Biloba STX

Parser Rules

Version 1.0
7-Apr-2004

Contents

1 Introduction **3**
- 1.1 About this Document . . . 3
- 1.2 Goal of Rules . . . 3

2 Structured Text Parser Rules **4**
- 2.1 Structural Rules . . . 4
  - 2.1.1 Paragraphs . . . 4
  - 2.1.2 Captions . . . 4
  - 2.1.3 Figures . . . 5
  - 2.1.4 Term and Definition . . . 5
  - 2.1.5 Quotes . . . 6
  - 2.1.6 Lists . . . 6
  - 2.1.7 Horizontal Delimiters . . . 7
- 2.2 Inline Rules . . . 8
  - 2.2.1 Linking to other Documents . . . 8
- 2.3 Processing Instructions . . . 8
  - 2.3.1 Comments . . . 8
  - 2.3.2 Pass-through . . . 9
  - 2.3.3 Pseudo-Paragraph . . . 9
  - 2.3.4 Escaping . . . 9
- 2.4 Rules Not Implemented in the Prototype . . . 9
  - 2.4.1 Footnotes . . . 9
  - 2.4.2 Abbreviations . . . 10
  - 2.4.3 Special Symbols . . . 10
  - 2.4.4 Tables . . . 10
  - 2.4.5 Referencing . . . 10

Chapter 1

Introduction

1.1 About this Document

This is a summary of formatting rules for Structured Text that were defined as part of the Final Year Project entitled *Document Formatting Systems* completed as part of the requirements for the BSc. (Hons) Computer Studies by Viktor C. Pavlu in the years 2003-2004. This document is also available on the CD that accompanies the project report.

Biloba implements most of these rules. Future extensions to Biloba should be implemented according to this document to retain a consistent source format and consistent formatting across different versions of the parser and other Structured Text parsers.

The syntax is made to be as intuitive as possible, however intuitive does not mean lax. There is a small set of strict rules that need to be obeyed when creating a structured document. This document describes these rules from a developer's point of view. For a user's point of view, please consult the *User's Guide* and *Expert's Guide*.

1.2 Goal of Rules

The rules must be intuitive, consistent, easy to remember and unambiguous while at the same time explicit markup should be avoided where possible. Users must be able to create a simple document right away without reading a manual.

Chapter 2

Structured Text Parser Rules

2.1 Structural Rules

The basic unit of text in Biloba is the line. A line is a sequence of characters terminated with CRLF (carriage return, line feed; hexadecimal: '`0D 0A`'), CR or LF (depending on operating system used) or terminated by EOF.

A line that consists of only whitespace and the delimiter is an empty or blank line.

Non-empty lines have a certain level of indentation, that is the number of blanks between the start of the line and the first non-whitespace character. Tab characters account for two blanks (this can be configured with the '`tab-size`' variable in %main.r).

An empty line or a change in indentation delimit blocks of text. The various block elements are described below.

2.1.1 Paragraphs

Multiple non-empty lines that share the same indentation level are joined to form one paragraph.

An empty line separates paragraphs.

2.1.2 Captions

A caption is a single line followed by one or more lines that are more indented.

Then the lower level and all following text elements on the same level form the body of the section in the document and the caption line serves as section heading.

Two lines followed by text on a lower level are not captions, rather the text is "spontaneously indented".

2.1.3 Figures

Text that is spontaneously indented, a line of text that starts with two blanks without a caption immediately before, is regarded as a figure.

Figure text is usually retained the way it was typed in and will be rendered using a non-proportional font so that the '`i`' and the '`X`' character have the same width. Whitespace is also preserved. These properties make figure test ideal for code samples or other examples within a technical document.

The layout of a figure depends on the type of the figure. By default no further processing is performed but by adding a figure header of the form '`#mode:image`', the figure text is interpreted by the *image* figure module. The *image* module interprets the figure text as reference to an image and inserts the image as figure.

A programmer can add figure modules to the system by writing a REBOL function with two parameters, *headers* and *lines*. The function processes the figure text contained in *lines* and the optional figure headers contained in *headers* to create a document node that represents the figure in the document tree. This function must be saved in the directory %modules/ under a filename which will be the name of the module. For details see the *image* module in '`modules/image`'.

By adding a figure header of the form '`#An Example`', the figure will be given the caption "An Example". All figures are numbered automatically as well.

In addition to the '`#mode:`' header any header of the general form '`#key:value`' can be added to the top of a figure. These key/value pairs are the way passing parameters to the figure modules (via *headers*).

2.1.4 Term and Definition

Inside a line[1], two dashes '`--`' are used to separate a term and its definition.

If the user wanted a hyphen instead of the terminus/definition pair, three dashes are required.

[1] In the current version of Biloba term/definition pairs are not allowed to span multiple lines.

2.1.5 Quotes

A paragraph enclosed in double quotation marks followed by two dashes and a name is rendered as a quote. The quoted text is the actual quote and the text after the dashes refers to the person that is quoted.

2.1.6 Lists

There are two kinds of lists in Biloba:

- itemized lists (unordered, "bulleted")
- enumerated lists (ordered, "numbered")

Lists are a group of paragraphs introduced with either a bullet or a number indicating the type of list. Every element in a list may span multiple lines and lists can be nested. An empty line delimits a list.

In list items spanning multiple lines, the subsequent lines must be aligned with the text rather than the bullet token.

As users sometimes intuitively indent lists to separate them from the rest of the text without the intention to create a new sub block, this manner is accounted for in Biloba and spontaneously indented lists are treated as if they were not indented.

The result is improved usability in most of the cases where lists are used, however it also introduces ambiguity if a list is the first element after a caption:

Multiple lines of text followed by an indented list are parsed as a paragraph followed by a list on the same level.

A single line of text followed by an indented list is parsed as a line followed by a list on the same level. This is what one would expect. However this clashes with the definition of a caption "...a single line followed by one or more lines that are more indented".

Therefore lists are not allowed to be the very first element after a caption if the caption is followed by an empty line unless the list is spontaneously indented in respect to the level of the sub block introduced by the caption (or double indented in other words).

If there is no blank line between the list and the caption, normal indentation is enough to discern the list in a sub block from a list indented for better readability only. The underlying assumption is that if accentuating the list was the only motivation for indentation, the user also would have added a blank line to bring

the list out more clearly — otherwise the list was indented on purpose yielding a caption and a list in a sub block.

Itemized Lists

Itemized lists are used to group text elements into a concise presentation where the elements do not appear in specific order.

The following tokens can be used to indicate an element of an itemized list:

- ‘`o text`’
- ‘`- text`’
- ‘`* text`’
- ‘`*) text`’

Ordered Lists

Ordered lists are similar to itemized lists but their elements’ order plays a role to the meaning of the text. Therefore the elements are usually numbered.

The following tokens can be used to indicate an element of an ordered list:

- ‘`1. text`’
- ‘`1, text`’
- ‘`1) text`’

Instead of ‘`1`’ any number can be used, however the actual numbering is done automatically by Biloba to allow easy re-ordering of elements.

2.1.7 Horizontal Delimiters

A line that consists of (at least 3) dashes only is regarded as a horizontal delimiter. Either a horizontal rule will be inserted or the text flow continues on the next page or there is a reasonable pause before the rest of the text is to be read out, depending entirely on the output media.

2.2 Inline Rules

All elements discussed so far were "structural" elements. They started a new block or represented a part of an existing block.

Inline formatting is done within the structural elements. Simple symbols inside the text are used to indicate the desired type of the text enclosed by the symbols. As the format of Biloba is declarative throughout, the actual rendered output depends on the output writer, however these symbols are commonly used in newsgroups to apply a certain type of emphasis to words:

strong text enclosed in asterisks will be typeset to bring it out stronger than other text

keyboard text enclosed in a pair of two single quotes is displayed as if typed in by the user

emphasized text enclosed in slashes will be emphasized

deleted text enclosed in tilde characters is used to denote something has been deleted

underlined text enclosed in underscore characters will be rendered underlined

2.2.1 Linking to other Documents

Hyperlinks to other documents are written as '`[label>>target]`' where label is the text that will be displayed and target is the resource that can be reached when following this link.

2.3 Processing Instructions

This section describes processing instructions to the parser. This is the only form of explicit markup in Biloba and will not be required for most uses.

2.3.1 Comments

Lines with a hash sign '`#`' as their very first character will be ignored by the parser. This can be used to add instructions targeted to your editor for example.

Text between the lines '`#ignore`' and '`#end`' is ignored entirely.

2.3.2 Pass-through

Text between the lines '`#preserve`' and '`#end`' is verbatim copied to the output writer encapsulated in a *preserve* node. No further processing is applied to these sections.

Pass-through sections can be used to directly manipulate the physical output of a document. For example it is perfectly valid to add TEX commands to typeset mathematical formulae in such a section. However if the document is then to be rendered in a format other than TEX, the TEX commands will not be interpreted but rather appear in the output as they were typed. Therefore note must be taken that this sacrifices the content/representation independence to a certain amount, nevertheless it is sometimes a powerful feature.

2.3.3 Pseudo-Paragraph

A line that consists of a single period '.' only will produce no output. It is solely there to circumvent ambiguities in indentation, for example after a list.

2.3.4 Escaping

The characters and their influences on the parser were outline in this document. Sometimes, however it is desired to have a hash sign '`#`' as the first character in a line without having the line ignored by the parser, or to have square brackets '`[]`' in a paragraph and not having them replaced with a hyperlink.

Adding a backslash character '\' in front of any character preserves the character as it is. The escaped character is not interpreted as formatting symbol of any kind. The backslash character itself can be added to a document by writing '\\'.

2.4 Rules Not Implemented in the Prototype

All rules this far were implemented and tested in the prototype of Biloba. The following rules act as a starting point for further work.

2.4.1 Footnotes

Adding footnotes to inline tokens while typing is done by appending '`^(footnote-text)`' to the word where the footnote should be added. The text between the parenthe-

ses will be displayed where the output writer deems it appropriate, usually on the bottom of the current page.

2.4.2 Abbreviations

The first time an abbreviation appears in the text it should be written out in parentheses. If the parser finds an abbreviation followed by text enclosed in double parentheses, the abbreviation and the spelled-out form from within the parentheses will be added to an internal synonym database.

Every time the abbreviation is used, the spelled-out form can be added by the output writer automatically, if desired.

The synonym database is intended to be created on the fly for each document alone, but it is also possible to have one central synonym database multiple authors are sharing.

2.4.3 Special Symbols

Special characters normally not available on a standard keyboard could be added via escape sequences. Mathematical operators, greek letters and Umlaute, ... are examples of this type of text.

2.4.4 Tables

Tables are not part of the Biloba rules but should be implemented by means of figure modules instead.

2.4.5 Referencing

Adding references to inline tokens while typing is done by appending '`^[ref-id]`' to the word where the reference should be added. *Ref-id* is an unique identifier of the source being referenced which Biloba looks up in a BibTEX database. Referenced entries will automatically be added to the bibliography at the end of the document.