Simon's Improved Layout Engine

SILE is a new text layout engine. It takes the best ideas from TeX and friends (pdftex, XeTeX and so on), adds some ideas of its own, and is implemented from the ground up using modern technologies. SILE's focus is on extensibility and clean design.

1 Installing

SILE requires working installations of Lua, Pango and Cairo. Once you have these in place through your OS's package manager, you then need to install SILE's prerequisite modules:

```
> luarocks install stdlib luaepnf luaexpat lgi lpeg ...
```

Until SILE has its own working installer, you should manually copy the `sile` binary to `/usr/local/bin/`, and copy the `class`, `core`, `languages` and `packages` directories to `/usr/local/sile`. (Alternatively, you can set the `SILE_PATH` environment variable to the directory where you downloaded SILE.)

2 SILE Concepts

If you are familiar with TeX, most of SILE's concepts will be familar to you. There are a couple of new ideas:

Package. A SILE package is a Lua file which extends the way SILE operates. It may implement new commands, change the operation of the typesetter, and so on. This is distinct from a SILE file, which contains text to be typeset and basic macros.

Frame. Whereas TeX makes up pages entirely out of boxes and glue, SILE implements the *frame* as an additional abstraction between boxes and pages, and there can be multiple frames on a page. (There are two frames on this page: a content frame and a folio frame.) We will example frames in more detail later, but the basic concept should be familiar to anyone who has used graphical page layout tools such as InDesign or PageMaker.

3 A Basic SILE document

SILE documents are, by convention, either XML files or TeX-like files.

SILE is designed so that its back-end component and its input front-end component know little about each other, and it's perfectly possible to write another input processing front-end.

End-users may wish to use the TeX-like front-end, while those generating files for typesetting from existing data may find it easier to emit an XML file.

Here is the most basic example of a SILE document in XML style:

```
<sile papersize="a4">
Hello world.
</sile>
```

And here it is in TeX-like style:

```
\begin[papersize=a4]{document}
Hello world.
\end{document}
```

This will produce an A4 document with the text "Hello world" at the top, and the page number centered at the bottom. The `papersize` attribute is required; SILE knows about a number of standard paper sizes, but if you wish to use a non-standard size, you can specify the page size as width-by-height dimensions like so: `papersize="129mm x 198mm"`.

(SILE understands dimensions specified in mm, cm, in, pt or px. When the paper size is established, you can also specify dimensions as a percentage of the page size; horizontal dimensions will be understood as a percentage of the page width, and vertical dimensions as a percentage of the page height.)

The TeX-like format is sufficiently TeX-like to reassure intermediate users of TeX and deeply frustrate advanced users. It does not attempt to implement the TeX macro language to any degree, but consists of the following two syntactic elements:

• A command, with optional parameters and optional content:

```
\command[key=value,key=value]{content}
```

• An environment, with optional parameters and required content:

```
\begin[key=value,key=value]{command}
...
\end{command}
```

And that's all. There is no difference between an environment and a command with an argument.

4 SILE Commands

SILE has a number of built-in commands.

4.1 Including things

You can include another SILE file using the `include` command, like so:

```
\begin[papersize=a4]{document}
Hello world.
\include[src=chapter1.sil]
\include[src=chapter2.sil]
\end{document}
```

A TeX-like document can include an XML SILE document and vice-versa.

Or you can include a SILE package using the `script` command:

```
<sile papersize="a4">
<script src="packages/image" />
Hello world.
<img src="smiley.png" width="100px" height="100px"/>
</sile>
```

The brave and foolhardy can include raw Lua code within a `script` command or environment:

```
<sile papersize="a4">
Hello world.
<script>print("Hello Lua");</script>
</sile>
```

However, it is recommended that any serious scripts go in their own package file.

4.2 Styling things

The basic way to change the way text appears is using the `font` command. This command takes the following attributes:

`family`: Specifies the font family. This can be done in a CSS-like "stack", a comma-separated list. SILE will use the first font it can find which provides the characters you are trying to typeset.

`size`: The font size in points.

`weight`: The font weight in CSS numeric format.

`style`: The font slant style ("oblique", "italic" or "normal").

`language`: A two-letter language tag, which will be used for font shaping and hyphenation.

You may provide as many or as little of these attributes as required. The command can be used in two ways. An empty command changes the parameter permanently until the next command:

```
<font size="24" weight="700"/>
I am shouting here!
<font size="12" weight="400"/>
I am not shouting any more.
```

Whereas a command with content will only change the parameter for the content within the command, restoring the previous font style afterwards:

```
SILE is <font style="italic">very</style> easy to use
```

Of course, it would be boring to have to type

`<font style="italic">` all the time, but we will improve that later.

4.3 Moving things

To round out the list of built-in commands, advanced users can typeset particular nodes directly, using the `penalty`, `glue`, and `skip` commands:

```
<penalty penalty="500" flagged="1"/>
<glue width="0" shrink="0" stretch="10000"/>
<skip height="2pt" shrink="0" stretch="2pt"/>
```

So, for instance, you can center a line by saying:

```
<glue stretch="10000"/>Hello, world<glue stretch="10000"/>
```

a

4.4 Macros

However, this gets cumbersome fast. Thankfully, SILE allows you a basic way to easily define your own commands. The last built-in command you need to know about is the `define` command. For instance, this document includes a macros file which starts with the following defines:

```
<define command="center"><glue stretch="10000"/><process/><glue stretch="10000"/></define>
<define command="em"><font style="italic"><process/></font></define>
<define command="sectionsfont">
 <font family="Frutiger LT Std" weight="700"><process/></font>
</define>
```

As you can see, we define a new command with the `define` command, giving the command name as an attribute. When this command is used, the content inside the `define` command will be processed as though it appeared in that location of the document file. Additionally, the `process` instruction instructs SILE to process the contents of the command. At risk of belabouring the obvious, this means that:

```
SILE is <em>very</em> easy to use!
```

is now precisely equivalent to

```
SILE is <font style="italic">very</font> easy to use!
```

Macros cannot take any parameters or attributes to modify their activity other than `<process>`ing their contents. If you want parameters, define a command using a package. Macros can, however, contain other macros, and this gives a certain degree of expressibility:

```
<define command="headline">
<skip height="1cm"/>
<sectionsfont><center><font size="20"><process/></font></center></sectionsfont>
<skip height="0.5cm"/>
</define>
```

5 Frame and flow

As SILE is still in development, not all of this section is precisely true yet.

As mentioned above, SILE arranges your input on the page in the form of *frames*. The current page contains four frames, just to make things a bit more interesting. I have also told SILE to debug the frames by labelling them and drawing a box around them.

The four frames are:

a: the main content frame, which is the same as on every page. SILE was typesetting into this frame until the start of this section heading when we moved everything around.

a1: A left column frame. We told SILE to stop typesetting into *a* at the start of this section and start typesetting into *a1* instead, which is why the width of the text has changed.

b1: A right column frame, which SILE will typeset into once *a1* has been completely filled up. In other words, *a1* and *b1* are connected together; *b1* is logically `after' *a1* and text will flow from *a1* into *b1*. SILE calls

these *flowed frames*.

folio: This is not a flowed frame, it is a floating frame. The main body of the text will never run into this frame; text only gets put there explicitly. In the case of *folio*, the output routine which runs at the end of a page will put the current page number into the folio frame. (Typesetters call page numbers 'folios', which is why SILE calls the special page number frame *folio* by convention.) Other floating frames might be populated by footnote commands or floating images.

You can set a page layout for the whole of a document, or on a page-by-page basis. You can even change the page layout while SILE is typesetting the page, as we have done in this example.

5.1 Document classes

Each document has a *class* which specifies, amongst other things, the default page layout. If you don't specify a `class` attribute on the `<sile>` document command, you get the `plain` class. In other words, this:

```
<sile papersize="a4">
Hello world.
</sile>
```

is equivalent to

```
<sile papersize="a4" class="plain">
Hello world.
</sile>
```

In most cases you will specify a document class which will provide additional commands as well as a page layout.

The plain class defines two frames; a content frame with left and right margins of 5% of the page's width, a top margin of 5% and a bottom margin of 10%. It also has a folio frame. Looking at the source of the `plain` class will help you understand this:

```
  plain.pageTemplate.frames =
  "a": SILE.newFrame( left: "5%", right: "95%", top: "5%", bottom: "90%", id: "a" ),
  "folio": SILE.newFrame(left: "5%", right: "95%", top: "92%", bottom: "97%", id: "folio")
;
plain.pageTemplate.firstContentFrame = plain.pageTemplate.frames["a"];
```

We define two frames, and then specify that frame `a` is the first content frame. Ordinary document text will be flowed into this frame. To connect multiple flowed frames together, you can specify a `next` parameter with the ID of the next frame. We'll see this in a minute.

Since not everyone wants to mess about with Lua, SILE also has a command at the XML level which allows you to specify the layout for a page. For instance, to set up a page with two columns and a folio frame, we could say this:

```
<pagetemplate first-content-frame="lcol">

<frame id="lcol" top="5
```

There are a few things to note about this declaration. First, notice that the `first-content-frame` attribute specifies where text should be placed. If SILE is in the middle of typesetting a frame when it sees this declaration, it stops working on the current frame at a convenient point <note>(Due to the way that SILE arranges your input into boxes---which is the same as the way TeX does it---that may not be exactly at the point where the `pagetemplate` command appears.)</note> and starts writing text into the new `first-content-frame`.

Second, you should notice that the dimensions of frames can be given relative to other frames. These frames must already be defined; in other words, you cannot use this facility as a constraint system. However, you can use ordinary "calculator arithmetic" in these expressions: if you want to specify a dimension in the middle of two frames, you could say `(bottom(a) + top(b)) / 2`.

• More about absolute versus relative frames here.
• Insertions

6 Using SILE packages

SILE comes with a number of packages which provide additional functionality, and it is expected that other packages will be created which may or may not ship with SILE in the future. Those packages which ship as part of SILE are documented here.

6.1 counters

The counters package, which can loaded via `<script src="packages/counter" />`, provides three new SILE commands: `set-counter`, `show-counter`, and `increment-counter`. They are best understood by example.

```
<set-counter id="section" value="1"/>
This is section <show-counter id="section"/>. ("This is section 1.")
This is section <increment-counter id="section"/>. ("This is section 2.")
This is section <increment-counter id="section" display="roman"/>. ("This is section ii.")
This is section <increment-counter id="section" display="Roman"/>. ("This is section II.")
```

The plain class uses the counter `folio` for page numbers. We have made the current page number appear in Roman numbers by issuing the command `<set-counter id="folio" display="roman"/>`.

<set-counter id="folio" display="roman"/>

6.2 grid

<script src="packages/grid"/> <grid spacing="18pt"/>

The `grid` package alters the way that SILE's typesetter operates to implement grid-based typesetting. In other words, the baseline of each line of text will be aligned to fit upon a regularly-spaced grid. <note>(It is quite a small package, and reading the source is quite instructive for SILE hackers; it is a good demonstration of SILE's flexibility to see how much can be customized in quite a straightforward way.)</note>

The `grid` command takes a single attribute called `spacing`, which is a standard SILE dimension. After this empty command, all following text until SILE sees a `no-grid` command will be typeset on a grid. This section is typeset with `<grid spacing="18pt"/>`.

<no-grid/>

6.3 image

SILE has rudimentary support for including images. At present, those images must be PNG files <note>(For the technically inclined, this is because we delegate the handling of images to Cairo, which only supports PNG, PDF and SVG.)</note> and both their width and height must be specified, although SILE will handle scaling. (It will not, at present, scale proportionately given one dimension. <note>Or rather, it will, but the hbox will be the wrong size and so surrounding text will appear in the wrong place. This is because SILE doesn't know how big the image is until it's passed to Cairo for outputting.</note>)

An image is included like so: `<img src="smiley.png" width="100px" height="100px"/>`. Yes, we tried to keep this familiar to HTML people.

7 SILE hacking

XXX later.