

12r

r21

a few tips

hans hagen

1 Introduction

With `CONTEXT` you can typeset in two directions: from left to right and from right to left. In fact you can also combine these two directions, like this:

There are many `{\righttoleft \maincolor \it scripts in use}` and some run into the other direction. However, there is `{\righttoleft \maincolor \it no fixed relation {\lefttoright \black \it between the}` direction of the script} and cars being driven left or right of the road.

There are many *esu ni stpircs* and some run into the other direction. However, there is *dexfi on tpircs eht fo noitcerid* between the *noitaler* and cars being driven left or right of the road.

Even someone not familiar with right to left typesetting can see what happens here, or not? In fact Luigi Scarso pointed out that the `fixed` reversed into `dexif` but not in the example where **fixed** becomes **dexfi**. This signals an important property of the way the text gets processed: you input something, at some points font features get applied (like ligatures) and in the end the resulting glyph stream is reversed. By that time the combination of **f+i** has become **fi**! So, be prepared for surprises.

This manual is written by a left to right user so don't expect a manual on semitic typesetting. Also don't expect a (yet) complete manual. I'll add whatever comes to mind. This is not a manual about Hebrew or Arabic, if only because I can't read any of those scripts (languages). I leave that to others to cover.

This is work in progress! So expect errors and typos.

Hans Hagen
Hasselt, NL

2 Setting up fonts

So let's see how Arabic and Hebrew come out:

The sentence \code{\quotation {I have no clue what this means.}} is translated (by Google Translate) into \code{\ar \rightrightleft ل ا م ع ة ر ك ف ي ا ي د ل س ي ل ا ذ ه .}

which is then translated back to \code{\quotation {I have no idea what this means.}} so maybe arabic has no clue what a clue is. The suggested Arabic pronunciation is \code{\quotation {lays laday 'ayu fikrat eamaa yaenih hadha}}. Hebrew also likes ideas more: \code{\quotation {\he \rightrightleft א י א ג ש ו מ י ל ן י א}}.

The sentence “I have no clue what this means.” is translated (by Google Translate) into “ليس لدي أي فكرة” which is then translated back to “I have no idea what this means.” so maybe arabic has no clue what a clue is. The suggested Arabic pronunciation is “lays laday 'ayu fikrat eamaa yaenih hadha”. Hebrew also likes ideas more: “אין לי מושג מזה זה אומר”.

According to Idris Hamid the Arabic should actually be this: “لَيْسَ لَدَيْ أَيِّ فِكْرَةٍ عَمَّا يَعْنِيهِ هَذَا” and the transliteration “Laysa ladayya ayyu fikratin ‘ammā ya‘nihi hādḥā”.

The CONTEXT (or any T_EX) ecosystem deals with languages and fonts. Languages (that relate to scripts) have specific characteristics, like running from right to left, and fonts provide a repertoire of glyphs and features. There is no real (standard) relationship between these. In for instance browsers, there are automatic fallback systems for missing characters in a font: another font is taken. These fallbacks are often not easy to tweak.

In this document we use DejaVu and although that font has Arabic shapes in its monospace variant, the serifs come without them (at least when I write this down). Before we actually define the bodyfont we hook in some fallbacks. The typescript for DejaVu has lines like this:

```
\definefontsynonym
  [SerifBoldItalic]
  [name:dejavuserifbolditalic]
  [features=default,
   fallbacks=SerifBoldItalic]
```

This permits us to do this:

```
\definefontfallback
  [Serif] [scheherazaderegular*arabic]
  [arabic] [force=yes,rscale=1.5]
\definefontfallback
  [SerifBold] [scheherazadebold*arabic]
  [arabic] [force=yes,rscale=1.5]
\definefontfallback
  [SerifItalic] [scheherazaderegular*arabic]
  [arabic] [force=yes,rscale=1.5]
\definefontfallback
  [SerifBoldItalic] [scheherazadebold*arabic]
```

```

[arabic] [force=yes,rscale=1.5]

\definefontfallback
  [Serif] [sileot*hebrew]
  [hebrew] [force=yes]
\definefontfallback
  [SerifBold] [sileot*hebrew]
  [hebrew] [force=yes]
\definefontfallback
  [SerifItalic] [sileot*hebrew]
  [hebrew] [force=yes]
\definefontfallback
  [SerifBoldItalic] [sileot*hebrew]
  [hebrew] [force=yes]

\definefontfallback
  [Mono] [almfixed*none]
  [arabic] [force=yes]

\definefontfallback
  [Mono] [sileot*none]
  [hebrew] [force=yes,factor=1] % factor forces a monospace

\setupbodyfont
  [dejavu,10pt]

```

In addition we set up the languages:

```

\setuplanguage[ar][font=arabic,bidi=right]
\setuplanguage[he][font=hebrew,bidi=right]

```

The following example demonstrates what the effects of these commands are:

```

{.|ז'א'ה ,ה'י'נ'ע'י'א'ס'מ'ע' ,ר'ק'פ' ,י'א'י'ד'ל'ש'י'ל' }
{.רמוא הז המ גשומ יל ןיא}
{\righttoleft |ז'א'ה ,ה'י'נ'ע'י'א'ס'מ'ע' ,ר'ק'פ' ,י'א'י'ד'ל'ש'י'ל' }
{\righttoleft |ז'א'ה ,ה'י'נ'ע'י'א'ס'מ'ע' ,ר'ק'פ' ,י'א'י'ד'ל'ש'י'ל' }
{\ar \righttoleft |ז'א'ה ,ה'י'נ'ע'י'א'ס'מ'ע' ,ר'ק'פ' ,י'א'י'ד'ל'ש'י'ל' }
{\he \righttoleft |ז'א'ה ,ה'י'נ'ע'י'א'ס'מ'ע' ,ר'ק'פ' ,י'א'י'ד'ל'ש'י'ל' }
{\ar \righttoleft |ז'א'ה ,ה'י'נ'ע'י'א'ס'מ'ע' ,ר'ק'פ' ,י'א'י'ד'ל'ש'י'ל' }
{\he \righttoleft |ז'א'ה ,ה'י'נ'ע'י'א'ס'מ'ע' ,ר'ק'פ' ,י'א'י'ד'ל'ש'י'ל' }

```

الذہمینعوا مَعَ قِرْكَوٰی اٰیٰیْدَلَسْیٰ
 ر.موا הז המ גשומ יל ןיא

لَيْسَ لَدَيَّ أَيُّ فِكْرَةٍ عَمَّا يَعْنيهِ هَذَا.
 אין לי מושג מה זה אומר.
 لَيْسَ لَدَيَّ أَيُّ فِكْرَةٍ عَمَّا يَعْنيهِ هَذَا.

אין לי מושג מה זה אומר.
 لَيْسَ لَدَيَّ أَيُّ فِكْرَةٍ عَمَّا يَعْنِيهِ هَذَا.
 אין לי מושג מה זה אומר.

In principle you can also rely on automatic direction changes, for instance by using the following command:

```
\setupdirections
  [bidi=global,
   method=three]
```

But that doesn't do a font switch for you, nor does it do any of the other language related settings. It really helps if you properly tag your document content, as in:

```
{\ar א' ז' ה' , ה' י' , נ' ע' י' א' ס' מ' ע' , ם' ר' ה' , פ' ' י' א' ' י' ד' ל' ' ש' ' י' ל' }
{\he רמוא הז המ גשומ יל ןיא }
```

One reason to set the `font` parameter for a language is that it will activate the right features in a font. Instead of falling back on some default, we can be very specific in what we want to enable.

3 A mixed layout

The typesetting engine normally works from left to right and top to bottom. Going from right to left actually involved two decisions:

- the direction of the display elements, the paragraphs
- the direction of the inline text, the lines

The first one is kept track of in a state variable. Every paragraph starts with a node that carries, among other information, that state. This node is added automatically and does not interfere with the typesetting. The inline direction is more intrusive as it is marked by nodes that indicate the beginning and end of a reversed strip. This mechanism is rather reliable and normally works out well. Take this:

```
left {\righttoleft right} left
left{ \righttoleft right} left
left {\righttoleft right }left
left{ \righttoleft right }left
```

You can see that we need to be careful with spaces as they can end up inside or outside a substream and by swapping next to each other:

```
left thgir left
left thgir left
left thgirleft
left thgirleft
```

We can wrap the lines in boxes as in:

```
\hbox{left\space{\bf\righttoleft right}\space left}
\hbox{left{\bf\space \righttoleft right}\space left}
\hbox{left\space{\bf\righttoleft right\space}left}
\hbox{left{\bf\space\righttoleft right\space}left}
```

When visualize the spaces we get this:

The space of a normal and bold font in the same family normally is the same but let's mix with a larger size:

```
\hbox{left {\bfa\righttoleft right} left}
\hbox{left{\bfa\space \righttoleft right} left}
\hbox{left {\bfa\righttoleft right }left}
\hbox{left{\bfa\space\righttoleft right }left}
```

Now we get the following. As you can see, it really matters where we put the braces.



Figure 3.1 Watch your spaces!



Once you are accustomed to tagging and $\text{T}_{\text{E}}\text{X}$ you will probably not fall into these traps. In figure 3.1 we show a large version.

The `\righttoleft` command actually has two meanings. This can best be seen from an example.

```
\righttoleft \bf How will this come out?
```

?tuo emoc siht lliw woH

```
And \righttoleft \bf how will this come out?
```

And **?tuo emoc siht lliw woh**

When we start a paragraph (or in $\text{T}_{\text{E}}\text{X}$ speak: when we are still in vertical mode) the paragraph direction as well as the inline direction is set. Otherwise only the inline direction is set. There are low level $\text{T}_{\text{E}}\text{X}$ commands (primitives) to set the direction but you can best *not* use these because we need to do a bit more than that.

There are quite some low level commands related to changing directions. Some deal with the layout, some with boxes. We might provide more in the future.

<code>\lefttoright</code>	l2r dir node or paragraph property
<code>\righttoleft</code>	r2l dir node or paragraph property
<code>\checkedlefttoright</code>	l2r dir node or paragraph property (unless already set)
<code>\checkedrighttoleft</code>	r2l dir node or paragraph property (unless already set)

<code>\synchronizeinlinedirection</code>	pickup a (possibly) reset state
<code>\synchronizelayoutdirection</code>	pickup a (possibly) reset state
<code>\synchronizedisplaydirection</code>	pickup a (possibly) reset state

<code>\righttoleft hbox</code>	r2l <code>\hbox</code>
<code>\lefttoright hbox</code>	l2r <code>\hbox</code>
<code>\righttoleft vbox</code>	r2l <code>\vbox</code>
<code>\lefttoright vbox</code>	l2r <code>\vbox</code>
<code>\righttoleft vtop</code>	r2l <code>\vtop</code>
<code>\lefttoright vtop</code>	l2r <code>\vtop</code>

<code>\lefttorright hbox</code>	l2r or r2l <code>\hbox</code>
<code>\lefttorright vbox</code>	l2r or r2l <code>\vbox</code>
<code>\lefttorright vtop</code>	l2r or r2l <code>\vtop</code>

<code>\autodir hbox</code>	l2r or r2l <code>\hbox</code> (a bit more clever)
<code>\autodir vbox</code>	l2r or r2l <code>\vbox</code> (a bit more clever)
<code>\autodir vtop</code>	l2r or r2l <code>\vtop</code> (a bit more clever)

<code>\bidilre</code>	character U+202A, enforce l2r state
<code>\bidirle</code>	character U+202B, enforce r2l state
<code>\bidipop</code>	character U+202C, return to last state
<code>\bidilro</code>	character U+202D, override l2r state
<code>\bidirlo</code>	character U+202E, override r2l state
<code>\lefttorightmark \lrm</code>	character U+200E, l2r indicator
<code>\righttoleftmark \rlm</code>	character U+200F, r2l indicator

<code>\dir lre</code>	switch to l2r mode using <code>\bidilre</code> or
<code>\dir rle</code>	switch to r2l mode using <code>\bidirle</code> or
<code>\dir lro</code>	enforce l2r mode using <code>\bidilro</code> or
<code>\dir rlo</code>	enforce r2l mode using <code>\bidirlo</code> or

<code>\natural hbox</code>	a normal l2r <code>hbox</code>
<code>\natural vbox</code>	a normal l2r <code>vbox</code>
<code>\natural vtop</code>	a normal l2r <code>vtop</code>
<code>\natural hpack</code>	a normal l2r <code>hpack</code>
<code>\natural vpack</code>	a normal l2r <code>vpack</code>

When we talk about layout, we mean the overall layout, concerning the document as a whole. We can have a dominantly l2r, dominantly r2l or mixed setup. In a next chapter we will give more details on the dominant setup. Here we stick to mentioning that the document flow direction is set with

```
\setupalign[r2l] % or r2l
```

When a command to setup an environment has a `align` parameter, the same keywords can be used as part of the specification.¹

¹ We haven't tested all situations and possible interferences. Just report anomalies to the mailing list.

4 Numbering and positioning

todo: columns (direction key), numbers (conversionsets), margins (begin/end), etc

5 The LUA interface

We assume that you run `CONTEXT MKIV` in combination with `LUATEX`. Direction support in this engine has been improved over time. Originally the OMEGA (ALEPH) direction model was used but in the meantime it has been stripped to the basics, and what used to be so called whatsits (extension nodes) are now first class nodes. Of the many directions only four are kept in `LUATEX` and they are indicated by three letters:

- 0 `TLT` left to right
- 1 `TRT` right to left
- 2 `LTL` not used in context (obsolete)
- 3 `RTT` not used in context (obsolete)

In `LUAMETATEX`, and therefore `CONTEXT LMTX` we only have the first two. Therefore in `LMTX` you normally don't have to worry about checking for them at the LUA end because they are irrelevant for calculations (the vertical ones swapped the horizontal and vertical progression). Also, when really needed, we use the `direction` keys with numerical indicators, so zero for `l2r` and one for `r2l`. These values are used for local par nodes as well as direction nodes. In addition a direction node has a subtype:

- 0 `normal` comparable to `+`
- 1 `cancel` comparable to `-`

6 Going vertical

Normally the term bidi is reserved for horizontal direction swapping but there is no reason to limit our view to that. So, here I will spend some words on how we can deal with vertical directions.

I will move some (not yet public) explanation from elsewhere to here in due time.