified as verse and each verse is an object i.e. a poem may consist of multiple verse each of which is identified as an object, unlike a text ”block” which is identified as a single object. 269

basic markup: 270

271

```
poem{

  Your poem here

}poem

Each verse in a poem is given an object number.
```

curly brace delimiter, resulting poem text output (broken into verse): 272

273

‘Fury said to a
mouse, That he
met in the

house,
"Let us
both go to
law: I will
prosecute
YOU. –Come,
I'll take no
denial; We
must have a
trial: For
really this
morning I've
nothing
to do."
Said the
mouse to the
cur, "Such
a trial,
dear Sir,
With
no jury
or judge,
would be
wasting
our
breath."
"I'll be
judge, I'll
be jury,"
Said
cunning
old Fury:
"I'll
try the
whole
cause,
and
condemn
you
to
death."'

curly brace delimiter, resulting poem text output (broken into verse): 274

275

The Road Not Taken[12]

276

by Robert Frost

277

Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;

278

[12] published in 1916 as the first poem in the collection Mountain Interval.

Then took the other, as just as fair,
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that the passing there
Had worn them really about the same,

279
And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back.

280
I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I
I took the one less traveled by,
And that has made all the difference.

tics delimiter, resulting group text output: 281

282
The Road Not Taken[13]

283
by Robert Frost

284
Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;

285
Then took the other, as just as fair,
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that the passing there
Had worn them really about the same,

286
And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back.

287
I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I
I took the one less traveled by,
And that has made all the difference.

4.7.6. Code 288

”Code” blocks are a single text object, in which the original text is preserved. 289

Code tags `code{ ... }code` (used as with other group tags described above) are 290

[13] published in 1916 as the first poem in the collection Mountain Interval.

used to escape regular sisu markup, and have been used extensively within this document to provide examples of SiSU markup. You cannot however use code tags to escape code tags. They are however used in the same way as group or poem tags.

A code-block is treated as an object and given a single object number. [an option to number each line of code may be considered at some later time] 291

use of code tags instead of poem compared, resulting output: 292

293

```
           `Fury said to a
          mouse, That he
        met in the
      house,
   "Let us
     both go to
       law:  I will
         prosecute
           YOU.  --Come,
              I'll take no
               denial; We
            must have a
        trial:  For
     really this
  morning I've
 nothing
to do."
  Said the
    mouse to the
      cur, "Such
        a trial,
          dear Sir,
                With
            no jury
         or judge,
       would be
     wasting
    our
     breath."
      "I'll be
        judge, I'll
          be jury,"
                Said
           cunning
             old Fury:
            "I'll
             try the
                whole
                 cause,
                    and
               condemn
              you
             to
              death."'
```

From SiSU 2.7.7 on you can number codeblocks by placing a hash after the opening code tag `code{#` as demonstrated here: 294

295

```
           `Fury said to a
          mouse, That he
        met in the
      house,
   "Let us
     both go to
```

```
                law:  I will
                  prosecute
                    YOU.  --Come,
                       I'll take no
                        denial; We
                     must have a
                 trial:  For
              really this
           morning I've
          nothing
         to do."
           Said the
             mouse to the
               cur, "Such
                 a trial,
                   dear Sir,
                         With
                     no jury
                  or judge,
                would be
              wasting
             our
              breath."
               "I'll be
                 judge, I'll
                   be jury,"
                         Said
                    cunning
                      old Fury:
                     "I'll
                      try the
                         whole
                          cause,
                             and
                        condemn
                       you
                      to
                       death."'
```

4.7.7. Tables 296

Tables may be prepared in two either of two forms 297

markup example: 298

299

```
table(c3: 40, 30, 30){

This is a table
this would become column two of row one
column three of row one is here

And here begins another row
column two of row two
column three of row two, and so on

}table
```

resulting output: 300

301

This is a table	this would become column two of row one	column three of row one is here
And here begins another row	column two of row two	column three of row two, and so on

Same as a tic table 302

303

This is a table	this would become column two of row one	column three of row one is here
And here begins another row	column two of row two	column three of row two, and so on

Without instruction 304

305

This is a table	this would become column two of row one	column three of row one is here
And here begins another row	column two of row two	column three of row two, and so on

a second form may be easier to work with in cases where there is not much information in each column 306

markup example:[14] 307

308

```
!_ Table 3.1: Contributors to Wikipedia, January 2001 - June 2005

{table(h; 24, 12, 12, 12, 12, 12, 12)}
                                |Jan. 2001|Jan. 2002|Jan. 2003|Jan. 2004|July 2004|June 2006
Contributors*                   |       10|      472|    2,188|    9,653|   25,011|   48,721
Active contributors**           |        9|      212|      846|    3,228|    8,442|   16,945
Very active contributors***     |        0|       31|      190|      692|    1,639|    3,016
No. of English language articles|       25|   16,000|  101,000|  190,000|  320,000|  630,000
No. of articles, all languages  |       25|   19,000|  138,000|  490,000|  862,000|1,600,000

* Contributed at least ten times; ** at least 5 times in last month; *** more than 100 times↩
     in last month.
```

resulting output: 309

Table 3.1: Contributors to Wikipedia, January 2001 - June 2005 310

311

	Jan. 2001	Jan. 2002	Jan. 2003	Jan. 2004	July 2004	June 2006
Contributors*	10	472	2,188	9,653	25,011	48,721
Active contributors**	9	212	846	3,228	8,442	16,945
Very active contributors***	0	31	190	692	1,639	3,016
No. of English language articles	25	16,000	101,000	190,000	320,000	630,000
No. of articles, all languages	25	19,000	138,000	490,000	862,000	1,600,000

* Contributed at least ten times; ** at least 5 times in last month; *** more than 100 times in last month. 312

4.8. Additional breaks - linebreaks within objects, column and page-breaks 313

4.8.1. line-breaks 314

To break a line within a ”paragraph object”, two backslashes
with a space before and a space or newline after them
may be used. 315

316

[14] Table from the Wealth of Networks by Yochai Benkler
「https://www.jus.uio.no/sisu/the_wealth_of_networks.yochai_benkler」

```
To break a line within a "paragraph object",
two backslashes \\ with a space before
and a space or newline after them \\
may be used.
```

The html break br enclosed in angle brackets (though undocumented) is available in versions prior to 3.0.13 and 2.9.7 (it remains available for the time being, but is depreciated). 317

To draw a dividing line dividing paragraphs, see the section on page breaks. 318

4.8.2. page breaks 319

Page breaks are only relevant and honored in some output formats. A page break or a new page may be inserted manually using the following markup on a line on its own: 320

page new == breaks the page, starts a new page. 321

page break -- breaks a column, starts a new column, if using columns, else breaks the page, starts a new page. 322

page break line across page -..- draws a dividing line, dividing paragraphs 323

page break: 324

325

```
-\\-
```

page (break) new: 326

327

```
=\\=
```

page (break) line across page (dividing paragraphs): 328

329

```
-..-
```

4.9. Excluding Object Numbers 330

Object numbers can be switched off by adding a # to the end of a text object. 331

Sometimes it is wished to switch off object numbers for a larger group of text. In this case it is possible before the group, body of text to be without object numbers on a new line with nothing else on it to open the un-numbered object block with – # and to close the un-numbered block, and restart object numbering with on a similarly otherwise empty new-line with –+# 332

333

```
--~#

un-numbered object block of text contained here

still un-numbered

--+#

object numbering returns here and for subsequent text objects

to switch of object numbering for a single objct, to the end of the object add ~# like so:~#
```

4.10. Bibliography / References 334

There are three ways to prepare a bibliography using sisu (which are mutually exclusive): (i) manually preparing and marking up as regular text in sisu a list of references, this is treated as a regular document segment (and placed before endnotes if any); (ii) preparing a bibliography, marking a heading level `1 !biblio` (note the exclamation mark) and preparing a bibliography using various metadata tags including for author: title: year: a list of which is provided below, or; (iii) as an assistance in preparing a bibliography, marking a heading level `1 !biblio` and tagging citations within footnotes for inclusion, identifying citations and having a parser attempt to extract them and build a bibliography of the citations provided. 335

For the heading/section sequence: endnotes, bibliography then book index to occur, the name biblio or bibliography must be given to the bibliography section, like so: 336

337

```
1~!biblio
```

4.10.1. a markup tagged metadata bibliography section 338

Here instead of writing your full citations directly in footnotes, each time you have new material to cite, you add it to your bibliography section (if it has not been added yet) providing the information you need against an available list of tags (provided below). 339

The required tags are au: ti: and year: [15] an short quick example might be as follows: 340

341

```
1~!biblio

au: von Hippel, E.
ti: Perspective: User Toolkits for Innovation
lng: (language)
jo: Journal of Product Innovation Management
vo: 18
ed: (editor)
yr: 2001
note:
```

[15] for which you may alternatively use the full form author: title: and year:

```
sn: Hippel, /{User Toolkits}/ (2001)
id: vHippel_2001
% form:

au: Benkler, Yochai
ti: The Wealth of Networks
st: How Social Production Transforms Markets and Freedom
lng: (language)
pb: Harvard University Press
edn: (edition)
yr: 2006
pl: U.S.
url: https://cyber.law.harvard.edu/wealth_of_networks/Main_Page
note:
sn: Benkler, /{Wealth of Networks}/ (2006)
id: Benkler2006

au: Quixote, Don; Panza, Sancho
ti: Taming Windmills, Keeping True
jo: Imaginary Journal
yr: 1605
url: https://en.wikipedia.org/wiki/Don_Quixote
note: made up to provide an example of author markup for an article with two authors
sn: Quixote & Panza, /{Taming Windmills}/ (1605)
id: quixote1605
```

Note that the section name !biblio (or !bibliography) is required for the bibliography to be treated specially as such, and placed after the auto-generated endnote section. 342

Using this method, work goes into preparing the bibliography, the tags author or editor, year and title are required and will be used to sort the bibliography that is placed under the Bibliography section 343

The metadata tags may include shortname (sn:) and id, if provided, which are used for substitution within text. Every time the given id is found within the text it will be replaced by the given short title of the work (it is for this reason the short title has sisu markup to italicize the title), it should work with any page numbers to be added, the short title should be one that can easily be used to look up the full description in the bibliography. 344

345

```
The following footnote~{ quixote1605, pp 1000 - 1001, also Benkler2006 p 1. }~
```

would be presented as: 346

Quixote and Panza, *Taming Windmills* (1605), pp 1000 - 1001 also, Benkler, *Wealth of Networks*, (2006) p 1 or rather[16] 347

348

```
au: author Surname, FirstNames (if multiple semi-colon separator)
    (required unless editor to be used instead)
ti: title  (required)
st: subtitle
jo: journal
vo: volume
ed: editor (required if author not provided)
tr: translator
src: source (generic field where others are not appropriate)
```

[16] Quixote and Panza, *Taming Windmills* (1605), pp 1000 - 1001 also, Benkler, *Wealth of Networks* (2006), p 1

```
in: in (like src)
pl: place/location (state, country)
pb: publisher
edn: edition
yr: year (yyyy or yyyy-mm or yyyy-mm-dd) (required)
pg: pages
url: https://url
note: note
id: create_short_identifier e.g. authorSurnameYear
    (used in substitutions: when found within text will be
    replaced by the short name provided)
sn: short name e.g. Author, /{short title}/, Year
    (used in substitutions: when an id is found within text
    the short name will be used to replace it)
```

4.10.2. Tagging citations for inclusion in the Bibliography 349

Here whenever you make a citation that you wish be included in the bibliography, you tag the citation as such using special delimiters (which are subsequently removed from the final text produced by sisu) 350

Here you would write something like the following, either in regular text or a footnote 351

352

```
See .: Quixote, Don; Panza, Sancho /{Taming Windmills, Keeping True}/ (1605) :.
```

SiSU will parse for a number of patterns within the delimiters to try make out the authors, title, date etc. and from that create a Bibliography. This is more limited than the previously described method of preparing a tagged bibliography, and using an id within text to identify the work, which also lends itself to greater consistency. 353

4.11. Glossary 354

Using the section name `1 !glossary` results in the Glossary being treated specially as such, and placed after the auto-generated endnote section (before the bibliography/list of references if there is one). 355

The Glossary is ordinary text marked up in a manner deemed suitable for that purpose. e.g. with the term in bold, possibly with a hanging indent. 356

357

```
1~!glossary

_0_1 *{GPL}* An abbreviation that stands for ”General Purpose License.” ...

_0_1 [provide your list of terms and definitions]
```

In the given example the first line is not indented subsequent lines are by one level, and the term to be defined is in bold text. 358

4.12. Book index 359

To make an index append to paragraph the book index term relates to it, using an equal sign and curly braces. 360

Currently two levels are provided, a main term and if needed a sub-term. Sub-terms are separated from the main term by a colon. 361

362

```
Paragraph containing main term and sub-term.
={Main term:sub-term}
```

The index syntax starts on a new line, but there should not be an empty line between paragraph and index markup. 363

The structure of the resulting index would be: 364

365

```
Main term, 1
  sub-term, 1
```

Several terms may relate to a paragraph, they are separated by a semicolon. If the term refers to more than one paragraph, indicate the number of paragraphs. 366

367

```
Paragraph containing main term, second term and sub-term.
={first term; second term: sub-term}
```

The structure of the resulting index would be: 368

369

```
First term, 1,
Second term, 1,
  sub-term, 1
```

If multiple sub-terms appear under one paragraph, they are separated under the main term heading from each other by a pipe symbol. 370

371

```
Paragraph containing main term, second term and sub-term.
={Main term:
    sub-term+2|second sub-term;
  Another term
 }

A paragraph that continues discussion of the first sub-term
```

The plus one in the example provided indicates the first sub-term spans one additional paragraph. The logical structure of the resulting index would be: 372

373

```
Main term, 1,
  sub-term, 1-3,
  second sub-term, 1,
Another term, 1
```

5. Composite documents markup 374

It is possible to build a document by creating a master document that requires other documents. The documents required may be complete documents that could be generated independently, or they could be markup snippets, prepared so as to be easily available to be placed within another text. If the calling document is a master document (built from other documents), it should be named with the suffix **.ssm** Within this document you would provide information on the other documents that should be included within the text. These may be other documents that would be processed in a regular way, or markup bits prepared only for inclusion within a master document **.sst** regular markup file, or **.ssi** (insert/information) A secondary file of the composite document is built prior to processing with the same prefix and the suffix **._sst** 375

basic markup for importing a document into a master document 376

377

```
<< filename1.sst

<< filename2.ssi
```

The form described above should be relied on. Within the Vim editor it results in the text thus linked becoming hyperlinked to the document it is calling in which is convenient for editing. 378

6. Substitutions 379

markup example: 380

381

```
The current Debian is ${debian_stable} the next debian will be ${debian_testing}

Configure substitution in _sisu/sisu_document_make

@make:
 :substitute: /${debian_stable}/,'*{Wheezy}*' /${debian_testing}/,'*{Jessie}*'
```

resulting output: 382

The current Debian is ${debian_stable} the next debian will be ${debian_testing} 383

Another test ${sisudoc} ok? 384

Configure substitution in _sisu/sisu_document_make 385

7. Footnote, endnote stress test 386

Globalisation is to be observed as a trend intrinsic to the world economy.[17] Rudimentary economics explains this runaway process, as being driven by competition within the business community to achieve efficient production, and to reach and extend available markets.[18] Technological advancement particularly in transport and communications has historically played a fundamental role in the furtherance of international commerce, with the Net, technology's latest spatio-temporally transforming offering, linchpin of the ”new-economy”, extending exponentially the global reach of the business community. The Net covers much of the essence of international commerce providing an instantaneous, low cost, convergent, global and borderless: information centre, marketplace and channel for communications, payments and the delivery of services and intellectual property. The sale of goods, however, involves the separate element of their physical delivery. The Net has raised a plethora of questions and has frequently offered solutions. The increased transparency of borders arising from the Net's ubiquitous nature results in an increased demand for the transparency of operation. As economic activities become increasingly global, to reduce transaction costs, there is a strong incentive for the ”law” that provides for them, to do so in a similar dimension. The appeal of transnational legal solutions lies in the potential reduction in complexity, more widely dispersed expertise, and resulting increased transaction efficiency. The Net reflexively offers possibilities for the development of transnational legal solutions, having in a similar vein transformed the possibilities for the promulgation of texts, the sharing of ideas and collaborative ventures. There are however, likely to be tensions within the legal community protecting entrenched practices against that which is new, (both in law and technology) and the business community's goal to reduce transaction costs. This here 「`https://sisudoc.org/now`」 is a test and repeat
「does this work?」 387

Within commercial law an analysis of law and economics may assist in developing a better understanding of the relationship between commercial law and the commercial sector it serves.[19] ”...[T]he importance of the interrelations between law and economics can be seen in the twin facts that legal change is often a function of economic ideas and conditions, which necessitate and/or generate demands for legal change, and that economic change is often governed by legal change.”[20] In doing so, however, it is important to be aware that there are several 388

[17] As Maria Cattaui Livanos suggests in *The global economy - an opportunity to be seized* in *Business World* the Electronic magazine of the International Chamber of Commerce (Paris, July 1997) at 「`https://www.iccwbo.org/html/globalec.htm`」
”Globalization is unstoppable. Even though it may be only in its early stages, it is already intrinsic to the world economy. We have to live with it, recognize its advantages and learn to manage it.
That imperative applies to governments, who would be unwise to attempt to stem the tide for reasons of political expediency. It also goes for companies of all sizes, who must now compete on global markets and learn to adjust their strategies accordingly, seizing the opportunities that globalization offers.”

[18] To remain successful, being in competition, the business community is compelled to take advantage of the opportunities provided by globalisation.

[19] Realists would contend that law is contextual and best understood by exploring the interrelationships between law and the other social sciences, such as sociology, psychology, political science, and economics.

[20] Part of a section cited in Mercuro and Steven G. Medema, *Economics and the Law: from Posner to*

competing schools of law and economics, with different perspectives, levels of abstraction, and analytical consequences of and for the world that they model.[21] This sentence trails test endnote. $$$

$$$

Difference?[22] 389

* !glossary 390

head 391

header document header, containing document specific (i) metadata information or (ii) make instructions 392

(document) structure relationship between headings and sub-headings, and the objects they contain. Document structure is extracted from heading levels, which are either: explicitly marked up, or; determined from a make regex provided in the document header. Use of document structure allow for the meaningful representation of documents in alternative ways and the use of ocn permits easy reference across different output formats. 393

heading document heading, each heading is marked indicating its level (in relation to other headings), and this is used as basis for determininge document structure. There are 8 levels, which are can be distinguesed as being one of three types: (i) 1 title level (marked up A or numeric 0); (ii) 3 optional document division levels, above text separating headings (marked up B - D, or numeric 1 to 3); (iii) 4 text headings (marked up 1 - 4, or numeric 4 to 7) 394

levels == heading levels document heading level, see heading and structure 395

marked up headings / mark up level 396

collapsed headings / collapsed levels 397

numeric levels 398

dummy heading a markup level 1 / dummy level 4 that does not exist in the original text that is manually inserted to maintain the documents structure rule that text follows a heading of markup level 1 (rather than A to D) (numeric level 4 rather than 0 to 3) 399

relatives? see ancestors and descendants 400

document ... 401

ancestors heading levels above the current heading level which it logically falls 402

Post-Modernism (Princeton, 1997) p. 11, with reference to Karl N. Llewellyn The Effect of Legal Institutions upon Economics, American Economic Review 15 (December 1925) pp 655-683, Mark M. Litchman Economics, the Basis of Law, American Law Review 61 (May-June 1927) pp 357-387, and W. S. Holdsworth A Neglected Aspect of the Relations between Economic and Legal History, Economic History Review 1 (January 1927-1928) pp 114-123.

[21] For a good introduction see Nicholas Mercuro and Steven G. Medema, *Economics and the Law: from Posner to Post-Modernism* (Princeton, 1997). These include: Chicago law and economics (New law and economics); New Haven School of law and economics; Public Choice Theory; Institutional law and economics; Neoinstitutional law and economics; Critical Legal Studies.

[22] puzzle away

under and to which it belongs (headings preceding current level under which it occurs)

decendants decendant headings are sub-headings beneath the current heading level, heading levels below the current heading level which are derived from it and belong to it (sub-headings contained beneath current level); decendant objects are the range of objects contained by a heading (ocn ranges for each heading in document body) 403

(document) sections a document can be divided into 3 parts: front; body and; back. Front matter includes the table of contents (which is generated from headings) and any parts of the document that are presented before the document body (this might include a copyright notice for example). The document body, the substantive part of the document, all its substantive objects, including: headings, paragraphs, tables, verse etc. This is followed by optional backmatter: endnotes, generated from inline markup; glossary, from section using a subset of regular markup, with an indication that section is to be treated as glossary. Note two things glossary might do that it does not, there is: no automatic (sorting) alphabetisation of listing; no creation of term anchor tags (perhaps it should); bibliography, created from a specially marked up section, with indication that section is to be treated as bibliography; bookindex generated from dedicated markup appended to objects providing index terms and the relevant range; blurb made up of ordinary markup, with indication that section is to be treated as blurb 404

segment, segmented text certain forms of output are conveniently segmented, e.g. epub and segmented html. The document is broken into chunks indicated by markup level 1 heading (numeric level 4 headings) as the significant level at which the document should be segmented, and including all decendant objects of that level. For a longer text/book this will usually the chapter level. (this is significant in e.g. for epub and segmented html, which are broken by segment, usually chosen to be chapter) 405

scroll the document as a ”scroll”, e.g. as a single text file, or continuous html document 406

object a unit of text. Objects include: headings; paragraphs; code blocks; grouped text; verse of poems; tables. Each substantive object is given an object number, that should make it citable. 407

ocn (object citation number / citation number) numbers assigned sequentially to each substantive object of a document. An ocn has the characteristic of remaining identical across output formats. Translations should be prepared so number remains identical across objects in different languages 408

unnumbered paragraph (place marker at end of paragraph) 409

unnumbered paragraph, delete when not required (place marker at end of paragraph) [used in dummy headings, eg. sometimes used for segmented html, e.g. to mark a prologue that is not otherwise identified as such as belonging to its own segment, segment will be created as such an placed in toc, but will not be found in scroll versions of the document] 410

citation number (see ocn / object citation number) 411

heading auto-numbering set in header, switched off in markup level 1 with an 412 appended minus 1 - or 1 given_segname-

document abstraction (== internal representation) intermediate step, preprocessing of document, into abstraction / representation that is used by all downstream processing, i.e. for all output formats. This allows normalisation, reducing alternative markup options to common representations, e.g. code blocks (open and close), tables, ways of instructing that text be bold, shortuct way of providing and endnote reference to a link 413

(document) internal representation (== document abstraction) see document abstraction 414

node representation 415

attribute (object attributes) when the document is abstracted attributes associated with an object, for example for a: paragraph, indent (hang ... check & add), bulleted, for a: code block, the language syntax, whether the block is numbered 416

inline markup when the document is abstracted, markup that remains embedded in the text, such as its font face (bold, italic, emphasis, underscore, strike, superscript, subscript), links, endnotes 417

sequential all objects backkeeping number? 418

8. Sample Commands 419

8.1. general 420

sdp/bin/sdp-ldc -v –epub –html –sqlite-update –output-dir=tmp/program-output data/sisupod/sisu-manual 421

time (sdp/bin/sdp-ldc -v –epub –html –sqlite-update –output-dir=tmp/program-output data/sisupod/*) 422

8.2. source & sisupod 423

sdp/bin/sdp-ldc -v –source –sisupod –output-dir=tmp/program-output data/sisudir/media/text/sisu-manual.sst 424

sdp/bin/sdp-ldc -v –source –sisupod –output-dir=tmp/program-output data/sisupod/sisu-manual 425

sdp/bin/sdp-ldc -v –source –sisupod –output-dir=tmp/program-output data/sisupod/* 426

8.3. sqlite 427

sdp/bin/sdp-ldc -v –sqlite-db-drop –output-dir=tmp/program-output 428

sdp/bin/sdp-ldc -v –sqlite-db-create –output-dir=tmp/program-output 429

sdp/bin/sdp-ldc -v –sqlite-db-recreate –output-dir=tmp/program-output 430

sdp/bin/sdp-ldc -v –sqlite-db-recreate –sqlite-insert –output-dir=tmp/program-output data/sisupod/* 431

sdp/bin/sdp-ldc -v –sqlite-db-recreate –sqlite-update –output-dir=tmp/program-output data/sisupod/* 432

sdp/bin/sdp-ldc -v –sqlite-db-drop –sqlite-db-create –sqlite-update –epub –html –output-dir=tmp/program-output data/sisupod/* 433

sdp/bin/sdp-ldc -v –sqlite-db-drop –sqlite-db-create –sqlite-update –epub –html –output-dir=tmp/program-output data/sisupod/* 434

sdp/bin/sdp-ldc -v –sqlite-db-drop –sqlite-db-create –sqlite-update –epub –html –output-dir=tmp/program-output data/sisupod/sisu-manual 435

sdp/bin/sdp-ldc -v –sqlite-db-drop –sqlite-db-create –sqlite-update –epub –html –output-dir=tmp/program-output data/sisupod/sisu-manual 436

sdp/bin/sdp-dmd -v –epub –html –output-dir=tmp/program-output data/sisudir/media/text/sisu_markup.sst 437

Endnotes

Book Index 439

Index

Blurb 440

SiSU was started in 1997, open-sourced in 2005.