The SILE Book

for SILE version 0.9.2

Simon Cozens

Table of Contents

Chapter 1
SILEとは？

SILEは組版システムです。その目的は美しい文書を生成することにあります。SILEについて理解する最も良い方法は、あなたが聞いたことがあるであろう他のシステムと比較することでしょう。

1.1 SILEとWord

多くの人たちはパソコンを使って印刷用に文書を作成するとき、Word（Microsoft Officeの一部です）やWriter（OpenOfficeやLibreOfficeに含まれます）といったソフトウェア、あるいはそれらに似たワープロソフトを利用します。 しかしながら、SILEはワープロソフトではありません。それは組版システムです。そこにはいくつかの重要な違いがあります。

ワープロソフトの目的はあなたがスクリーン上で入力したものと全く同一に見える文書を作成することにあります。一方、SILEはあなたが入力したものを、文書を作成するための指示だとみなし、それをもとに可能な限り良く見える文書を生成します。

少し具体的にみてみましょう。ワープロソフトでは、あなたが入力をしている文章が行の右端にさしかかると、カーソルは自動的に次の行に移動します。ワープロソフトは改行位置をあなたに示してくれます。SILEでは、あなたが文章を入力している段階では改行位置を知らせてくれません。その段階ではまだそれは明らかになっていないからです。あなたは好きなだけ長い行を打ち込むことができます。SILEはそれを処理する段階になって、パラグラフを構築するために、文章の最適な改行位置を探します。この処理はひとつの入力に対して（最大で）3回行われます。2つの連続した行がハイフネートされた語で終わっていないか、などあまりよろしくない状況が考慮され、最適な改行位置が見つかるよう処理が繰り返されます。

ページ分割に対しても同様です。ワープロソフトではいずれあなたは新しいページに移動することになりますが、SILEでは入力自体は好きなだけ継続されます。文章がどのようにページに分割されるかは文書全体のレイアウトを検討したのちに決定されるからです。

ワープロソフトはしばしばWYSIWYG—What You See Is What You Get（見たままが得られる）—であると言われます。SILEは全くWYSIWYGではありません。実際、結果はそれが得られるまで分からないのです。むしろ、SILE文書はテキストエディタ—テキストを入力するためのもので、整形された文書を作成するためのものではない—を用いて準備され、PDF文書を生成するためにSILEによって処理されます。

言い換えると、SILEはあなたが求める結果を記述するための言語であって、SILEはあなたが与えた指示に対し、最良の印刷物を得るための文書整形の処理を行います。

1.2 SILEとTeX

いくらかの人たちは、なんだかTeXのようだ、と思うかもしれません。[1] もしあなたがTeXに

1. ひとりのTeXユーザとして言わせれば“なんだかTEXのようだ”だろうか。

ついてよく知らない、あるいは関心がないのであれば、このセクションは読み飛ばしてもらっても構いません。

実際、TeX のようだというのは正しい意見です。SILE は TeX からかなりのものを引き継いでいます。SILE のような小さなプロジェクトが、TeX という、“The Art of Computer Programming” の著者たる某教授の、偉大な創造物の後継者だと名乗るのはおこがましいかもしれませんが …SILE は TeX の現代的な再創生です。

TeX は組版システムのなかでも最初期のもののうちのひとつで、それゆえほとんど何もないところから設計されなければなりませんでした。そのうちいくつかは時の試練に耐え — そして TeX はその創造から 30 年以上たった今でも最もよく利用される組版システムのうちのひとつであり、それはその設計とパフォーマンスの証である — 多くはそうではありませんでした。実際、Knuth の時代からの TeX の発展の歴史の大部分は彼の元々の設計を取り除き、新たな業界標準技術で置き換えることでした。例えば、我々は METAFONT ではなく TrueType フォントを使い（xetex のように）、DVI ではなく PDF を使い（pstex や pdftex）、7 ビットの ASCII ではなく Unicode を使い（これも xetex）、マクロ言語ではなくマークアップ言語や組込みのプログラミング言語を使います（xmltex や luatex）。現在、我々が 依然として利用する TeX のオリジナルの部分は、⑴ ボックスとグルー・モデル、⑵ ハイフネーション・アルゴリズム、⑶ 改行処理アルゴリズムです。

SILE は上記 3 つの点を TeX から受け継いでいます。SILE は TeX の改行処理アルゴリズムのほぼ丸写しな移植を含み、それは同じ入力が与えられたとき、TeX と全く同じ出力が得られるようにテストされています。しかしながら、SILE 自身がスクリプト言語で書かれているため、[2]SILE の組版エンジンの動作を拡張したり、変更したりすることが容易にできます。

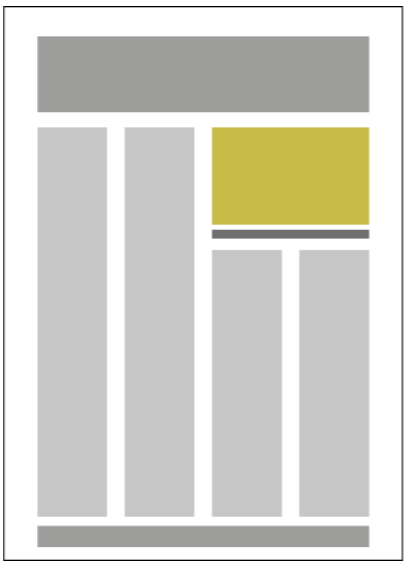
例えば TeX が苦手とすることのひとつとしてグリッド上での組版があります。この機能は聖書を組むような人にとっては重要なものです。これを TeX 上で行う試みはなされてきましたが、どれもひどいものでした。SILE では組版エンジンの動作を変更し、ごく簡単なアドオンパッケージを用意することでグリッド上での組版を可能にします。

もちろん、いまどきだれも plain TeX を使いません — だれもが LaTeX で同様のことを行い、そのうえ CTAN から入手可能な巨大なパッケージ群を活用しています。SILE は未だ TeX が持つような巨大なコミュニティやリソースを持たず、そのようなものを活用することができません。この点において TeX は SILE よりもずっと先を行っています。しかし、 可能性 という点において、TeX と同等か、あるいはもっと進んでいるとも言えるところがあるかもしれません。

2. もしもあなたが`TeX capacity exceeded` というメッセージに馴染んでいるならば、これはさぞかし興味深いことでしょう。

1.3 SILE と InDesign

人々が出版物をデザインするとき辿りつくツールとして InDesign（あるいはそれと似た DTP ソフト、例えば Scribus）があります。



InDesign は複雑で高価な商用出版ツールです。それは非常にグラフィカルです — クリックやドラッグといったマウス操作でテキストや画像をスクリーン上で移動させます。SILE は自由なオープンソースの組版ツールで、完全にテキストベースです。SILE ではエディタでコマンドを入力し、それらのコマンドをファイルに保存し、SILE に組版させるために渡します。これらの根本的な違いにかかわらず、この 2 つには共通した特徴があります。

InDesign では文章はページ上の フレーム に流しこまれます。左の図は InDesign でよくあるレイアウトがどのようなものかを示しています。

SILE もまたページ上でどこに文章が表示されるべきかをフレームという概念を用いて決定します。そのため SILE では TeX でできうるよりもっと複雑で柔軟なページレイアウトを設計することが可能です。

InDesign で有用な機能として、構造化された XML データ形式を用いたカタログや名簿などの出版があります。InDesign でこれを行うには、まずそれぞれの XML 要素にどのようなスタイルが適用されるか宣言します。データが InDesign に読み込まれると、InDesign は与えられたルールに従ってデータを整形し出力します。

あなたは全く同じことを SILE でできるのです。ただし SILE では XML 要素がどのように整形されるのかをより詳細に制御することができ、これは SILE ではあなたが XML 要素を処理するのに、例えば Lua コードを呼び出したりすることができるからです。SILE はコマンドラインのフィルタープログラムであるため、適切な指示が与えられれば、XML ファイルから PDF へ、いとも簡単に変換することができます。これは素晴らしいことです。

この解説書の最後の章では、複雑な XML 文書をスタイル付して PDF を生成するための クラスファイル のいくつかの例を示します。

1.4 結論

SILE は入力として与えられたテキストの指示をもとに PDF を出力します。SILE は TeX と InDesign にインスパイアされた機能を持ち、かつより柔軟で拡張可能、プログラム可能なものを目指しています。この文書（これは SILE で書かれています）のようなものを作成したり、構造化されたデータを整形して出力するシステムとして有用です。

Chapter 2
さあ始めよう

さて、SILE とは何か、何をするものなのか、いくらか理解したところで SILE そのものについて話題を移しましょう。

2.1 基本的な SILE 文書

SILE をどうやって使用するのか示す前に SILE 文書がどのようなものなのかひとつ例を示しましょう。 これは SILE に対する入力であり、SILE によって処理され PDF ファイルへと変換されるものです。

> これらの文書はプレーンテキストです。あなたがあなた自身の SILE 文書を作成するにはテキストエディタが必要です。Unix 上では例えば、vi やemacs、Mac OS X では Sublime Text、TextMate、あるいは TextEdit など、Windows では Notepad や Notepad+ などです。SILE の入力として用いるにはテキストファイルとして保存する必要があります。Word のようなワープロソフトでは作成できません。それらは文書をプレーンテキストではなく独自のフォーマットで保存するからです。

とりあえず、もっとも簡単な SILE 文書から始めましょう。

```
\begin[papersize=a4]{document}
Hello SILE!
\end{document}
```

今のところは、SILE 文書はこのようなものだというだけにしておいて、詳細は次の章で取り上げましょう。

分かり切ったことを言うようですが、これは左上部に**Hello SILE** と書かれ、ページ番号（1）がページ下部中央に配置された A4 サイズの PDF 文書を生成します。さて、どうやってその PDF を得るのでしょうか ？

2.2 インストール

なにはともあれ、あなたは SILE を手に入れ、あなたのパソコンで走らせなければなりません。SILE はホームページ`http://www.sile-typesetter.org/` から入手できます。

SILE をインストールし、実行するにはいくつか他のソフトウェアが必要です —Lua プログラミング言語のインタプリタと Harfbuzz テキストシェーピング・ライブラリです。

SILE にはそれ自身の PDF 生成ライブラリが付属しており、それもまたいくつかのソフトウェアを要求します。`freetype`、`fontconfig`、`libz`、そして`libpng`です。[1]Homebrew（Mac OS X ではおすすめです）を利用する Mac OS X 上でこれらをインストールするには、

- `brew install automake libtool harfbuzz fontconfig libpng lua luarocks freetype`

Debian や Ubuntu などの Debian 系 Linux OS では、

- `apt-get install lua5.1 luarocks libharfbuzz-dev libfreetype6-dev libfontconfig1-dev libpng-dev`

Redhat 系 Linux では次のようになるでしょう。

- `yum install harfbuzz-devel make automake gcc freetype-devel fontconfig-devel lua-devel lua-lpeg lua-expat libpng-devel`

これらの依存ライブラリがインストールされれば、次は Lua ライブラリをそろえる必要があります。

- `luarocks install lpeg luaexpat`

以上のことが済めばようやく本題に移れます。SILE のホームページからダウンロードしたファイルを解凍し、ディレクトリを移動してから以下を実行します。

- `./configure; make`

これが終われば SILE を未インストールの状態で実行できます。[2]

- `./sile examples/simple.sil`

すべてが順調であれば、`examples/simple.pdf`というファイルが生成されるはずです。
SILE を本格的に使うには`sile`コマンドと SILE ライブラリ・ファイルをシステムにインストールします。これを行うには次のようにします。

- `make install`

これで`sile`コマンドがどのディレクトリからも利用可能になりました。

2.3 SILE の実行

では新たなディレクトリに移り、テキストエディタを開いて先ほど例示した内容をファイル`hello.sil`に保存しましょう。そしてコマンドを実行します。

- `sile hello`

1. 代わりに Pango と Cairo を使うようにもできますが、その出力は特に Linux において劣ります。 あえてそうする場合は`libcairo-gobject2`と`libpango1.0-0`パッケージをシステムにインストールし、`lgi` Lua モジュールを追加する必要があります。
2. Mac OS X で SILE をシステムにインストールせずに実行するには、環境変数を次のようにして設定する必要があります

```
DYLD_LIBRARY_PATH=./libtexpdf/.libs ./sile examples/simple.sil
```

（SILE は引数のファイル名に拡張子が与えられなければ、自動的に拡張子`.sil` を追加します）

これによってファイル`hello.pdf` ができるでしょう。あなたはめでたく SILE での最初の文書を作成することができました。

2.4 少し凝ってみる

`examples/article-template.xml` は典型的な DocBook 5.0 文書です。DocBook を印刷する場合、しばしば、XSLT プロセッサ、FO プロセッサ、そして場合によっては奇妙な LaTeX パッケージに振り回されなければなりません。しかし、SILE は XML ファイルを読み込むことができ、しかも DocBook（実際にはそのサブセット）を処理するための`docbook` クラスが付属しています。

例、`examples/article-template.xml` を`examples/article-template.pdf` に変換するには、単純にこうします。

```
% ./sile -I docbook examples/article-template.xml
This is SILE 0.9.2
Loading docbook
<classes/docbook.sil><examples/article-template.xml>[1] [2] [3]
```

ここで`-I` フラグは入力ファイルを読み込む前にクラスファイルを読み込むための指示です。 `docbook` クラスファイルが読み込まれたのち、DocBook ファイルは直接読み込まれ、タグは SILE コマンドとして解釈されます。

第 10 章では`docbook` クラスがどのようなものか見てみます。そこでは他の XML フォーマットをいかに処理するか学ぶでしょう。

さあ始めよう

Chapter 3
SILE文書の作成

さて、ここで最初の例に戻りましょう。

```
\begin[papersize=a4]{document}
Hello SILE!
\end{document}
```

文書は`\begin{document}`コマンドで始まります。それには用紙サイズの指定が 必須 です。そして文書は`\end{document}`で終わります。その間には 2 種類の SILE 文書を構成する要素が来ます。ページ上に出力されるテキスト、ここでは “Hello SILE!”、とコマンドです。

用紙サイズ

SILE は国際規格 ISO のＡ・Ｂ・Ｃシリーズの用紙サイズを認識します。これに加えて次の伝統的によく用いられる用紙サイズも利用可能です。letter、note、legal、executive、halfletter、halfexecutive、statement,、folio、

もしも標準的でない用紙サイズを指定したければ、具体的なサイズを直接指定することも可能です。`papersize=<basic length> x <basic length>`。

単位

SILE では長さを指定するいくつかの方法があります。上記`<basic length>`は数と単位（の省略記号）の指定からなります。認識される単位はポイント（`pt`）、ミリメートル（`mm`）、センチメートル（`cm`）、インチ（`in`）です。例えば、ペーパーバックサイズの B-format は`papersize=198mm x 129mm`のように指定されます。後ほど長さを指定する別の方法についてもみることとなるでしょう。

3.1 テキスト

通常のテキストについてはこれといって述べることはありません。単に入力してください。

TeX ユーザーは SILE がテキストについても何らかの処理を行うものと期待するかもしれません。例えば、あなたが TeX において、ふたつの連続したバッククォート（``）を入力すると、TeX はそれを開始用のダブルクォート（“）に置き換えてくれます。SILE はそのようなことは行いません。ダブルクォートを入力してください。同様に en ダッシュと em ダッシュでも、-- や--- ではなく、Unicode で該当する文字を入力してください。

テキスト処理においていくつか挙げる点があるとすれば以下のものでしょうか。

まずひとつ目は、スペースの扱いについてです。もしあなたがスペース 3 つを用いて`Hello SILE!` と書いたとしても、それはスペース 1 つ分、`Hello SILE!` と同じ結果になります。

同様に、改行文字を好きなところに入れることができます。[1]SILE はパラグラフ全体を取扱い、与えられた行幅で可能な、最適な改行位置を計算します。例として挙げるならば、あなたの入力が仮に

```
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
```

だったとしても、SILE の出力において ‘eiusmod’ で改行が起こるとは限りません。改行は常に、適切な位置で行われます。実際の出力は以下のようなものとなるでしょう。

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

パラグラフを終了する場合は、改行を 2 個続けて入れてください。例えば、

```
Paragraph one.

Paragraph two.
This is not paragraph three.

This is paragraph three.
```

注意点として挙げられるふたつ目は、いくつかの（4 つです）文字は SILE では特別な意味を持つことです。これらは TeX ユーザにとっては馴染み深いものでしょう。

バックスラッシュはコマンドの開始に用いられます。（コマンドの詳細については後ほどすぐに述べましょう）波括弧（`{`、`}`）はグループ化に、特にコマンドの引数を扱う際に、用いられます。最後にパーセント記号はコメント行の開始として用いられます。パーセント記号から次の改行文字までは SILE によって無視されます。

1. 訳注 : わかち書きをする言語ではです。改行文字はスペース 1 個分と同じように扱われます。

これらの文字を出力したければ、バックスラッシュを前に付けましょう。`\\` は '¥'[2] を、`\{` は '{' を、`\}` は '}' を、そして`\%` は '%' を出力します。
3つ目の点はハイフネーションです。SILE はそれによってパラグラフ全体の見た目が良くなると判断できるときはいつでも、自動的に語をハイフネートして改行します。ハイフネーションはその時の言語の設定が反映されます。特に指定がなければ、SILE はデフォルトで英語を仮定し、ハイフネーション処理を行います。上記のラテン語のテキストの例ではハイフネーションは無効化されています。

最後に挙げる点はリガチャです。（ふたつあるいはそれ以上の文字が、見た目を良くするために、ひとつの文字に結合される）SILE は自動的にリガチャ処理を行います。このため、あなたがもし`affluent fishing` と入力すると（実際には使用するフォントに依存します）、出力結果は 'affluent fishing' のようになります。リガチャを抑制したい場合は、空のグループ（グループ化文字`{` と`}` を使って）を挿入します。`af{}f{}luent f{}ishing` ではaffluent fishing のようになります。リガチャやそのほかの機能の制御に関する詳細についてはOpenType フィーチャ の節を参照してください。

3.2 コマンド

Typically (and we'll unpack that statement later), SILE commands are made up of a backslash followed by a command name, and a document starts with a `\begin{document}` command and ends with `\end{document}`.

A command may also take two other optional components: some parameters, and an argument. The `\begin` command at the start of the document is an example of this.[3]

```
\begin[papersize=a4]{document}
```

The parameters to a command are enclosed in square brackets and take the form *key=value*; multiple parameters are separated by commas or semicolons, as in `[key1=value1,key2=value2,…]` Spaces around the keys are not significant; we could equally write that as `[key1 = value1; key2 = value2; …]`. If you need to include a comma or semicolon within the value to a parameter, you can enclose the value in quotes: `[key1 = "value1, still value 1", key2 = value2; …]`.

The optional argument (of which there can only be at most one) is enclosed in curly braces.[4]

Here are a few more examples of SILE commands:

---

2. 訳注 ： フォントによっては円記号になってしまいます。
3. Strictly speaking `\begin` isn't actually a command but we'll pretend that it is for now and get to the details in a moment.
4. TeX users may forget this and try adding a command argument "bare", without the braces. This won't work; in SILE, the braces are mandatory.

```
\eject                                  % A command with no parameters or argument
\font[family=Times,size=10pt]            % Parameters, but no argument
\chapter{Introducing SILE}              % Argument but no parameters
\font[family=Times,size=10pt]{Hi there!} % Parameters and argument
```

3.3 Environments

Commands like `\chapter` and `\em` (emphasises text by making it italic) are normally used to enclose a relatively small piece of text; a few lines at most. Where you want to enclose a larger piece of the document, you can use an environment; an environment begins with `\begin{`*name*`}` and encloses all the text up until the corresponding `\end{`*name*`}`. We've already seen an example, the `document` environment, which must enclose the entire document.

Here is a secret: there is absolutely no difference between a command and an environment. In other words, the following two forms are equivalent:

```
\font[family=Times,size=10pt]{Hi there!}
```

```
\begin[family=Times,size=10pt]{font}
Hi there!
\end{font}
```

However, in some cases the environment form of the command will be easier to read and will help you to be clearer on where the command begins and ends.

3.4 The XML Flavour

While we're on the subject of alternative forms, SILE can actually process its input in a completely different file format. What we've seen so far has been SILE's "TeX-like flavor", but if the first character of the input file is an angle bracket (`<`) then SILE will interpret its input as an XML file. (If it isn't well-formed XML, then SILE will get very upset.)

Any XML tags within the input file will then be regarded as SILE commands, and tag attributes are interpreted as command parameters; from then on, the two file formats are exactly equivalent, with one exception: instead of a `<document>` tag, SILE documents can be enclosed in any tag. (Although `<sile>` is conventional for SILE documents.)

In other words, the XML form of the above document would be:

```
<sile papersize="a4">
```

```
Hello SILE!
</sile>
```

Commands without an argument need to be well-formed self-closing XML tags (for instance, `<break/>`), and commands with parameters should have well-formed attributes. The example above, in XML flavor, would look like this:

```
<font family="Times" size="10pt">Hi there!</font>
```

We don't expect humans to write their documents in SILE's XML flavor—the TeX-like flavor is much better for that—but having an XML flavor allows for computers to deal with SILE a lot more easily. One could create graphical user interfaces to edit SILE documents, or convert other XML formats to SILE.

However, there is an even smarter way of processing XML with SILE. For this, you need to know that you can define your own SILE commands, which can range from very simple formatting to fundamentally changing the way that SILE operates. If you have a file in some particular XML format—let's say it's a DocBook file—and you define SILE commands for each possible DocBook tag, then the DocBook file becomes a valid SILE input file, as-is.

In the final two chapters, we'll provide some examples of defining SILE commands and processing XML documents.