# Contents

## Introduction

1 **Some basics**
   1.1 Introduction 5
   1.2 Available languages 5
   1.3 Switching 7
   1.4 Hyphenation 8
   1.5 Primitives 9
   1.6 Control 9
   1.7 Installing 10
   1.8 Modes 10

2 **Hyphenation**
   2.1 How it works 13
   2.2 The last words 15
   2.3 Explicit hyphens 16
   2.4 Extended patterns 17
   2.5 Exceptions 18
   2.6 Boundaries 19
   2.7 Plug-ins 20
   2.8 Blocking ligatures 24
   2.9 Special characters 25
   2.10 Counting 26
   2.11 Tracing 27

3 **Labels**
   3.1 Introduction 29
   3.2 Defining labels 29
   3.3 Using labels 29
   3.4 Hooks 30

4 **Numbering**
   4.1 Introduction 33
   4.2 Dates 36

5 **Typesetting**
   5.1 Introduction 39
   5.2 Spacing 39
   5.3 Frequencies 44
   5.4 Quotes 47
   5.5 Sentences 50
   5.6 Local control 51

6 **Goodies**
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>53</td>
</tr>
<tr>
<td>6.2</td>
<td>Spell checking</td>
<td>53</td>
</tr>
<tr>
<td>7</td>
<td>Sorting</td>
<td>57</td>
</tr>
<tr>
<td>7.1</td>
<td>Introduction</td>
<td>57</td>
</tr>
<tr>
<td>7.2</td>
<td>How it works</td>
<td>57</td>
</tr>
<tr>
<td>A</td>
<td>Appendix</td>
<td></td>
</tr>
<tr>
<td>A.1</td>
<td>The language files</td>
<td>63</td>
</tr>
<tr>
<td>A.2</td>
<td>The mtx-patterns script</td>
<td>63</td>
</tr>
<tr>
<td>A.3</td>
<td>Installed sorters</td>
<td>63</td>
</tr>
<tr>
<td>A.4</td>
<td>Verbose counters</td>
<td>72</td>
</tr>
</tbody>
</table>
Introduction

This document describes an important property of the \TeX typesetting system and Con-\TeXt in particular: the ability to deal with different languages at the same time. With languages we refer to natural languages. So, we’re not going to discuss the \TeX language itself, not MetaPost, nor Lua.

The original application of \TeX was English that uses the Latin script. The fonts that came with \TeX were suitable for that usage. When lines became too long they could be hyphenated using so called hyphenation patterns. Due to the implementation for many years there was a close relationship between fonts and hyphenation. Although at some point many more languages and scripts were supported, it was only when the Unicode aware variants showed up that hyphenation and fonts were decoupled. This makes it much more easier to mix languages that use different scripts. Although Greek, Cyrillic, Arabic, Chinese, Japanese, Korean and other languages have been supported for a while using (sometimes dirty) tricks, we now have cleaner implementations.

We can hyphenate words in all languages (and scripts) that have a need for it, that is, split it at the end of a line and add a symbol before and/or after the break. The way words are broken into parts is called hyphenation and so called patterns are used to achieve that goal. The way these patterns are constructed and applied was part of the research related to \TeX development. The method used is also applied in other programs and is probably one of the few popular ways to deal with hyphenation. There have been ideas about extensions that cover the demands of certain languages but so far nothing better has shown up. In the end \TeX does a pretty decent job and more advanced tricks don’t necessarily lead to better results.

Hyphenation is driven by a language number and that’s about it. This means that one cannot claim that \TeX in its raw form supports languages, other than that it can hyphenate and use fonts that provide the glyphs. It’s up to a macro package to wrap this into a mechanism that provides the user an interface. So, when we speak about language support, hyphenation is only one aspect. Labels, like the figure in figure 1.2 need to adapt to the main document language. When dates are shown they can be language specific. Scientific units and math function names can also be subjected to translation. Registers and other lists have to be sorted according to specific rules. Spacing differs per language.

In this manual we will cover some of functionality in Con\TeXt MkIV that relates to languages (and scripts). This manual is a compliment to other manuals, articles and documentation. Here we mostly focus on the language aspects. Some of the content (or maybe most) might looks alien and complex to you. This is because one purpose of this manual is to provide a place to wrap up some aspects of Con\TeXt. If you’re not interested in that, just stick to the more general manuals that also cover language aspects.

This document is still under construction. The functionality discussed here will stay and more might show up. Of course there are errors, and they’re all mine. The text is not
checked for spelling errors. Feel free to let me know what should get added.

Hans Hagen
PRAGMA ADE, Hasselt NL
2013 — 2016
1 Some basics

1.1 Introduction

In this chapter we will see how we can toggle between languages. A first introduction to patterns will be given. Some details of how to control the hyphenation with specific patterns will be given in a later chapter.

1.2 Available languages

When you use the English version of ConTeXt you will default to US English as main language. This means that hyphenation will be US specific, which by the way is different from the rules in GB. All labels that are generated by the system are also in English. Languages can often be accessed by names like `english` or `dutch` although it is quite common to use the short tags like `en` and `nl`. Because we want to be as compatible as possible with MkII, there are quite some synonyms. The following table lists the languages that for which support is built-in.¹

<table>
<thead>
<tr>
<th>tag</th>
<th>n</th>
<th>parent</th>
<th>file</th>
<th>synonyms</th>
<th>patterns</th>
<th>characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>af</td>
<td>6</td>
<td>af</td>
<td>af</td>
<td>afrikaans</td>
<td></td>
<td></td>
</tr>
<tr>
<td>agr</td>
<td>50</td>
<td>gr</td>
<td>agr</td>
<td>ancientgreek</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ala</td>
<td>56</td>
<td>la</td>
<td>ala</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar</td>
<td>26</td>
<td>ar</td>
<td>ar</td>
<td>arabic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar-ae</td>
<td>27</td>
<td>ar</td>
<td>ar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar-bh</td>
<td>28</td>
<td>ar</td>
<td>ar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar-dz</td>
<td>43</td>
<td>ar</td>
<td>ar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar-eg</td>
<td>29</td>
<td>ar</td>
<td>ar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar-in</td>
<td>30</td>
<td>ar</td>
<td>ar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar-iq</td>
<td>40</td>
<td>ar-sy</td>
<td>ar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar-jo</td>
<td>41</td>
<td>ar-sy</td>
<td>ar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar-kw</td>
<td>31</td>
<td>ar</td>
<td>ar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar-lb</td>
<td>42</td>
<td>ar-sy</td>
<td>ar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar-ly</td>
<td>32</td>
<td>ar</td>
<td>ar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar-ma</td>
<td>44</td>
<td>ar</td>
<td>ar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar-om</td>
<td>33</td>
<td>ar</td>
<td>ar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar-qa</td>
<td>34</td>
<td>ar</td>
<td>ar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar-sa</td>
<td>35</td>
<td>ar</td>
<td>ar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar-sd</td>
<td>36</td>
<td>ar</td>
<td>ar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar-sy</td>
<td>39</td>
<td>ar</td>
<td>ar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar-tn</td>
<td>37</td>
<td>ar</td>
<td>ar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar-ye</td>
<td>38</td>
<td>ar</td>
<td>ar</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

¹ More languages can be defined. It is up to users to provide the information.
<table>
<thead>
<tr>
<th>Code</th>
<th>Number</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>ca</td>
<td>53</td>
<td>catalan</td>
</tr>
<tr>
<td>cn</td>
<td>46</td>
<td>cn</td>
</tr>
<tr>
<td>cs</td>
<td>16</td>
<td>cz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>czech</td>
</tr>
<tr>
<td>da</td>
<td>4</td>
<td>da</td>
</tr>
<tr>
<td></td>
<td></td>
<td>danish</td>
</tr>
<tr>
<td>de</td>
<td>3</td>
<td>de</td>
</tr>
<tr>
<td></td>
<td></td>
<td>deu</td>
</tr>
<tr>
<td>de-at</td>
<td>11</td>
<td>de</td>
</tr>
<tr>
<td>de-ch</td>
<td>12</td>
<td>de</td>
</tr>
<tr>
<td>de-de</td>
<td>10</td>
<td>de</td>
</tr>
<tr>
<td>deo</td>
<td>9</td>
<td>deo</td>
</tr>
<tr>
<td>en</td>
<td>2</td>
<td>us</td>
</tr>
<tr>
<td></td>
<td></td>
<td>eng</td>
</tr>
<tr>
<td></td>
<td></td>
<td>english</td>
</tr>
<tr>
<td>en-gb</td>
<td>13</td>
<td>en</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gb</td>
</tr>
<tr>
<td></td>
<td></td>
<td>uk</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ukenglish</td>
</tr>
<tr>
<td>en-us</td>
<td>14</td>
<td>en</td>
</tr>
<tr>
<td></td>
<td></td>
<td>us</td>
</tr>
<tr>
<td></td>
<td></td>
<td>usenglish</td>
</tr>
<tr>
<td>es</td>
<td>52</td>
<td>es</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>spanish</td>
</tr>
<tr>
<td>es-es</td>
<td>59</td>
<td>es</td>
</tr>
<tr>
<td>es-la</td>
<td>60</td>
<td>es</td>
</tr>
<tr>
<td>fi</td>
<td>22</td>
<td>fi</td>
</tr>
<tr>
<td></td>
<td></td>
<td>finnish</td>
</tr>
<tr>
<td>fr</td>
<td>51</td>
<td>fr</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fra</td>
</tr>
<tr>
<td></td>
<td></td>
<td>french</td>
</tr>
<tr>
<td>gr</td>
<td>49</td>
<td>agr</td>
</tr>
<tr>
<td></td>
<td></td>
<td>greek</td>
</tr>
<tr>
<td>hr</td>
<td>18</td>
<td>hr</td>
</tr>
<tr>
<td></td>
<td></td>
<td>croatian</td>
</tr>
<tr>
<td>hu</td>
<td>23</td>
<td>hu</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hungarian</td>
</tr>
<tr>
<td>it</td>
<td>54</td>
<td>it</td>
</tr>
<tr>
<td></td>
<td></td>
<td>italian</td>
</tr>
<tr>
<td>ja</td>
<td>47</td>
<td>ja</td>
</tr>
<tr>
<td>kr</td>
<td>48</td>
<td>kr</td>
</tr>
<tr>
<td>la</td>
<td>55</td>
<td>la</td>
</tr>
<tr>
<td></td>
<td></td>
<td>latin</td>
</tr>
<tr>
<td>lt</td>
<td>45</td>
<td>lt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lithuanian</td>
</tr>
<tr>
<td>ml</td>
<td>64</td>
<td>ml</td>
</tr>
<tr>
<td></td>
<td></td>
<td>malayalam</td>
</tr>
<tr>
<td>nb</td>
<td>7</td>
<td>nb</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bokmal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>norwegian</td>
</tr>
<tr>
<td>nl</td>
<td>1</td>
<td>nl</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dutch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nld</td>
</tr>
<tr>
<td>nn</td>
<td>8</td>
<td>nn</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nynorsk</td>
</tr>
<tr>
<td>pl</td>
<td>15</td>
<td>pl</td>
</tr>
<tr>
<td></td>
<td></td>
<td>polish</td>
</tr>
<tr>
<td>pt</td>
<td>57</td>
<td>pt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>portuguese</td>
</tr>
<tr>
<td>pt-br</td>
<td>58</td>
<td>pt</td>
</tr>
<tr>
<td>ro</td>
<td>61</td>
<td>ro</td>
</tr>
<tr>
<td></td>
<td></td>
<td>romanian</td>
</tr>
<tr>
<td>ru</td>
<td>20</td>
<td>ru</td>
</tr>
<tr>
<td></td>
<td></td>
<td>russian</td>
</tr>
</tbody>
</table>
You can call up such a table with the following commands:

\usemodule[languages-system]
\loadinstalledlanguages
\showinstalledlanguages

Instead you can run context --global languages-system.mkiv.

As you can see, many languages have hyphenation patterns but for Japanese, Korean, Chinese as well as Arabic languages they make no sense. The patterns are loaded on demand. The number is the internal number that is used in the engine; a user never has to use that number. Numbers < 1 are used to disable hyphenation. The file tag is used to locate and load a specification. Such files have names like type lang-nl.lua.

Some languages share the same hyphenation patterns but can have demands that differ, like labels or quotes. The characters shown in the table are those found in the pattern files. The number of patterns differs a lot between languages. This relates to the systematic behind them. Some languages use word stems, others base their hyphenation on syllables. Some language have inflections which adds to the complexity while others can combine words in ways that demand special care for word boundaries. Of course a low or high number can signal a low quality as well, but most pattern collections are assembled over many years and updated when for instance spelling rules change. I think that we can safely say that most patterns are quite stable and of good quality.

1.3 Switching

The document language is set with

\mainlanguage[en]

but when you want to apply the proper hyphenation rules to an embedded language you can use:

\language[en]

or just:
Some basics

The main language determines what labels show up, how numbering happens, in what way dates get formatted, etc. Normally the `\mainlanguage` command comes before the `\starttext` command.

### 1.4 Hyphenation

In LuaTeX each character that gets typeset not only carries a font id and character code, but also a language number. You can switch language whenever you want and the change will be carried with the characters. Switching within a word doesn’t make sense but it is permitted:

1. \texttt{\de incredibly\kompliziert} \hspace{1cm} \texttt{incredibly-kompliziert}
2. \texttt{\en incredibly\kompliziert} \hspace{1cm} \texttt{incredibly-kompliziert}
3. \texttt{\en incredibly\texttt{\de kompliziert}} \hspace{1cm} \texttt{incredibly-kompliziert}
4. \texttt{\en incredibly\texttt{\de -kompliziert}} \hspace{1cm} \texttt{incredibly-kompliziert}
5. \texttt{\en incredibly\texttt{\de -kompliziert}} \hspace{1cm} \texttt{incredibly-kompliziert}

In the line 4 we have a `-` between the two words, and in the last line just a `-`. If you look closely you will notice that the snippets can be quite small. If we typeset a word with a 1mm text width we get this:

\begin{verbatim}
incredibly-kompliziert
\end{verbatim}

If you are familiar with the details of hyphenation, you know that the number of characters at the end and beginning of a word is controlled by the two variables `\lefthyphenmin` and `\righthyphenmin`. However, these only influence the hyphenation process. What bits and pieces eventually end up on a line is determined by the par builder and there the `\hsize` matters. In practice you will not run into these situations, unless you have extreme long words and a narrow column.

Hyphenation normally is limited to regular characters that make up the alphabet of a language. It is insensitive for capitalization as the following text shows:

This time the mu-si-cal dis-trac-tion while de-vel-op-ing code came from watch-ing youtube per-for-mances of Cory Henry (also known from Snarky Puppy, a con-glom-er-ate of ex-cel-lent play-ers). Just search the web for his name with ‘Stevie Won-der and Michael Jack-son Tribute’. There is no key-board he can't play. Another in-ter-est-ing key-board player is Sun Rai (a short name for Rai Thistle-thwaite, just google for ‘The Bea-tles, Come To-gether, Live Piano Acoustic with Loop Pedal’, or do a com-bined search with ‘Matt Cham-ber-lain’. Okay, and talk-ing of key-boards, let’s not for-get Vika Yer-molyeva (vk-goeswild) as she’s one of
Some basics

a kind too on the web. And then there is Ja-cob Col-lier, in one word: in-cred-i-ble (or hyphenated the Dutch way in-cre-di-ble, let me re-peat that in French in-cre-di-ble).\footnote{Get me right, there are of course many more fantastic musicians.}

Of course, names are often short and don’t need to be hyphenated (or the left and right settings prohibit it). Another complication with names is that they can come from another language so we either need to switch language temporarily or we need to add an exception (more about that later).

1.5 Primitives

In traditional \TeX the language is not a property of a character but is triggered by a signal in the (so called) list. Think of:

```
<language 1>this is <language 2>nederlands<language 1> mixed with english
```

This number is set by the primitive \textbackslash language. Language triggers are injected into the list depending on the value of this number. There is also a \textbackslash setlanguage primitive that can inject triggers without setting the \textbackslash language number. Because in Lua\TeX the state is kept with the character you don’t need to worry about the subtle differences here.

In Con\TeXt the \textbackslash language and \textbackslash setlanguage commands are overloaded by a more advanced switch macro. You cannot assume that they work as explained in general manuals about \TeX. Currently you can still assign a number but that might change. Just consider the language to be an abstraction and don’t mess with this number. Both commands not only change the current language but also do specific initializations when needed.

What characters get involved in hyphenation is historically determines by the so called \textbackslash lccode values. Each character can have such a value which maps an uppercase to a lowercase character. This concept has been extended in \varepsilon-\TeX where it binds to a pattern set (language). However, in Con\TeXt the user never has to worry about such details.

In traditional hyphenation there will not be hyphenated if the sum of \textbackslash lefthyphenmin and \textbackslash righthyphenmin exceeds 62. This limitation is not present in the to be presented Lua variant of this routine as there is no good reason for this limitation other than implementation constraints.

1.6 Control

We already mentioned \textbackslash lefthyphenmin and \textbackslash righthyphenmin. These two variables control the area in a word that is subjected to hyphenation. Setting these values is a matter of taste but making them too small can result in bad hyphenation when the patterns are
made with the assumptions that certain minima are used. Using a `\lefthyphenmin` of 2 while the patterns are made with a value of 3 in mind is a bad idea.

<table>
<thead>
<tr>
<th>\lefthyphenmin</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>in-ter-est-ing</td>
<td>in-ter-est-ing</td>
<td>in-ter-est-ing</td>
<td>in-ter-est-ing</td>
<td>in-ter-est-ing</td>
</tr>
<tr>
<td>2</td>
<td>in-ter-est-ing</td>
<td>in-ter-est-ing</td>
<td>in-ter-est-ing</td>
<td>in-ter-est-ing</td>
<td>in-ter-est-ing</td>
</tr>
<tr>
<td>3</td>
<td>inter-est-ing</td>
<td>inter-est-ing</td>
<td>inter-est-ing</td>
<td>inter-est-ing</td>
<td>inter-est-ing</td>
</tr>
<tr>
<td>4</td>
<td>inter-est-ing</td>
<td>inter-est-ing</td>
<td>inter-est-ing</td>
<td>inter-est-ing</td>
<td>inter-est-ing</td>
</tr>
<tr>
<td>5</td>
<td>inter-est-ing</td>
<td>inter-est-ing</td>
<td>inter-est-ing</td>
<td>inter-est-ing</td>
<td>inter-est-ing</td>
</tr>
</tbody>
</table>

When \TeX breaks a paragraph into lines it will try so without hyphenation. When that fails (read: when the badness becomes too high) a next effort will take hyphenation into account.\(^3\) When the badness is still too high, an optional emergency pass can be made but only when the tolerances are set to permit this. In Con\TeX you can try these settings when you get too many over- or underfull boxes reported on the console.

\texttt{\setupalign[tolerant]}
\texttt{\setupalign[verytolerant]}
\texttt{\setupalign[verytolerant,stretch]}

Personally I tend to use the last setting, especially in automated flows. After all, \TeX will not apply stretch unless it’s really needed.

The two `\*hyphenmin` parameters can be set any time and the current value is stored with each character. They can also be set with the language which we will see later.

When \TeX hyphenates words it has to decide where a word starts and ends. In traditional \TeX the words starts normally at a character that falls within the scope of the hyphenator. It ends at when a box (hlist or vlist) is seen, but also at a rule, discretionary, accent (forget about this in Con\TeX) or math. An example will be given in the chapter that discussed the Lua alternative.

### 1.7 Installing

todo

### 1.8 Modes

Languages are one of the mechanisms where you can access the current state. There are for instance two (official) macros that contain the current (main) language:

\(^3\) Because in Lua\TeX we always hyphenate there is no real gain in trying not to hyphenate. Because in traditional \TeX hyphenation happens on the fly a pass without hyphenating makes more sense.
When we have set `\language[nl]` we get this:

<table>
<thead>
<tr>
<th>macro</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\currentmainlanguage</code></td>
<td>en</td>
</tr>
<tr>
<td><code>\currentlanguage</code></td>
<td>en</td>
</tr>
</tbody>
</table>

If you write a style that needs to adapt to a language you can use modes. There are several ways to do this:

- `\language[nl]`
- `\startmode[**en]`
  - `\color[darkred]{main english}`
  - `\stopmode`
- `\startmode[*en]`
  - `\color[darkred]{local english}`
  - `\stopmode`
- `\startmode[**nl]`
  - `\color[darkblue]{main dutch}`
  - `\stopmode`
- `\startmode[*nl]`
  - `\color[darkblue]{local dutch}`
  - `\stopmode`
- `\startmodeset`
  - `[*en] {\color[darkgreen]{english set}}`
  - `[*nl] {\color[darkgreen]{dutch set}}`
- `\stopmodeset`

This typesets:

- **main english**
- *local dutch*
- **dutch set**

When you use setups you can use the following trick:

- `\language[nl]`
\startsetups language:en
  \color[darkorange]{something english}
\stopsetups

\startsetups language:nl
  \color[darkorange]{something dutch}
\stopsetups

\setups{language:currentlanguage}

As expected we get:

something dutch
2 Hyphenation

2.1 How it works

Proper hyphenation is one of the strong points of \TeX. Hyphenation in \TeX is done using so called hyphenation patterns. Making these patterns is an art and most users (including me) happily use whatever is available. Patterns can be created automatically using \texttt{patgen} but often manual tweaking is needed too. A pattern looks as follows:

\begin{verbatim}
  \textbf{pattern}
\end{verbatim}

This means as much as: you can split the word \texttt{pattern} in two pieces, with a hyphen between the two t’s. Actually it will also split the word \texttt{patterns} because the hyphenation mechanism looks at substrings. When no number between characters in a pattern is given, a zero is assumed. This means as much as \texttt{undefined}. An even number inhibits hyphenation, an odd number permits it. The larger the number (weight), the more influence it has. A more restricted pattern is:

\begin{verbatim}
  .\textbf{pattern}.
\end{verbatim}

Here the periods set the word boundaries. The pattern dictionary for us english has smaller patterns and the next trace shows how these are applied.

\begin{verbatim}
   .\textbf{pattern} . \textbf{pattern} .
   1 1p0a0 100000000
   2 0a0t5t0e0 100500000
   3 4t3t2 104520000
   .1p0a4t5t2e0r0n0. .\textbf{pattern} .
\end{verbatim}

The effective hyphenation of a word is determined by several factors:

- the current language, each language can have different patterns
- the characters, as some characters might block hyphenation
- the settings of isd\texttt{lefthyphenmin} and isd\texttt{righthyphenmin}

A place where a word can be hyphenated is called a discretionary. When \TeX analyzes a stream, it will inject discretionary nodes into that stream.

\begin{verbatim}
  \texttt{pat/discretionary{-}{-}{-}{-}{-}{-}}\texttt{tern.}
\end{verbatim}

In traditional \TeX hyphenation, ligature building and kerning are tightly interwoven which is quite effective. However, there was also a strong relationship between the current font and hyphenation. This is a side effect of traditional \TeX having at most 256 characters in a font and the fact that the used character is fact a reference to a slot in a font. There a character in the input initially ends up as a character node and eventually becomes a glyph node. For instance two characters fi can become a ligature glyph
representing this combination.

In Lua\TeX{} the hyphenation, ligature building and kerning stages are separated and can be overloaded. In Con\TeX{}t all three can be replaced by code written in Lua. Because normally hyphenation happens before font logic is applied, there is no relationship with font encoding. I wrote the first Lua version of the hyohenator on a rainy weekend and the result was not that bad so it was presented at the 2014 Con\TeX{}t meeting. After some polishing I decided to add this routine to the standard MkIV repertoire which then involved some proper interfacing.

You can enable the Lua variant with the following command:

\begin{verbatim}
\ setupsphynenation [method=traditional]
\end{verbatim}

We call this method traditional because in principle we can have many more methods and this one is (supposed to be) mostly compatible to the built-in method. This is a global setting. You can switch back with:

\begin{verbatim}
\ setupsphynenation [method=default]
\end{verbatim}

In the next sections we will see how we can provide alternatives within the traditional method. These alternatives can be set local and therefore can operate over a limited range of characters.

One complication in interfacing is that \TeX{} has grouping (which permits local settings) and we want to limit some of the above functionality using groups. At the same time hyphenation is a paragraph related action so we need to enable the hyphenation related code at a global level (or at least make sure that it gets exercised by forcing a \texttt{\par}). That means that the alternative hyphenator has to be quite compatible so that we could just enable it for a whole document. This can have an impact on performance but in practice that can be neglected. In Lua\TeX{} the Lua variant is 4 times slower than the built-in one, in Luajit\TeX{} it’s 3 times slower. But the good news is that the amount of time spent in the hyphenator is relatively small compared to other manipulations and macro expansion. The additional time needed for loading and preparing the patterns into a more Lua specific format can be neglected.

You can check how words get hyphenated using the patterns management script:

\begin{verbatim}
> mtxrun --script patterns --hyphenate language
\end{verbatim}

\begin{verbatim}
hyphenator  |      language language
hyphenator  | 0a2n0     0 0 2 0 0 0 0 0 0
hyphenator  | 2a0n0g0   0 2 2 0 0 0 0 0 0
hyphenator  | 0n1g0u0   0 2 2 1 0 0 0 0 0
hyphenator  | 0g0u4a0   0 2 2 1 0 4 0 0 0
hyphenator  | 2g0e0.0   0 2 2 1 0 4 2 0 0
\end{verbatim}
2.2 The last words

Mid 2014 we had to upgrade a style for a pdf assembly service: chapters from (technical) school books are combined into arbitrary new books. There are some nasty aspects with this flow: for instance, all section numbers in a chapter are replaced by new numbers and this also involves figure and table prefixes. It boils down to splitting up books, analyzing the typeset content and preparing it for replacements. The structure is described in xml files so that we can generate tables of contents. The reason for not generating from xml sources is that the publisher doesn’t have a xml workflow and that books already were available. Also, books from several series are combined and even within a series structure (and rendering) differs.

What has this to do with hyphenation? Writing a style for such a flow always results in a more complex one that estimated and as usual it’s in the details. The original style was written in MkII and used some box juggling to achieve reasonable results but in MkIV we can do better.

Each chapter has a title and books get titles and subtitles as well. The titles are typeset each time a new book is composed. This happens within some layout constraints. Think of constraints like these:

- the title goes on top of a shape that doesn’t permit much overflow
- there can be very long words (not uncommon in Dutch or German)
- a short word or hyphenated part should not end up on the last line
- the left and right hyphenation minima are at least four

The last requirement is a compromise because in most cases publishers seem to want ragged right not hyphenated rendering (at least in Dutch schoolbooks). The arguments for this are quite weak and probably originate in fear of bad rendering given past experiences. It’s this kind of situations that drive the development of the more obscure features that ship with ConTExt and a (partial) solution for this specific case will be given later.

If you look at thousands of titles and turn these into (small) paragraphs TeX does a pretty good job. It’s the few exceptions that we need to catch. The next examples demonstrate such an extreme case.

| 1 | a very long word and then an even longer word | a very long word and then an even longer word |
Of course in practice there need to be some reasonable width and when we pose these limits the longest possible word should fit into the allocated space. In these examples the rule shows the width. In the right columns we see a red colored word and that one will not get hyphenated.

### 2.3 Explicit hyphens

Another special case that we needed to handle were (compound) words with explicit hyphens. Because often data comes from xml files we can not really control the typesetting as in a $\LaTeX$ document where the author sees what gets done. So here we need a way to turn these hyphens into proper hyphenation directives and at the same time permit the words to be hyphenated.
2.4 Extended patterns

As with more opened up mechanisms, in MkIV we can extend functionality. As an example I have implemented the extensions discussed in the article by László Németh in the Proceedings of EuroTeX 2006: *Hyphenation in OpenOffice.org* (TUGboat, Volume 27, 2006). The syntax for these extension is somewhat ugly and involves optional offsets and ranges.\(^4\)

```
\registerhyphenationpattern[nl][elë/e=e]
\registerhyphenationpattern[nl][a9atje./a=t,1,3]
\registerhyphenationpattern[en][eigh1tee/t=t,5,1]
\registerhyphenationpattern[de][c1k/k=k]
\registerhyphenationpattern[de][schif1f/ff=f,5,2]
```

These patterns result in the following hyphenations:

- reëel → re-eel
- omaatje → oma-tje
- eighteen → eight-teen
- Zucker → Zuk-ker
- Schiffahrt → Schiff-fahrt

In a specification, the . indicates a word boundary and numbers indicate the weight of a breakpoint. The optional extended specification comes after the /. The values separated by a = are the pre and post sequences: these end up at the end of the current line and

\(^4\) I’m not sure if there were ever patterns released that used this syntax.
beginning of the next one. The optional numbers are the start position and length. These default to 1 and 2, so in the first example they identify eë (the weights don’t count).

There is a pitfall here. When the language already has patterns that for instance prohibit a hyphen between e and type è, like e2ë, we need to make sure that we give our new one a higher priority, which is why we used a e9ë.

This feature is somewhat experimental and can be improved. Here is a more Lua-ish way of setting such patterns:

```lua
local registerpattern =
  languages.hyphenators.traditional.registerpattern

registerpattern("nl","elë", {
    start  = 1,
    length = 2,
    before = "e",
    after  = "e",
})

registerpattern("nl","a9atje./a=t,1,3")
```

Just adding extra patterns to an existing set without much testing is not wise. For instance we could add these to the dutch dictionary:

```latex
\registerhyphenationpattern[nl][e3ë/e=e]
\registerhyphenationpattern[nl][o3ë/o=e]
\registerhyphenationpattern[nl][e3ï/e=i]
\registerhyphenationpattern[nl][i3ë/i=e]
\registerhyphenationpattern[nl][a5atje./a=t,1,3]
\registerhyphenationpattern[nl][toma8at5je]
```

That would work oke well for words like

coëfficiënt
geïntroduceerd
copiëren
omaatje
tomaatje

However, the last word only goes right because we explicitly added a pattern for it. One reason is that the existing patterns already contain rules to prevent weird hyphenations. The same is true for the accented characters. So, consider these examples and coordinate additional patterns with other users so that errors can be identified.

### 2.5 Exceptions

We have a variant on the \TeX\ primitive `\hyphenation`, the official way to register a specific way to hyphenate a word.
\registerhyphenationexception[aaaaa-bbbbb]
aaaaabbbbb \par

This code is self explaining and results in:

aaaaa-
bbbbbb

There can be multiple hyphens and even multiple words in such a specification:

\registerhyphenationexception[aaaaa-bbbbb cc-ccc-ddd-dd]
aaaaabbbbb \par
ccccccddddd \par

We get:

aaaaa-
bbbbbb
c-
ccc-
ddd-
dd

2.6 Boundaries

A box, rule, math or discretionary will end a word and prohibit hyphenation of that word. Take this example:

whatever \par
whatever\hbox{!} \par
\vl whatever\vl \par
whatever$x$ \par
whatever-whatever \par

These lines will hyphenate differently and in traditional \TeX you need to insert penalties and/or glue to get around it. In the Lua variant we can enable that limitation.

\definehyphenationfeatures
[strict]
[rightedge=tex]

Here we show the three variants: traditional \TeX and Lua with and without strict settings.

<table>
<thead>
<tr>
<th>default</th>
<th>traditional</th>
<th>traditional strict</th>
</tr>
</thead>
</table>

By default ConTeXt is configured to hyphenate words that start with an uppercase character. This behaviour is controlled in \TeX by the \texttt{\uchyph} variable. A positive value will enable this and a negative one disables it.

<table>
<thead>
<tr>
<th>default 0</th>
<th>default 1</th>
<th>traditional 0</th>
<th>traditional 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TExi-</td>
<td>TEx-</td>
<td>TExi-</td>
<td>TEx-</td>
</tr>
<tr>
<td>fied</td>
<td>i-</td>
<td>fied</td>
<td>i-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Lua variants behaves the same as the built-in implementation (that of course remains the reference).

## 2.7 Plug-ins

The default hyphenator is similar to the built-in one, with a couple of extensions as mentioned. However, you can plug in your own code, given that it does return a proper hyphenation result. One reason for providing this plug is that there are users who want to play with hyphenators based on a different logic. In ConTeXt we already have some methods to deal with languages that (for instance) have no spaces but split on words or syllabes. A more tight integration with the hyphenator can have advantages so I will explore these options when there is demand.

A result table indicates where we can break a word. If we have a four character word and can break after the second character, the result looks like this:

```latex
result = \{ false, true, false, false \}
```

Instead of \texttt{true} we can also have a table that has entries like the extensions discussed in a previous section. Let’s give an example of a plug-in.

```latex
\startluacode
local subset = \{
```
a = true,
e = true,
i = true,
o = true,
u = true,
y = true,
}
languages.hyphenators.traditional.installmethod("test",
  function(dictionary,word,n)
    local t = { }
    for i=1,#word do
      local w = word[i]
      if subset[w] then
        t[i] = {
          before = "<" .. w,
          after = w .. ",",
          left = false,
          right = false,
        }
      else
        t[i] = false
      end
    end
    return t
  end
)
\stopluacode

Here we hyphenate on vowels and surround them by angle brackets when split over lines. This alternative is installed as follows:

\definehyphenationfeatures
  [demo]
  [alternative=test]

We can now use it as follows:

\setuphyphenation[method=traditional]
\sethyphenationfeatures[demo]

When applied to one the tufte example we get:

We thrive in information-thick worlds because of our marvelous and everyday c<a
a>acity to select, edit, single out, structure, highlight, group, pair, merge, harm<o
o>ize, synthesize, focus, organize, condense, reduce, boil down, choose, categ<o
Hyphenation

A more realistic (but not perfect) example is the following:

\startluacode
local packslashes = false
local specials = {
    ["!"]  = "before", ["?" ] = "before",
    [":" ] = "before", ["::"] = "before",
    ["/" ] = "before", ["\" "] = "before",
    ["#" ] = "before",
    ["$" ] = "before",
    ["%" ] = "before",
    ["&" ] = "before",
    ["*" ] = "before",
    ["+" ] = "before", ["-" ] = "before",
    ["," ] = "before", ["." ] = "before",
    [":" ] = "before", [";" ] = "before",
    ["<" ] = "before", [">" ] = "before",
    ["=" ] = "before",
    ["@" ] = "before",
    ["{" ] = "before",
    ["[" ] = "before",
    ["{" ] = "before",
    ["^" ] = "before", ["_" ] = "before",
    ["\" "] = "before",
    ["|" ] = "before",
    ["-" ] = "before",
    [")" ] = "after",
    ["]" ] = "after",
    ["}" ] = "after",
}

languages.hyphenators.traditional.installmethod("url",
    function(dictionary,word,n)
        local t = { }
        local p = nil
        for i=1,#word do
"o>ize, catalog, classify, list, abstract, scan, look into, idealize, discriminate, discriminate, distinguish, screen, pigeonhole, pick over, sort, integrate, blend, inspect, filter, lump, skip, smooth, chunk, average, approximate, cluster, aggregate, o<u u>line, summarize, itemize, review, dip into, flip through, browse, glance into, leaf through, skim, refine, enumerate, glean, synopsize, winnow the wheat from the chaff and separate the sheep from the goats.
local w = word[i]
local s = specials[w]
if s == "after" then
    s = {
        start = 1,
        length = 1,
        after = w,
        left = false,
        right = false,
    }
    specials[w] = s
elseif s == "before" then
    s = {
        start = 1,
        length = 1,
        before = w,
        left = false,
        right = false,
    }
    specials[w] = s
end
if not s then
    s = false
elseif w == p and w == "/" then
    t[i-1] = false
end
    t[i] = s
if packslashes then
    p = w
end
end
return t end

\stopluacode

Again we define a plug:

\definehyphenationfeatures
[url]
[characters=all, alternative=url]

So, we only break a line after symbols.
A quick test can look as follows:

\starthyphenation[traditional]  
  \sethyphenationfeatures[url]  
  \tt  
  \dontcomplain  
  \hs 1mm  
  http://www.pragma-ade.nl  
\stop hyphenation

Or:

http:  
/  
/  
www.  
pragma-ade.nl

2.8 Blocking ligatures

Yet another predefined feature is the ability to block a ligature. In traditional \TeX this can be done by putting a \{\} between the characters, although that effect can get lost when the text is manipulated. The natural way to do this in a Unicode environment is to use the special characters zwj and zwnj.

We use the following example lines:

supereffective \blank  
superef\zwnj fective

and define two feature sets:

\definehyphenationfeatures  
  [demo-1]  
  [characters=\zwnj \zwj,  
    joiners=yes]  
\definehyphenationfeatures  
  [demo-2]  
  [joiners=no]

We limit the width to 1mm and get:
2.9 Special characters

The characters example can be used (to some extend) to do the same as the breakpoints mechanism (compounds).

\definehyphenationfeatures
  [demo-3]
  [characters={()}]]
\starthyphenation[traditional]
  \sethyphenationfeatures[demo-3]
  \dontcomplain
  \hsizelmm
  \we use (super)special(ized) patterns
\stophyphenation

\we
\use
(su-
per)spe-
cial(ized)

\pat-
terns

We can make this more clever by adding patterns:

\registerhyphenationpattern[en][]9]
\registerhyphenationpattern[en][9]

This gives:

\we
\use
Hyphenation

A detailed trace shows that these patterns get applied:

```
. ( s u p e r ) s p e c i a l ( i z e d ) . . ( s u p e r ) s p e c i a l ( i z e d ) .
   1  9(0)                                           9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   2  1s0u0                                          9 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   3  0u1p0e0                                        9 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   4  0)9                                            9 1 0 1 0 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   5  0p0e2c0                                         9 1 0 1 0 0 0 9 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   6  0e0c2i0                                         9 1 0 1 0 0 0 9 0 0 0 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0
   7  1c0i0                                           9 1 0 1 0 0 0 9 0 0 0 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0
   8  3c0i0a0                                         9 1 0 1 0 0 0 9 0 0 0 3 2 0 0 0 0 0 0 0 0 0 0 0 0 0
   9  2i1a0                                           9 1 0 1 0 0 0 9 0 0 0 3 2 1 0 0 0 0 0 0 0 0 0 0 0 0
  10  0i2a0l0                                         9 1 0 1 0 0 0 9 0 0 0 3 2 2 0 0 0 0 0 0 0 0 0 0 0 0
  11  9(0)                                           9 1 0 1 0 0 0 9 0 0 0 3 2 2 0 9 0 0 0 0 0 0 0 0 0 0
  12  2z0e0                                          9 1 0 1 0 0 0 9 0 0 0 3 2 2 0 9 0 2 0 0 0 0 0 0 0 0
  13   0)9                                            9 1 0 1 0 0 0 9 0 0 0 3 2 2 0 9 0 2 0 0 0 0 0 0 0 0
```

The somewhat weird hyphens at the edges will in practice not show up because there is always one regular character there.

### 2.10 Counting

There is not much you can do about patterns. It’s a craft to make them and so they are shipped with the distribution. In order to hyphenate well, TeX looks at some character properties. In ConTeXt only the characters used in the patterns of a language get tagged as valid in a word.

The following example illustrates that there can be corner cases. In fact, this example might render differently depending on the patterns available. First we define an extra language, based on French.

```
\installlanguage[frf][default(fr),patterns=fr,factor=yes]
```

Here we set the factor parameter which tells the loader that it should look at the characters used in a special way: some count for none, and some count for more than one when determining the min values used to determine if and where hyphenation is to be applied.

```
\startmixedcolumns[n=3,balance=yes]
   \hsize 1mm \dontcomplain
   \language[fr] aesop oedipus æsop œdipus \column
   \hsize 1mm \dontcomplain
```

Hyphenation
We get three (when writing this manual) different columns:

\begin{tabular}{ccc}
\text{ae-} & \text{ae-} & \text{ae-} \\
\text{sop} & \text{sop} & \\
\text{oe-} & \text{oe-} & \\
\text{di-} & \text{di-} & \\
\text{pus} & \text{pus} & \\
\text{æsop} & \text{æsop} & \text{æ-} \\
\text{œdi-} & \text{œ-} & \\
\text{pus} & \text{di-} & \\
\end{tabular}

The trick is in the factor: when set to yes an æ is counted as two characters. Combining marks count as zero but you will not find them being used as we already resolve them in an earlier stage.

U+000C6 Æ 2 letter
U+000DF ß 2 letter
U+000E6 æ 2 letter
U+00132 İJ 2 dubious
U+00133 ij 2 dubious
U+00152 Ç 2 dubious
U+00153 œ 2 dubious
U+001C7 LJ 2 letter
U+001C8 Lj 2 letter

It is very unlikely to find an æ in the input and even an ij is rare. The æ is marked as character and the œ a ligature in Unicode. Maybe all the characters here are dubious but at least we provide a way to experiment with them.

2.11 Tracing

Among the tracing options (low level trackers) there is one for pattern developers:

\usemodule{s-languages-hyphenation}

\startcomparepatterns[de,nl,en,fr]
\input zapf \quad (\showcomparepatternslegend)
The different hyphenation points are shown with colored bars. Some valid points might not be shown because the font engine can collapse successive discretionary.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC’s tricks, and think that a widely-praised program, called up on the screen, will make everything automatic from now on.
3 Labels

3.1 Introduction

When we started using \TeX, I naturally started with plain \TeX. But it didn’t take long before we tried \LaTeX. Because our documents were in Dutch one of the first fights with this package was to get rid of the English labels. Because rather soon we decided to cook up an alternative package, a decent label mechanism was one of the first things to show up. And as soon as multiple language typesetting gets into view, such a mechanism becomes one of those language dependent features. In this chapter the basics will be covered.

3.2 Defining labels

Before we define a label we need to define a label class. You probably seldom need that but this is how it’s done:

\begin{verbatim}
definelabelclass [mylabel]
\end{verbatim}

There are some classes predefined:

- **head**: (complete) titles like and
- **label**: in–text labels like and Figure
- **mathlabel**: function names like sin and cos
- **taglabel**: labels used for tagging purposed in the backend
- **btxlabel**: labels used in typesetting bibliographic items

The physical units mechanism also uses labels: unit, operator, prefix and suffix. All these labels are defined per language with a fall back on English.

Given that we have defined class *mylabel*, a label itself is set like this:

\begin{verbatim}
\setupmylabeltext
 [en]
 [first=\{after first},
  second=\{before second\},\{after second\}]
\end{verbatim}

The first argument (the language) is optional. In the next section we will see how these labels are used. A lot of labels are predefined, in MkIV this happens in the file lang-txt.lua. There is no need to adapt this file as you can always add labels run time.

3.3 Using labels

How a label is called depends on the way it needs to be used. In any case the main language set determines the language of the label. So, when in an Dutch text we temporary
switch to German, the Dutch labels are used.

<table>
<thead>
<tr>
<th>command</th>
<th>first</th>
<th>second</th>
</tr>
</thead>
<tbody>
<tr>
<td>\leftmylabeltext{tag}</td>
<td>&lt;after first</td>
<td>before second&gt;</td>
</tr>
<tr>
<td>\rightmylabeltext{tag}</td>
<td>&lt;after second</td>
<td></td>
</tr>
<tr>
<td>\mylabeltext{tag}</td>
<td>&lt;after first</td>
<td>before second&gt;</td>
</tr>
<tr>
<td>\mylabeltexts{tag}{text}</td>
<td>&lt;after firsttext</td>
<td>before second&gt;text&lt;after second</td>
</tr>
</tbody>
</table>

### 3.4 Hooks

Some mechanisms have label support built in, most noticeably sections heads and numbered items, like figure captions.

```latex
\definehead
  [myhead]
  [subsection]

\setuphead
  [myhead]
  [bodypartlabel=bodypartmyhead]

\setuplabeltext
  [en]
  [bodypartmyhead=My Head: ]

\myhead{Welcome}
```

**My Head: 3.4.1 Welcome**

The head text label class can be used as follows:

```latex
\setupheadtext
  [SomeHead=Just A Title]

\subsection
  [title=\headtext{SomeHead}]
```

**3.4.2 Just A Title**

A label will obey the style settings, as in:

```latex
\definehead
  [MyFancyHead]
  [subsection]
  [style={\bs\setcharactercasing[Words]}]
```

---

**Labels**
3.4.3 Just Another Title
4 Numbering

4.1 Introduction

Numbering is complex and in ConTeXt it’s not easy either. This is because we not only have 1, 2, 3 ... but also sub numbers like 1a, 1b, 1ic ... or 1.a, 1.b, 1.c ... There can be many levels, different separators, final symbols. As we’re talking languages we only discuss conversion here: the mechanism that turns a number in for instance a letter. It happens that the mapping from a number onto a letter is language dependent. The next lines show how English, Spanish and Slovenian numbers:

\begin{verbatim}
a b c d e f g h i j k l m n o p q r s t u v w x y z aa ab
a b c d e f g h i j k l m n ñ o p q r s t u v w x y z aa
a b c č d e f g h i j k l m n o p r s š t u v z ž aa ab ac
\end{verbatim}

You convert a number into a letter with:

\convertnumber{alphabetic}{15}

There is also \uconvertnumber which does not expand unless typesetting is going on. Normally you don’t need to bother about this.

The alphabetic converter adapts to the current main language. When a language has no special alphabet, the regular 26 characters are used.

A converter can also convert to a roman numeral, a language specific ordered list, a day or month, an ordinal string and again there can be a language specific conversion. The general conversion macro takes a conversion name and a number. When a conversion can be set (for instance in an itemized list, or in section numbering) you can use these names. You can define additional converters if needed, as long as the converter can handle a number.

\defineconversion [alphabetic] [\alphabeticnumerals]

Here \alphabeticnumerals is a converter. If you look into the source of ConTeXt you will see that many converters are calling out to Lua, where we have implemented those specific conversions. The following table has long and short names. The short one are historic.

<table>
<thead>
<tr>
<th></th>
<th>\monthlong</th>
<th>\monthshort</th>
</tr>
</thead>
<tbody>
<tr>
<td>month</td>
<td></td>
<td></td>
</tr>
<tr>
<td>month:mnem</td>
<td></td>
<td></td>
</tr>
<tr>
<td>character</td>
<td>\character</td>
<td>\Character</td>
</tr>
<tr>
<td>Character</td>
<td>\Character</td>
<td>\characters</td>
</tr>
<tr>
<td>characters</td>
<td>\characters</td>
<td>\Characters</td>
</tr>
<tr>
<td>Characters</td>
<td>\Characters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>\texttt{\textbackslash smallcappedcharacters}</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>AK</td>
<td>\texttt{\textbackslash smallcappedcharacters}</td>
<td></td>
</tr>
<tr>
<td>KA</td>
<td>\texttt{\textbackslash smallcappedcharacters}</td>
<td></td>
</tr>
<tr>
<td>alphabetic a</td>
<td>\texttt{\textbackslash alphabeticnumerals}</td>
<td></td>
</tr>
<tr>
<td>Alphabetic A</td>
<td>\texttt{\textbackslash alphabeticnumerals}</td>
<td></td>
</tr>
<tr>
<td>number numbers n</td>
<td>\texttt{\textbackslash numbers}</td>
<td></td>
</tr>
<tr>
<td>Numbers N</td>
<td>\texttt{\textbackslash Numbers}</td>
<td></td>
</tr>
<tr>
<td>mediaeval m</td>
<td>\texttt{\textbackslash mediaeval}</td>
<td></td>
</tr>
<tr>
<td>word words</td>
<td>\texttt{\textbackslash verbose\textbackslash number}</td>
<td></td>
</tr>
<tr>
<td>Word Words</td>
<td>\texttt{\textbackslash VerboseNumber}</td>
<td></td>
</tr>
<tr>
<td>ordinal</td>
<td>\texttt{\textbackslash ordinalnumber}</td>
<td></td>
</tr>
<tr>
<td>Ordinal</td>
<td>\texttt{\textbackslash Ordinalnumber}</td>
<td></td>
</tr>
<tr>
<td>roman\textbackslash numerals i r</td>
<td>\texttt{\textbackslash roman\textbackslash numerals}</td>
<td></td>
</tr>
<tr>
<td>Roman\textbackslash numerals I R</td>
<td>\texttt{\textbackslash Roman\textbackslash numerals}</td>
<td></td>
</tr>
<tr>
<td>o</td>
<td>\texttt{\textbackslash oldstyle\textbackslash numerals}</td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>\texttt{\textbackslash oldstyle\textbackslash numerals}</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td>\texttt{\textbackslash oldstyle\textbackslash roman\textbackslash numerals}</td>
<td></td>
</tr>
<tr>
<td>KR</td>
<td>\texttt{\textbackslash smallcapped\textbackslash roman\textbackslash numerals}</td>
<td></td>
</tr>
<tr>
<td>RK</td>
<td>\texttt{\textbackslash smallcapped\textbackslash roman\textbackslash numerals}</td>
<td></td>
</tr>
<tr>
<td>greek g</td>
<td>\texttt{\textbackslash greek\textbackslash numerals}</td>
<td></td>
</tr>
<tr>
<td>Greek G</td>
<td>\texttt{\textbackslash greek\textbackslash numerals}</td>
<td></td>
</tr>
<tr>
<td>mathgreek</td>
<td>\texttt{\textbackslash math\textbackslash greek}</td>
<td></td>
</tr>
<tr>
<td>abjad\textbackslash numerals</td>
<td>\texttt{\textbackslash abjad\textbackslash numerals}</td>
<td></td>
</tr>
<tr>
<td>abjad\textbackslash nodot\textbackslash numerals</td>
<td>\texttt{\textbackslash abjad\textbackslash nodot\textbackslash numerals}</td>
<td></td>
</tr>
<tr>
<td>abjad\textbackslash naive\textbackslash numerals</td>
<td>\texttt{\textbackslash abjad\textbackslash naive\textbackslash numerals}</td>
<td></td>
</tr>
<tr>
<td>thainumeral\s</td>
<td>\texttt{\textbackslash thainumeral\s}</td>
<td></td>
</tr>
<tr>
<td>devanagarinumerals</td>
<td>\texttt{\textbackslash devanagarinumerals}</td>
<td></td>
</tr>
<tr>
<td>gurmukhинumeral\s</td>
<td>\texttt{\textbackslash gurmukhинumeral\s}</td>
<td></td>
</tr>
<tr>
<td>gujaratинumeral\s</td>
<td>\texttt{\textbackslash gujaratинumeral\s}</td>
<td></td>
</tr>
<tr>
<td>tibetan\textbackslash numerals</td>
<td>\texttt{\textbackslash tibetan\textbackslash numerals}</td>
<td></td>
</tr>
<tr>
<td>greek\textbackslash numerals</td>
<td>\texttt{\textbackslash greek\textbackslash numerals}</td>
<td></td>
</tr>
<tr>
<td>Greek\textbackslash numerals</td>
<td>\texttt{\textbackslash Greek\textbackslash numerals}</td>
<td></td>
</tr>
<tr>
<td>arabic\textbackslash numerals</td>
<td>\texttt{\textbackslash arabic\textbackslash numerals}</td>
<td></td>
</tr>
<tr>
<td>persian\textbackslash numerals</td>
<td>\texttt{\textbackslash persian\textbackslash numerals}</td>
<td></td>
</tr>
<tr>
<td>arabic\textbackslash ex\textbackslash numerals</td>
<td>\texttt{\textbackslash arabic\textbackslash ex\textbackslash numerals}</td>
<td></td>
</tr>
<tr>
<td>arabic\textbackslash decimal\s</td>
<td>\texttt{\textbackslash arabic\textbackslash decimal\s}</td>
<td></td>
</tr>
<tr>
<td>persian\textbackslash decimal\s</td>
<td>\texttt{\textbackslash persian\textbackslash decimal\s}</td>
<td></td>
</tr>
<tr>
<td>korean\textbackslash numerals kr</td>
<td>\texttt{\textbackslash korean\textbackslash numerals kr}</td>
<td></td>
</tr>
<tr>
<td>korean\textbackslash parenthesis\textbackslash numerals kr-p</td>
<td>\texttt{\textbackslash korean\textbackslash parenthesis\textbackslash numerals kr-p}</td>
<td></td>
</tr>
<tr>
<td>korean\textbackslash circ\textbackslash numeral\s kr-c</td>
<td>\texttt{\textbackslash korean\textbackslash circ\textbackslash numeral\s kr-c}</td>
<td></td>
</tr>
</tbody>
</table>
The alphabetic and Alphabetic converters adapt to slovenian and spanish as do their small capped alternatives. There are more general helpers for it too:

\languagecharacters{number}
\languageCharacters{number}

Also language related is the \continuednumber macro. Here we see an application:

1 \continuednumber{1}
1, 2 \continuednumber{2}
1, 2, 3 \continuednumber{3}

What renders as:

1
1,2 (continued)
1,2,3 (continued)

Such a macro is typically used in combination with counters and it just typesets a label text depending on the value being non-zero.

\setuplabeltext[en][continued={and so on}]
1, 2, 3 \continuednumber{3}
1, 2, 3 \convertnumber{continued}{3}

This gives:

1,2,3 and so on
1,2,3 and so on

In the rare case that you want to check if a conversion is defined you can use

\doifelseconversiondefined{name}{true}{false}

So,

\doifelseconversiondefined{characters}{we can convert}{forget about it}
Gives:

\textbf{we can convert}

There are also some non language related converters that we mention here for completeness:

\begin{itemize}
\item \textbf{set 0:} • − ⋆ ▷ ∘ ○ • − ⋆ ▷ ∘ ○ • − ⋆ ▷ ∘ ○ • − ⋆ ▷ ∘ ○ • − ⋆ ▷ ∘ ○ • − ⋆ ▷ ∘ ○ • − ⋆ \textbf{set 1:} ⋆ ⋆⋆ ⋆ ⋆ ⋆ ‡ ‡‡ ‡ ‡ ‡ ∗ ∗∗ ∗ ∗ ∗ ⋆ ⋆⋆ ⋆ ⋆ ⋆ ‡ ‡‡ ‡ ‡ ‡ ∗ ∗∗ ∗ ∗ ∗ ⋆ ⋆⋆ ⋆ ⋆ ⋆ ‡ ‡‡ ‡ ‡ ‡ ∗ ∗∗ ∗ ∗ ∗ ⋆ ⋆⋆ \textbf{set 2:} ⋆ † ‡ ∗ ∗ ∗ † † † ‡‡ ∗ ∗ ∗ † † † † † ‡‡ ∗ ∗ ∗ † † † † † ‡‡ \textbf{set 3:} ⋆ ⋆⋆ ⋆ ⋆ ⋆ ‡ ‡‡ ‡ ‡ ‡ ¶ ¶¶ ¶¶¶ § §§ §§§ ∗ ∗∗ ∗ ∗ ∗ ⋆ ⋆⋆ ⋆ ⋆ ⋆ ‡ ‡‡ ‡ ‡ ‡ ∗ ∗∗ ∗ ∗ ∗ ⋆ ⋆⋆ \textbf{set 2:} ⋆ † ‡ ∗ ∗ ∗ † † † ‡‡ ∗ ∗ ∗ † † † † † ‡‡ ∗ ∗ ∗ † † † † † ‡‡ \textbf{set 3:} ⋆ ⋆⋆ ⋆ ⋆ ⋆ ‡ ‡‡ ‡ ‡ ‡ ¶ ¶¶ ¶¶¶ § §§ §§§ ∗ ∗∗ ∗ ∗ ∗ ⋆ ⋆⋆ ⋆ ⋆ ⋆ ‡ ‡‡ \end{itemize}

When a set overruns we start again at the first element.

The ordinal converter produces output like 123	extsuperscript{rd} and 654	extsuperscript{th}. The corresponding string renderer is \textbackslash highordinalstr.

\subsection*{4.2 Dates}

Dates are also language dependent. The following macros take a number and return the name of the month or day.

\begin{itemize}
\item \texttt{\%monthlong} October \texttt{\%monthshort} oct \texttt{\%MONTH} OCTOBER \texttt{\%MONTHLONG} OCTOBER \texttt{\%MONTHSHORT} OCT
\item \texttt{\%weekday} Thursday \texttt{\%WEEKDAY} THURSDAY
\end{itemize}

The current date can be typeset with \texttt{\%currentdate} and a specific date with \texttt{\%date}, for instance:

\begin{itemize}
\item \texttt{\%currentdate[weekday,day,month,year]}
\item \texttt{\%currentdate[WEEKDAY,day,MONTH,year]}
\item \texttt{\%date[d=12,m=12,y=1998][weekday]}
\item \texttt{\%date[d=12,m=12,y=1998]}
\end{itemize}

\begin{itemize}
\item Saturday 23 July 2016
\item SATURDAY 23 JULY 2016
\item Saturday
\item December 12, 1998
\end{itemize}

Possible elements of the specification are:
There are also some converters built in (more can be added), for instance:

The current \emph{gregorian} date \currentdate [month, day, \{,\}, year] is in \emph{jalali} \currentdate [jalali:to, month, day, \{,\}, year] but we can also as a specific one, so \emph{jalali} \date [y=1395, m=4, d=18] [month, day, \{,\}, year] is \emph{gregorian} \date [y=1395, m=4, d=18] [jalali:from, month, day, \{,\}, year].

The current \emph{gregorian} date July 23, 2016 is in \emph{jalali} May 2, 1395 but we can also as a specific one, so \emph{jalali} April 18, 1395 is \emph{gregorian} July 8, 2016.
5 Typesetting

5.1 Introduction

In this chapter we will discuss a few settings and mechanisms that deal with typesetting from the perspective of languages. We will not go into details about the often obscure demands that a language puts on a system like ConTeXt. Often these are rooted in tradition, limitations of past rendering (brushed, written, mechanical or electronic), subjective decisions made by committees, contradicting opinions of typographers, etc. The most we can do is provide the mechanism that make it possible to honour most of these demands and provide a reasonable set of defaults. It’s good to mention here that wasting energy on discussing language specific issues only makes sense when a similar amount of energy is spent on getting the rest of the document rendering right. It really makes no sense to whine about a lost (or bad) ligature, or a missed hyphenation point, or a loose paragraph when vertical spacing is sloppy, the use of color messy, the choice of fonts debatable, etc. The worst discussions I ran into are those involving inter-character spacing as way to optimize look, feel and readability while at the same time the choice of fonts and rest of the layout were not that attractive anyway.

5.2 Spacing

The look and feel of a paragraph is determined by several factors and language is undeniable one of them. Dutch and German have compound words that can be quite long, English and French use short words, some with only one character, Czech, Polish and many other languages have diacritics.

Interword spacing makes the text lighter, and the more short words there are, the more spacing shows up. Although, if the hyphenation patterns are suboptimal, spaces can stretch which can become annoying. Many uppercase characters (as in German) and accented characters make the text darker.

The user has not much influence on this, apart from rewriting the text. When a narrow column is used, that can be a bit of a challenge, as too narrow columns can just look bad. In the next example we see a sample text (tuftetex) typeset with the align options normal.

We thrive in information-thick worlds because of our marvelous and everyday capacity to select, edit, single out, structure, highlight, group, pair, merge, harmonize, synthesize, focus, organize, condense, reduce, boil down, choose, categorize, catalog, classify, list, abstract, scan, look into, idealize, isolate, discriminate, distinguish, screen, pigeonhole, pick over, sort, integrate, blend, inspect, filter, lump, skip, smooth, chunk, average, approximate, cluster, aggregate, outline, summarize, itemize, review, dip into, flip through, browse, glance into, leaf through, skim, refine, enumerate, glean, synopsize, winnow the wheat from the chaff and separate the sheep from the goats.
We see one word sticking into the margin, and there is not much that \TeX{} can do about it, given its constraints. We can be a bit more tolerant if we also add the option `tolerant`. This sample text is always good for lots of successive hyphenation I must admit that I never make a big deal about that if only because trying to avoid it often gives worse results.

We thrive in information–thick worlds because of our marvelous and everyday capacity to select, edit, single out, structure, highlight, group, pair, merge, harmonize, synthesize, focus, organize, condense, reduce, boil down, choose, categorize, catalog, classify, list, abstract, scan, look into, idealize, isolate, discriminate, distinguish, screen, pigeonhole, pick over, sort, integrate, blend, inspect, filter, lump, skip, smooth, chunk, average, approximate, cluster, aggregate, outline, summarize, itemize, review, dip into, flip through, browse, glance into, leaf through, skim, refine, enumerate, glean, synopsize, winnow the wheat from the chaff and separate the sheep from the goats.

Normally `normal`, `tolerant` is good enough for a document, but if you really want to play safe you can better also permit some stretch.

So, `normal`, `tolerant`, `stretch` or `normal`, `verytolerant`, `stretch` gives \TeX{} enough degrees of freedom to produce good results. When we typeset Dutch documents we always use that alignment setting.

Let’s simulate a language with words of an average length. We just randomize the length, permit some hyphenation.

If we assume more longer words we get:
and if we have less short words we get:

```
x x x X X
```

Even in the last case \TeX{} can still quite well make a good paragraph, thanks to the large number of possible breakpoints (each color is a unit within a word). If you look careful you will see that the amount of whitespace differs per line.

A space in a text stream becomes so called glue: a horizontal skip with optional stretch and/or shrink. The values come from the font or are derived from spacing related properties of the font. This value can be overloaded by setting the \texttt{\spaceskip} register. In the next line we show the font driven spacing of some different fonts:

```
x x x x X X
```

When we set \texttt{\spaceskip} to the 10pt we get:

```
x x x x X X
```

When we use the font related spacing typesetting the Zapf quote gives:

```
x x x x X X
```

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC’s tricks, and think that a widely-praised program, called up on the screen, will make everything automatic from now on.
but when we use a fixed \spaceskip of 10pt we get the following. This demonstrates that it really makes sense to have some stretch in the skip specification.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC’s tricks, and think that a widely-praised program, called up on the screen, will make everything automatic from now on.

The French add spaces before punctuation but no extra space after punctuation. Traditional \TeX can only handle additional spacing after characters at the end of a word. The term ‘frenchspacing’ is used for the feature that disables such extra spacing and nonfrenchspacing enables it. The extra space can differ per character and is specified by a factor. That factor (devided by 1000) is applied to the \xspaceskip. Spacing before such a final character is supported by Con\TeX but not a generic \TeX feature. Frenchspacing looks like:

```
```

Contrary to:

```
29! foo: 30!
```

The logic is as follows. When the \spaceskip is zero, then the spacing after a character is the one related to the font. Otherwise the \spaceskip is added. However, when a character has an $\mathsf{sfcode}$ other than 1000, the stretch and shrink component of that glue are multiplied by $\mathsf{sfcode}/1000$. When the $\mathsf{sfcode}$ exceeds 2000 the $\mathsf{xspaceskip}$ parameter is used. So, what frenchspacing actually does, is resetting those space related codes.

<table>
<thead>
<tr>
<th></th>
<th>broad</th>
<th>fixed</th>
<th>packed</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>3000</td>
<td>1000</td>
<td>1050</td>
</tr>
<tr>
<td>,</td>
<td>1250</td>
<td>1000</td>
<td>1050</td>
</tr>
<tr>
<td>?</td>
<td>3000</td>
<td>1000</td>
<td>1050</td>
</tr>
<tr>
<td>!</td>
<td>3000</td>
<td>1000</td>
<td>1050</td>
</tr>
<tr>
<td>:</td>
<td>2000</td>
<td>1000</td>
<td>1050</td>
</tr>
<tr>
<td>;</td>
<td>1500</td>
<td>1000</td>
<td>1050</td>
</tr>
</tbody>
</table>
The fixed variant is the french spacing as known for ages and discussed in the TeX book. You switch the model with:

```
\setupspacing[fixed] % french spacing
```

Additional models can be installed with \installspacingmethod and in the source code you can see how we did that for the above.

As mentioned, in TeX a space character U+32) is turned into glue. When you input some text with macros, you sometimes need to get a space into the stream. Take:

I am a \TeX user.

Here scanning the control sequence \TEX will gobble the space, so we need to do something:

I am a {\TeX} user, a happy \TeX user, yes, a proud \TeX{} user, forever a \TeX\space user!

All these cases will inject a space after the logo. The second solution, the \ originally was different because it introduces a space with an explicit \sfcode of 1000 (so a factor 1.0) but in ConTeXt we now just let that be a regular space in text mode and the original (primitive) meaning in math mode.

Another special case is ~ which is a nobreak space with a fixed width and in some cases (like in tables) a space width the same width as a digit.

```
| fixed  | xx xx | X | xx xx | X | xx dr. | X | xx dr. | X |
| broad  | xx xx | \X | xx xx | \X | xx dr. | \X | xx dr. | \X |
| fixed  | xx xx | X | xx xx | X | xx dr. | X | xx dr. | X |
| broad  | xx xx | X | xx xx | X | xx dr. | X | xx dr. | X |
| fixed  | xx xx X | xx xx | X | xx dr. X | xx dr. X |
| broad  | xx xx X | xx xx | X | xx dr. X | xx dr. X |
```

These subtle details probably seldom get noticed. For instance, most languages use the packed variant while English, Turkish and Arabic are configured to use broad. To some extend you can say that the European continent uses the same spacing setup. You can adapt spacing for a language with:

```
\setuplanguage[en][spacing=packed]
```

In addition to a regular space there are many other spacing directives but these always concern fixed width spaces. When possible these spaces travel through the system as Unicode characters which also means that you can use such characters directly.

| \nobreakspace \nbsp | U+00A0 | space |
| \ideographicisspace | U+2000 | quad/2 |
The \nospancing macro makes spaces disappear, not even a zero glue will be injected. Of course this macro will only be used grouped and in special cases.

5.3 Frequencies

Right from when Con\TeX{}t became multilingual there have been users submitting language specific settings for their language. Some changed over time which indicates that there can be different views, which is not that surprising because many ‘typographic rules’ are just formalizations of ‘this is the way we did it for ages’. I often wonder what rules came from limitations in the systems used to get things on paper: pens, pensils, letter by letter in wood or lead, line based printing, the size of paper, the reading direction, the quality of ink and paper, and so on. I’ve been present at debates about how high an accent should be placed on a character, depending on language, referring to some standard well known font that did it this or that way.

The dimensions of a page, the text area, the size and weight of characters, spacing, (the often neglectable or even debatable positive influence of) protrusion into the margin of certain characters, expansion of glyphs to give a better grayness … all this can lead to hefty discussions. They don’t make a bad looking design or text with all kind of textual elements (not all documents are novels) look better.

The optimal width of a text column is one of these properties can opinions can vary on. Long lines in a small font or short lines in a big one are often not pleasant to read and if you have to turn the page every ten lines you might loose track of the content. But what does it say to have for instance 65 characters on a line. This is quite language dependent. How do spaces count? Anyway, in Con\TeX{}t we have a variable that can help you decide what line length (text width) is acceptable. You can decide yourself what looks better:
The Earth, as a habitat for animal life, is in old age and has a fatal illness. Several, in fact. It would be happening whether humans had ever evolved or not. But our presence is like the effect of an old-age patient who smokes many packs of cigarettes per day — and we humans are the cigarettes.

The differences are not that large but at least you can play with it. The relevant helpers are:

\texttt{\textbackslash averagecharwidth}
These are used like:

\hsize=65\averagecharwidth
\hsize=65\languagecharwidth{nl}

or when a width is asked for

\setsomething[width=65\averagecharwidth]
\setsomething[width=65\languagecharwidth{de}]

Keep in mind that these are font dependent so you might want to use

\freezemeasure[MyWidth][65\averagecharwidth]

and then use \measure{MyWidth} when needed. The frequencies themselves are stored in tables like lang-freq-nl.lua:

```lua
return {
    language = "nl",
    source = "http://www.onzetaal.nl/advies/letterfreq.html",
    frequencies = {
        [0x61] = 7.47, [0x62] = 1.58, [0x63] = 1.24, [0x64] = 5.93, [0x65] = 18.91,
        [0x66] = 0.81, [0x67] = 3.40, [0x68] = 2.38, [0x69] = 6.50, [0x6A] = 1.46,
        [0x6B] = 2.25, [0x6C] = 3.57, [0x6D] = 2.21, [0x6E] = 10.03, [0x6F] = 6.06,
        [0x70] = 1.57, [0x71] = 0.009, [0x72] = 6.41, [0x73] = 3.73, [0x74] = 6.79,
        [0x75] = 1.99, [0x76] = 2.85, [0x77] = 1.52, [0x78] = 0.04, [0x79] = 0.035,
        [0x7A] = 1.39,
    }
}
```

As we only have a few such files, feel free to submit ones that suit your language.

You can show the frequencies in a nice table by loading a module (there is no need to have this in the core):

```
\usemodule[s-languages-frequencies]

\startcolumns[balance=yes] \en \showfrequencies \stopcolumns
\startcolumns[balance=yes] \de \showfrequencies \stopcolumns
\startcolumns[balance=yes] \nl \showfrequencies \stopcolumns
```

We get three tables:

<table>
<thead>
<tr>
<th>en</th>
<th>4.96529pt</th>
</tr>
</thead>
<tbody>
<tr>
<td>U+00061</td>
<td>a 8.040</td>
</tr>
<tr>
<td>U+00062</td>
<td>b 1.540</td>
</tr>
<tr>
<td>U+00063</td>
<td>c 3.060</td>
</tr>
<tr>
<td>U+00064</td>
<td>d 3.990</td>
</tr>
<tr>
<td>U+00065</td>
<td>e 12.510</td>
</tr>
<tr>
<td>U+00066</td>
<td>f 2.300</td>
</tr>
<tr>
<td>U+00067</td>
<td>g 1.960</td>
</tr>
<tr>
<td>U+00068</td>
<td>h 5.490</td>
</tr>
<tr>
<td>U+00069</td>
<td>i 7.260</td>
</tr>
</tbody>
</table>
The limited support in first generation text editors has made some of these cultural
aspects of typesetting disappear or at least it made users sloppy and a new generation forget about them. Quotes are an example and the default rendering on ascii keyboards hasn’t helped either. For instance nowadays the Dutch double quotation marks, let’s call them lower and upper nine quotes according to their shape, seems to have been replaced by upper double six quotes at the left and upper right double nine quotes on the right. Of course the real names are different.

\lowerleftsingleninequote \quotesinglebase
\lowerleftdoubleninequote \quotedblbase

In the language definition file (lang-def.mkiv) we use these names; they date from the MkII times:
In traditional \TeX fonts a ' ' and `` were implemented as ligatures but in Con\TeXt we never supported that (and we won't). In fact, even using explicit quotes is not advised. When you use \quotation, quote and friends the quotes will be used according the current main language.

Quotes are normally applied in quotations. The double quotes are used when we quote a person and single ones often when we refer to something. Each language does it different although computer usage manages to let most of us forget what is is the right tradition. Kids don’t seem to care much so in the end it might evolve into all of us using the (US) English traditions.

\usemodule[s-languages-frequencies]

\startcolumns[balance=yes] \en \showfrequencies \stopcolumns
\startcolumns[balance=yes] \de \showfrequencies \stopcolumns
\startcolumns[balance=yes] \nl \showfrequencies \stopcolumns

By default, we obey the current language:

\begin{tabular}{llll}
  \text{en: 4.96529pt} & \text{U+0006C} & \text{l} & \text{4.140} \\
  \text{U+00061} & \text{a} & \text{8.040} & \text{U+0006D} & \text{m} & \text{2.530} \\
  \text{U+00062} & \text{b} & \text{1.540} & \text{U+0006E} & \text{n} & \text{7.090} \\
  \text{U+00063} & \text{c} & \text{3.060} & \text{U+0006F} & \text{o} & \text{7.600} \\
  \text{U+00064} & \text{d} & \text{3.990} & \text{U+00070} & \text{p} & \text{2.000} \\
  \text{U+00065} & \text{e} & \text{12.510} & \text{U+00071} & \text{q} & \text{0.110} \\
  \text{U+00066} & \text{f} & \text{2.300} & \text{U+00072} & \text{r} & \text{6.120} \\
  \text{U+00067} & \text{g} & \text{1.960} & \text{U+00073} & \text{s} & \text{6.540} \\
  \text{U+00068} & \text{h} & \text{5.490} & \text{U+00074} & \text{t} & \text{9.250} \\
  \text{U+00069} & \text{i} & \text{7.260} & \text{U+00075} & \text{u} & \text{2.710} \\
  \text{U+0006A} & \text{j} & \text{0.160} & \text{U+00076} & \text{v} & \text{0.990} \\
  \text{U+0006B} & \text{k} & \text{0.670} & \text{U+00077} & \text{w} & \text{1.920} \\
\end{tabular}
But we can force the main language:

Instead of global one can set a known language, nothing or local. Anyway, global gives us:

### 5.5 Sentences

Another language specific property is sub-sentences (or asides). We just demonstrate some rendering here. Again this is configured in the language definition file.

```
 test \aside {test \aside {test} test} test |<|test test |>| test
```
For English, German, Dutch and French this looks as follows. As with quotations a nested instance can render differently.

```
test —test —test— test— test —test test— test
test - test - test - test - test - test - test - test
test —test —test— test —test test— test

test — test — test — test — test — test
```

5.6 Local control

Many ConTeXt commands can be controlled by the `align` parameter that accepts a list of directives. In addition to the justification directives the following ones are used to control the par builder.

<table>
<thead>
<tr>
<th>option</th>
<th>effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>tolerant</td>
<td>accept suboptimal hyphenation i.e. give less warnings</td>
</tr>
<tr>
<td>very tolerant</td>
<td>accept even less optimal hyphenation</td>
</tr>
<tr>
<td>stretch</td>
<td>permit stretch between words to satisfy the hyphenation demands</td>
</tr>
<tr>
<td>nothyphenated</td>
<td>don’t hyphenate at all</td>
</tr>
</tbody>
</table>

These directives are in fact just controllers for the following variables:

<table>
<thead>
<tr>
<th>variable</th>
<th>effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>\pretolerance</td>
<td>when a paragraph is not hyphenated (first pass) and the result stays within this tolerance \TeX doesn’t try further</td>
</tr>
<tr>
<td>\tolerance</td>
<td>when a paragraph is hyphenated (second pass) and the result stays within this tolerance \TeX doesn’t try further</td>
</tr>
<tr>
<td>\emergencystretch</td>
<td>permit in a third pass this extra stretch in a line before complaining</td>
</tr>
</tbody>
</table>

You need to keep in mind that the left and right hyphenmin variables (per language or locally) also influence the way a paragraph is broken into lines. If you want a paragraph to have more lines than \TeX want to give it, you can set the `\looseness` variable. Its value is forgotten when the paragraph is typeset so you don’t need to reset it yourself. This mechanism will only kick in when there are three passes and a large enough \emergencystretch is set.

The Earth, as a habitat for animal life, is in old age and has a fatal illness. Several, in fact. It would be happening whether humans had ever evolved or not. But our presence is like the effect of an old-age patient who smokes many packs of cigarettes per day — and we humans are the cigarettes.

The Earth, as a habitat for animal life, is in old age and has a fatal illness. Several, in fact. It would be happening whether humans had ever evolved or not. But our presence is like the effect of an old-age patient
who smokes many packs of cigarettes per day — and we humans are the cigarettes.

For this text the results looks quite bad. Personally I never use(d) this command.

There are two commands that can be used to increment or decrement the current values of \texttt{\lefthyphenmin} and \texttt{\righthyphenmin}:

\texttt{\lesshyphens}
\texttt{\morehyphens}

These can come in handy in for instance titles.
6 Goodies

6.1 Introduction

There are some features that will only be used in rare cases. They were often implemented as experiment but found useful enough to keep around.

6.2 Spell checking

There are some means to check the spelling of words in your document but get it right: ConTeXt is not a spell-checker. These features were added in order to be able to do some quick checking of documents written by multiple authors. There are currently three options and we only show a simple examples.

First you need to load word lists. These are either text files with just words separated by spacing.

foobard  foo-bar  foob=bar  foo{}{}{}bar  foo{}{}{}bar

All these words become foobar which means that one can use words with discretionary specifications. A text list is loaded with:

\loadspellchecklist[en][t:/manuals/lua/words-en.txt]

Instead you can load a Lua file with words. Here we use the same structure that we use for the spell checker provided for SciTE:

return {
    max   = 9,
    min   = 6,
    n     = 2,
    words = {
        ["barfoo"]  = "Barfoo"
        ["foobared"] = "foobared",
    }
}

We use the same load command (you can also load bytecode files with suffix luc this way):

\loadspellchecklist[nl][t:/scite/data/context/lexers/data/spell-nl.lua]

Usage boils down to enabling the checker. If needed we can add more methods. The first method colors the known and unknown colors. Words shorter then the threshold of 4 will be skipped.
Is this written right or is this wrong?

Is dit goed geschreven of niet?

You can change the colors:

\definecolor{word:yes}{g=.75}
\definecolor{word:no}{r=.75}

The second method doesn't show anything but produces a file `jobname.words` with used words. The found value of list is used as key in the produced table.

Is this written right or is this wrong?

Is dit goed geschreven of niet?

The produced table is:

```
return {
  "categories":{
    "found":{
      "languages":{
        "en":{
          "list":{
            "right":1,
            "this":2,
            "written":1,
            "wrong":1,
          },
          "parent":",
          "patterns":"us",
          "tag":"en",
          "total":5,
          "unique":4,
        },
        "nl":{
          "list":{
            "geschreven":1,
            "goed":1,
          },
        },
      },
    },
  },
}``
The result can be traced with a module:

\usemodule[s-languages-words]

\showwords

This shows up as:

**category: found, language: en, total: 5, unique: 4:** right (1) this (2) written (1) wrong (1)

**category: found, language: nl, total: 3, unique: 3:** geschreven (1) goed (1) niet (1)

The third mechanism colors languages differently. We only defined a few colors:

\definecolor[word:en] [b=.75]
\definecolor[word:de] [r=.75]
\definecolor[word:nl] [g=.75]
\definecolor[word:unknown][r=.75,g=.75]

but you can of course define a color for your favourite language in a similar way.

\setupspellchecking[state=start,method=3]
\en Is this written right or is this wrong?\par
\nl Is dit goed geschreven of niet?\par
\setupspellchecking[state=stop]

*Is this written right or is this wrong?*
*Is dit goed geschreven of niet?*
7 Sorting

7.1 Introduction

Sorting is complex, not so much for English, Dutch, German, etc. only texts but there are languages and scripts that are more demanding. There are several complications:

- There can be characters that have accents, like à, á, â, ã, ä . . . that have a base shape a and in an index these often end up close to each other. The order can differ per language.

- There are upper and lowercase words and there can be different expectations to them being mixed or separated.

- Some scripts have characters that are combinations, like Æ, and one might want to see them as one character or two, in which the second one obeys the sorting order. The shape can dominate here.

- Some scripts, like Japanese, are a combination of several scripts and sorting then depends on normalization.

- When there are many glyphs, like in Chinese, the order can depend on the complexity of the glyph and when we’re lucky that order is reflected in the numeric character order.

Often the rules are somewhat strict and one can doubt of the same rules would have been imposed if computers had been developed earlier. Given discussions one can doubt if the rules are really consistent or just there because someone (or a group) with influence set the standard (not so much different from grammar). So, if we deal with sorting, we do that in such a way that users can (to some extend) influence the outcome. After all, one important aspect of typesetting and organizing content is that the users gets the feeling of control and a diversion from a standard can be part of that. The reader will often not notice these details. In the next sections we will explore the way sorting is done in ConTeXt. The method evolved over a few decades. In MkII sorting happened between runs and it was just part of the processing of a document that users never really saw in action. Sorting just happened and few users will have noticed that we moved from a Modula program to a Perl script and ended up with a Ruby script. In fact, there is a Lua replacement but it never got tested well because we moved in to MkIV. There all happens inside the engine using Lua. Some principles stayed the same but we are more flexible now.

7.2 How it works

How does sorting work out? Take these words:
Sorting

abracadabra
abrãcadábra
àbracádabrä
ábracadâbra
äbrácadabrà

As long as they end up in an order where the reader can find it, we’re okay. After all we’re pretty good in pattern recognition.

There are probably many ways to implement a sorter but the one we use is more or less a follow up on the one we had for over a decade and was the result of an evolution based on user demand. It boils down to cleaning up the string in such a way that it can be split into meaningful characters. One can argue that we should use some kind of standardized sorting method but the problem is that we always have to deal with for instance embedded tex commands and mixed content, for instance numbers. And users using the same language can have different opinions about the rules too.

A word (or sequence of words) is split into characters. Because there can be \TeX\ commands in there some cleanup happens beforehand. After that we create several lists with numbers that will be compared when sorting two entries.

We can best demonstrate this with a few examples. As usual an English language example is trivial.

<table>
<thead>
<tr>
<th>en</th>
<th>abracadabra</th>
</tr>
</thead>
<tbody>
<tr>
<td>ch</td>
<td>a b r a c a d a b r a</td>
</tr>
<tr>
<td>uc</td>
<td>x61 x62 x72 x61 x63 x61 x64 x61 x62 x72 x61</td>
</tr>
<tr>
<td>zc</td>
<td>x61 x62 x72 x61 x63 x61 x64 x61 x62 x72 x61</td>
</tr>
<tr>
<td>mc</td>
<td>x61 x62 x72 x61 x63 x61 x64 x61 x62 x72 x61</td>
</tr>
<tr>
<td>pc</td>
<td>x61 x62 x72 x61 x63 x61 x64 x61 x62 x72 x61</td>
</tr>
<tr>
<td>zm</td>
<td>x02 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02</td>
</tr>
<tr>
<td>mm</td>
<td>x02 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02</td>
</tr>
<tr>
<td>pm</td>
<td>x02 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02</td>
</tr>
</tbody>
</table>

When we add an uppercase character we get a slightly different outcome:

<table>
<thead>
<tr>
<th>en</th>
<th>Abracadabra</th>
</tr>
</thead>
<tbody>
<tr>
<td>ch</td>
<td>A b r a c a d a b r a</td>
</tr>
<tr>
<td>uc</td>
<td>x41 x62 x72 x61 x63 x61 x64 x61 x62 x72 x61</td>
</tr>
<tr>
<td>zc</td>
<td>x61 x62 x72 x61 x63 x61 x64 x61 x62 x72 x61</td>
</tr>
<tr>
<td>mc</td>
<td>x60 x62 x72 x61 x63 x61 x64 x61 x62 x72 x61</td>
</tr>
<tr>
<td>pc</td>
<td>x62 x62 x72 x61 x63 x61 x64 x61 x62 x72 x61</td>
</tr>
<tr>
<td>zm</td>
<td>x02 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02</td>
</tr>
<tr>
<td>mm</td>
<td>x01 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02</td>
</tr>
<tr>
<td>pm</td>
<td>x03 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02</td>
</tr>
</tbody>
</table>
Some characters will be split, like æ:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>en</td>
<td>æsop</td>
</tr>
<tr>
<td>ch</td>
<td>Æsop</td>
</tr>
<tr>
<td>uc</td>
<td>xE6 x73 x6F x70</td>
</tr>
<tr>
<td>zc</td>
<td>xE6 x73 x6F x70</td>
</tr>
<tr>
<td>mc</td>
<td>xE6 x73 x6F x70</td>
</tr>
<tr>
<td>pc</td>
<td>xE6 x73 x6F x70</td>
</tr>
<tr>
<td>zm</td>
<td>x02 x0A x26 x1E x20</td>
</tr>
<tr>
<td>mm</td>
<td>x02 x0A x26 x1E x20</td>
</tr>
<tr>
<td>pm</td>
<td>x02 x0A x26 x1E x20</td>
</tr>
</tbody>
</table>

It gets more complex when language specific demands kick in. Compare an English, German and Austrian split:

<table>
<thead>
<tr>
<th></th>
<th>Abräcàdábra</th>
</tr>
</thead>
<tbody>
<tr>
<td>en</td>
<td>A b r ä c à d á b r a</td>
</tr>
<tr>
<td>ch</td>
<td>x41 x62 x72 x61 x65 x63 xE0 x64 xE1 x62 x72 x61</td>
</tr>
<tr>
<td>uc</td>
<td>x41 x62 x72 x61 x65 x63 xE0 x64 xE1 x62 x72 x61</td>
</tr>
<tr>
<td>zc</td>
<td>x41 x62 x72 x61 x65 x63 xE0 x64 xE1 x62 x72 x61</td>
</tr>
<tr>
<td>mc</td>
<td>x41 x62 x72 x61 x65 x63 xE0 x64 xE1 x62 x72 x61</td>
</tr>
<tr>
<td>pc</td>
<td>x41 x62 x72 x61 x65 x63 xE0 x64 xE1 x62 x72 x61</td>
</tr>
<tr>
<td>zm</td>
<td>x02 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02</td>
</tr>
<tr>
<td>mm</td>
<td>x02 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02</td>
</tr>
<tr>
<td>pm</td>
<td>x02 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Abräcàdábra</th>
</tr>
</thead>
<tbody>
<tr>
<td>de</td>
<td>A b r a e c à d á b r a</td>
</tr>
<tr>
<td>ch</td>
<td>x41 x62 x72 x61 x65 x63 xE0 x64 xE1 x62 x72 x61</td>
</tr>
<tr>
<td>uc</td>
<td>x41 x62 x72 x61 x65 x63 xE0 x64 xE1 x62 x72 x61</td>
</tr>
<tr>
<td>zc</td>
<td>x41 x62 x72 x61 x65 x63 xE0 x64 xE1 x62 x72 x61</td>
</tr>
<tr>
<td>mc</td>
<td>x41 x62 x72 x61 x65 x63 xE0 x64 xE1 x62 x72 x61</td>
</tr>
<tr>
<td>pc</td>
<td>x41 x62 x72 x61 x65 x63 xE0 x64 xE1 x62 x72 x61</td>
</tr>
<tr>
<td>zm</td>
<td>x02 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02</td>
</tr>
<tr>
<td>mm</td>
<td>x02 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02</td>
</tr>
<tr>
<td>pm</td>
<td>x02 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Abräcàdábra</th>
</tr>
</thead>
<tbody>
<tr>
<td>de-at</td>
<td>A b r ä c à d á b r a</td>
</tr>
<tr>
<td>ch</td>
<td>x41 x62 x72 x61 x65 x63 xE0 x64 xE1 x62 x72 x61</td>
</tr>
<tr>
<td>uc</td>
<td>x41 x62 x72 x61 x65 x63 xE0 x64 xE1 x62 x72 x61</td>
</tr>
<tr>
<td>zc</td>
<td>x41 x62 x72 x61 x65 x63 xE0 x64 xE1 x62 x72 x61</td>
</tr>
<tr>
<td>mc</td>
<td>x41 x62 x72 x61 x65 x63 xE0 x64 xE1 x62 x72 x61</td>
</tr>
<tr>
<td>pc</td>
<td>x41 x62 x72 x61 x65 x63 xE0 x64 xE1 x62 x72 x61</td>
</tr>
<tr>
<td>zm</td>
<td>x02 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02</td>
</tr>
</tbody>
</table>
The way a character gets replaced, like ä into ae, is defined in sort-lan.lua using Lua tables. We will not explain all the obscure details here; most of the work is already done, so users are not bothered by these definitions. And new ones can often be made by copying and adapting an existing one.

The sorting itself is specified by a sequence:

```
default   zc, pc, zm, pm, uc
before     mm, mc, uc
after      pm, mc, uc
first      pc, mm, uc
last       mc, mm, uc
```

The raw character is what we get after the (language specific) replacement has been applied and the unicodes are used when comparing. Lowercasing is done using the Unicode lowercase code, but one can define language specific ones too. The plus and minus variants can be used to force lowercase before or after uppercase. The mapping is based on an alphabet specification so this can differ per language and again we also provide plus and minus values that depend on case. When a character has no case we use shapes instead. For instance, the shape of à is a. Digits are treated special and currently get an offset so that they end up last in the sort order.

This three entry index should be sorted in the order: ぱー ぱぁ ぱあ.
<table>
<thead>
<tr>
<th>zm  zero mapping</th>
<th>x34 x02</th>
</tr>
</thead>
<tbody>
<tr>
<td>mm  minus mapping</td>
<td>x34 x02</td>
</tr>
<tr>
<td>pm  plus mapping</td>
<td>x34 x02</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>jp</th>
<th>ばー</th>
</tr>
</thead>
<tbody>
<tr>
<td>ch raw character</td>
<td>ばー</td>
</tr>
<tr>
<td>uc unicode</td>
<td>x3071 x30FC</td>
</tr>
<tr>
<td>zc  lowercase</td>
<td>x3071 x30FC</td>
</tr>
<tr>
<td>mc  lowercase - 1</td>
<td>x3071 x30FC</td>
</tr>
<tr>
<td>pc  lowercase + 1</td>
<td>x3071 x30FC</td>
</tr>
<tr>
<td>zm  zero mapping</td>
<td>x34 x315C</td>
</tr>
<tr>
<td>mm  minus mapping</td>
<td>x34 x315C</td>
</tr>
<tr>
<td>pm  plus mapping</td>
<td>x34 x315C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>jp</th>
<th>ばぁ</th>
</tr>
</thead>
<tbody>
<tr>
<td>ch raw character</td>
<td>ばぁ</td>
</tr>
<tr>
<td>uc unicode</td>
<td>x3071 x3042</td>
</tr>
<tr>
<td>zc  lowercase</td>
<td>x3071 x3042</td>
</tr>
<tr>
<td>mc  lowercase - 1</td>
<td>x3071 x3042</td>
</tr>
<tr>
<td>pc  lowercase + 1</td>
<td>x3071 x3042</td>
</tr>
<tr>
<td>zm  zero mapping</td>
<td>x34 x02</td>
</tr>
<tr>
<td>mm  minus mapping</td>
<td>x34 x02</td>
</tr>
<tr>
<td>pm  plus mapping</td>
<td>x34 x02</td>
</tr>
</tbody>
</table>

*To be continued!*
Appendix

A.1 The language files
Todo.

A.2 The mtx-patterns script
Todo.

A.3 Installed sorters
\usemodule[s-languages-sorting]
\showinstalledsorting

<table>
<thead>
<tr>
<th>language</th>
<th>DIN 5007-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent</td>
<td>default</td>
</tr>
<tr>
<td>method</td>
<td>mm,mc,uc</td>
</tr>
<tr>
<td>replacements</td>
<td>none</td>
</tr>
<tr>
<td>order</td>
<td>a b c d e f g h i j k l m n o p q r s t u v w x y z</td>
</tr>
<tr>
<td>entries</td>
<td>a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>language</th>
<th>DIN 5007-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent</td>
<td>default</td>
</tr>
<tr>
<td>method</td>
<td>mm,mc,uc</td>
</tr>
<tr>
<td>replacements</td>
<td>ä=ae Ä= Ae ö=oe Ö=Oe ü=ue Ü=Ue</td>
</tr>
<tr>
<td>order</td>
<td>a b c d e f g h i j k l m n o p q r s t u v w x y z</td>
</tr>
<tr>
<td>entries</td>
<td>a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>language</th>
<th>Duden</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent</td>
<td>default</td>
</tr>
<tr>
<td>method</td>
<td>mm,mc,uc</td>
</tr>
<tr>
<td>replacements</td>
<td>ß=s</td>
</tr>
<tr>
<td>order</td>
<td>a b c d e f g h i j k l m n o p q r s t u v w x y z</td>
</tr>
<tr>
<td>entries</td>
<td>a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>language</th>
<th>be</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent</td>
<td>default</td>
</tr>
<tr>
<td>Method</td>
<td>Replacements</td>
</tr>
<tr>
<td>--------</td>
<td>--------------</td>
</tr>
</tbody>
</table>
| mm,mc,uc | none | а б в г д е ё ж з и й к л м н о п р с т у ў ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ њ ѣ ѫ ј љ ћ �

Appendix
Appendix

<table>
<thead>
<tr>
<th>language</th>
<th>parent</th>
<th>method</th>
<th>replacements</th>
<th>order</th>
<th>entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>da</td>
<td>no</td>
<td>mm,mc,uc</td>
<td>none</td>
<td>abcd efghijklmnopqrstuvwxyz</td>
<td>a=a b=c c d=d e=e f=fg ghhi j=jk kl=lm m n o=0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>p=p q=q r=s s t=t u=u v=vw xyz</td>
</tr>
<tr>
<td>de</td>
<td>default</td>
<td>mm,mc,uc</td>
<td>none</td>
<td>abcd efghijklmnopqrstuvwxyz</td>
<td>a=a b=c c d=d e=e f=fg ghhi j=jk kl=lm m n o=0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>p=p q=q r=r s=s t=t u=u v=vw x=x y=y z=z</td>
</tr>
<tr>
<td>de-AT</td>
<td>default</td>
<td>mm,mc,uc</td>
<td>none</td>
<td>abcd efghijklmn opqrstuvwxyz</td>
<td>a=a b=c c d=d e=e f=fg ghhi j=jk kl=lm m n o=0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>p=p q=q r=r s=s t=t u=u v=vw x=x y=z</td>
</tr>
<tr>
<td>de-CH</td>
<td>de</td>
<td>mm,mc,uc</td>
<td>none</td>
<td>abcd efghijklmnopqrstuvwxyz</td>
<td>a=a b=c c d=d e=e f=fg ghhi j=jk kl=lm m n o=0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>p=p q=q r=r s=s t=t u=u v=vw x=x y=z</td>
</tr>
<tr>
<td>de-DE</td>
<td>de</td>
<td>mm,mc,uc</td>
<td>none</td>
<td>abcd efghijklmnopqrstuvwxyz</td>
<td>a=a b=c c d=d e=e f=fg ghhi j=jk kl=lm m n o=0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>p=p q=q r=r s=s t=t u=u v=vw x=x y=z</td>
</tr>
<tr>
<td>default</td>
<td>default</td>
<td>mm,mc,uc</td>
<td>none</td>
<td>abcd efghijklmnopqrstuvwxyz</td>
<td>a=a b=c c d=d e=e f=fg ghhi j=jk kl=lm m n o=0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>p=p q=q r=r s=s t=t u=u v=vw x=x y=z</td>
</tr>
</tbody>
</table>
method mm,mc,uc
replacements none
order a b c d e f g h i j k l m n o p q r s t u v w x y z
entries a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z

language deo
parent de
method mm,mc,uc
replacements ä=ae Ä=Ae ö=oe Ö=Oe ü=ue Ü=Ue ß=s
order a b c d e f g h i j k l m n o p q r s t u v w x y z
entries a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z

language en
parent default
method mm,mc,uc
replacements none
order a á b c d e é f g h í j k l m ñ o ó p q r s t û ú ü v w x y z
entries a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z á=a é=e í=i ñ=ñ ó=o û=u ù=u

language es
parent default
method mm,mc,uc
replacements none
order a á b c d e é f g h í j k l m ñ o ó p q r s t ú ü v w x y z
entries a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z á=a é=e í=i ñ=ñ ó=o ú=u û=u

language et
parent default
method mm,mc,uc
replacements none
order a b d e f g h i j k l m n o p r s š z ž t u v w õ ö ü x y
entries a=a b=b d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p r=r s=s t=t u=u v=v w=v x=x y=y z=z ä=ä ö=ö ö=o ü=ü š=š ž=ž

language fi
parent default
method mm,mc,uc
replacements none
order a b c d e f g h i j k l m n o p r s t u v w x y z å à ö
entries a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z ä=ä å=å ö=ö

Appendix
Appendix

<table>
<thead>
<tr>
<th>language</th>
<th>parent</th>
<th>method</th>
<th>replacements</th>
<th>order</th>
<th>entries</th>
</tr>
</thead>
</table>
| fr       | default| mm,mc,uc| none         | abcde| a=b=c=d=e=f=g|h=i=j=k=l|m=m
|          |        |        |              | p=q  | r=s=t=| u=v=w|x=y|z=z |
| gr       | default| mm,mc,uc| none         | αβγδε| ηθιζ  |                 |       |     |
|          |        |        |              |      |       |     |
| hr       | default| mm,mc,uc| dž=DŽ=lj=LJ| abcčđ|                              |       |     |
|          |        |        | dz=DZ=  |      |       |     |
| hu       | default| mm,mc,uc| cs=DZS= | abcčđ|                              |       |     |
Appendix

### Appendix

- **entries**
  - a=a b=b c=c d=d e=e f=f g=g h=h i=i j=k l=k m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z á=a é=e í=i ò=o õ=õ ú=û ü=ü

- **language**
  - is

- **parent**
  - default

- **method**
  - mm,mc,uc

- **replacements**
  - none

- **entries**
  - a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y á=a é=e í=i ò=o õ=õ ú=û ü=ü

- **language**
  - it

- **parent**
  - default

- **method**
  - mm,mc,uc

- **replacements**
  - none

- **entries**
  - a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y á=a é=e í=i ò=o õ=õ ú=û ü=ü

- **language**
  - jp

- **parent**
  - default

- **method**
  - zm

- **replacements**
  - 0x03041=0x03042 0x03043=0x03044 0x03045=0x03046 0x03047=0x03048 0x03049=0x0304A 0x0304B=0x0304D 0x0304F=0x03051 0x03053=0x03055 0x03057=0x03059 0x0305B=0x03061 0x03064=0x03066 0x03068=0x0306A 0x0306B=0x0306C 0x0306D=0x0306E 0x0306F 0x03072=0x03075 0x03078=0x0307B 0x0307E 0x03080=0x03081 0x03082=0x03084 0x03086 0x03089=0x0308A 0x0308B=0x0308C 0x0308D=0x0308F 0x03090=0x03091 0x03092=0x03093

- **entries**
  - 0x03042=0x03043=0x03044=0x03045=0x03046=0x03047=0x03048 0x03049=0x0304A 0x0304B=0x0304D 0x0304F=0x03051 0x03053=0x03055 0x03057=0x03059 0x0305B=0x03061 0x03064=0x03066 0x03068=0x0306A 0x0306B=0x0306C 0x0306D=0x0306E 0x0306F 0x03072=0x03075 0x03078=0x0307B 0x0307E 0x03080=0x03081 0x03082=0x03084 0x03086 0x03089=0x0308A 0x0308B=0x0308C 0x0308D=0x0308F 0x03090=0x03091 0x03092=0x03093

- **language**
  - is

- **parent**
  - default

- **method**
  - mm,mc,uc

- **replacements**
  - none

- **entries**
  - a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z á=a é=e í=i ò=o õ=õ ú=û ü=ü

- **language**
  - it

- **parent**
  - default

- **method**
  - mm,mc,uc

- **replacements**
  - none

- **entries**
  - a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z á=a é=e í=i ò=o õ=õ ú=û ü=ü
language kr
parent default
method mm,mc,uc
replacements none
order 0x03131 0x03134 0x03137 0x03139 0x03141 0x03142 0x03145
d e f g h i j k l m n o 0 p q r s t u v w x y z
entries a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o
p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z

language la
parent default
method mm,mc,uc
replacements æ=ae Æ=AE
order aāăbcdeēĕfghiīĭjklmnoōŏpqrstuūuvwx yŷź
entries a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o
p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y ų=ŷ=z=ź=ā=ā=â=â=e=e
ê=e ī=i ĭ=i ō=ō ū=ū ŭ=u ŭ=ŷ=y

language lt
parent default
method mm,mc,uc
replacements ch=0x10001 CH=0x1000B
order aąabcčdeēęfghįįjklmnoōŏprrsštuvzž
entries a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o
p=p q=q r=r s=s t=t u=u v=v y=y i=i z=z a=a ć=ć ė=ė ź=ź
ź=ź 0x10001=c

language lv
parent default
method mm,mc,uc
replacements none
order aāabcędēęfęgęhįįjįįkįįlļmnnőōoprṛrsstuvzž
entries a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o
p=p q=q r=r s=s t=t u=u v=v z=z ā=ā ē=ē ź=ź ź=ź

language nl
parent default
method mm,mc,uc
replacements ij=y IJ=Y
order a b c d e f g h i j k l m n o p q r s t u v w x y z
entries a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z
language no
parent default
method mm,mc,uc
replacements none
order a b c d e f g h i j k l m n o p q r s t u v w x y z æ ø å
entries a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z å=å æ=æ ø=ø
language ocs-scn
parent default
method mm,mc,uc
replacements ou=0x10001 OU=0x10015 g’=0x10002 G’=0x10016 št=0x10003 ŠT=0x10017 ju=0x10004 JU=0x10018 ja=0x10005 JA=0x10019 je=0x10006 JE=0x1001A je=0x10007 JE=0x1001B jo=0x10008 JØ=0x1001C ks=0x10009 KS=0x1001D ps=0x1000A PS=0x1001E th=0x1000B TH=0x1001F šć=0x1000C ŠĆ=0x10020
order a b v g d e ż 3 z i ů 0x10002 k l m n o p r s t u f x o c č š 0x10003
0x1000C ź y 0x10001 ź ė 0x10004 0x10005 0x10006 ė 0x10007 ő
0x10008 0x10009 0x1000A 0x1000B ü
entries a=a b=b c=c d=d e=e f=f g=g i=i k=k l=l m=m n=n o=o p=p r=r s=s t=t u=u v=v x=x y=y z=z į=i ů=ū ĉ=č ĕ=ę Ė=Ě Ž=Ž Ž=Ž q=ő ź=ź Ž=Ž š=š ź=ź
language pl
parent default
method mm,mc,uc
replacements none
order a b c ć d e e ę f g h i j k l l m n o ó p q r s ś t u v w x y z ź ź
entries a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z ó=ó a=a ć=ć ę=ę 1=1 ź=ź ź=ź ź=ź
language pt
parent default
method mm,mc,uc
replacements none
Appendix

<table>
<thead>
<tr>
<th>Order</th>
<th>Entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>a á â å b c ç d e é ê e f g h i í j k l m n ó õ ö o p q r s t u ú ü v w x y z</td>
<td>a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z à=â=a â=a ã=ã ç=c é=e è=e ë=e í=i ó=o ô=o õ=o u=u ü=u</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Language</th>
<th>Parent</th>
<th>Method</th>
<th>Replacements</th>
<th>Order</th>
<th>Entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>ro</td>
<td>default</td>
<td>mm,mc,uc</td>
<td>none</td>
<td>a á â å b c ç d e é ê e f g h i í j k l m n ó õ ö o p q r s t u ú ü v w x y z</td>
<td>a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z à=â=a â=a ã=ã ç=c é=e è=e ë=e í=i ó=o ô=o õ=o u=u ü=u</td>
</tr>
<tr>
<td>ru</td>
<td>default</td>
<td>mm,mc,uc</td>
<td>none</td>
<td>a á â å b c ç d e é ê e f g h i í j k l m n ó õ ö o p q r s t u ú ü v w x y z</td>
<td>a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z à=â=a â=a ã=ã ç=c é=e è=e ë=e í=i ó=o ô=o õ=o u=u ü=u</td>
</tr>
<tr>
<td>ru-iso9</td>
<td>default</td>
<td>mm,mc,uc</td>
<td>&quot;=0x10001</td>
<td>abvgdeëžziìjklmnoprstufhcčšʺ</td>
<td>a=a b=b v=v g=g d=d e=e j=j k=k l=l m=m n=n o=o p=p r=r s=s t=t u=u v=v y=y z=z à=â a â=à ã=ã ç=ç è=è ë=ë i=i ó=ô õ=õ u=û v=v</td>
</tr>
<tr>
<td>sk</td>
<td>default</td>
<td>mm,mc,uc</td>
<td>dz=0x10001</td>
<td>a á ab c cc d dd e e f f g g h h i=i j j k k l l m m n n o o p p q q r rs st uu v v w w x x y y z z ž ž ’ ’ ” &quot; 0x10001=“</td>
<td>a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z á=â a â=à ã=ã ç=ç è=è ë=ë i=i ó=ô õ=õ u=û v=v</td>
</tr>
<tr>
<td>sl</td>
<td>default</td>
<td>mm,mc,uc</td>
<td>0x10002=ch</td>
<td>a á ab c cc d dd e e f f g g h h i=i j j k k l l m m n n o o p p q=q r r s=s t=t u=u v=v w=w x=x y=y z=z á=â a â=à ã=ã ç=ç è=è ë=ë i=i ó=ô õ=õ u=û ý=y č=ć d=d ř=ř l=l ě=ě ý=ý ž=ž 0x10001=cz 0x10002=dž 0x10003=ch</td>
<td></td>
</tr>
<tr>
<td>en</td>
<td>es</td>
<td>number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>--------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zero</td>
<td>uno</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>one</td>
<td>dos</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>two</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A.4 Verbose counters

\usemodule{s-languages-counters}

\showverbosecounters[language={en,es}]
three  tres  3
four  cuatro  4
five  cinco   5
six   seis    6
seven siete  7
eight ocho   8
nine  nueve  9
ten  diez    10
eleven once  11
twelve doce  12
thirteen trece 13
fourteen catorce 14
fifteen quince 15
sixteen dieciséis 16
seventeen dieciséis 17
eighteen dieciocho 18
nineteen diecinueve 19
twenty veinte 20
veintiuno 21
veintidós 22
veintitrés 23
veinticuatro 24
veinticinco 25
veintiséis 26
veintisiete 27
veintiocho 28
veintinueve 29
thirty treinta 30
forty cuarenta 40
fifty cincuenta 50
sixty sesenta 60
seventy setenta 70
eighty ochenta 80
ninety noventa 90
hundred ciento 100
doscientos 200
trescientos 300
cuatrocientos 400
quinientos 500
seiscientos 600
setecientos 700
ochocientos 800
novecientos 900
thousand mil 1000
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>million</td>
<td>millón</td>
<td>1000000</td>
</tr>
<tr>
<td>billion</td>
<td>mil millones</td>
<td>1000000000</td>
</tr>
<tr>
<td>trillion</td>
<td>billón</td>
<td>1000000000000</td>
</tr>
</tbody>
</table>
This book explains how we support languages (and scripts) in ConTEXT MKIV and LuaTEX. Some of the mechanisms discussed are generic and not ConTEXT specific. We discuss the way languages are dealt with in the engine, hyphenation, standard features and additional goodies. Tracing and the extensibility of code are also discussed.