

# **The SILE Book**

for SILE version 0.9.2

**Simon Cozens**



# Table of Contents

<b>SILE とは?</b>	<b>1</b>
SILE と Word	1
SILE と TeX	1
SILE と InDesign	3
結論	3
<b>さあ始めよう</b>	<b>5</b>
基本的な SILE 文書	5
インストール	5
OS X ユーザのためのノート	6
Linux ユーザのためのノート	6
Windows ユーザのためのノート	7
SILE の実行	7
もうちょっとクールに	7
<b>SILE 文書の作成</b>	<b>9</b>
テキスト	9
コマンド	11
環境	12
XML 書式	12
<b>SILE コマンド</b>	<b>15</b>
フォント	15
文書構造	16
章と節	16
脚注	17
インデントとスペーシング	17
分割	17
言語とハイフネーション	18
ファイルの取り込みと Lua コード	19
<b>SILE パッケージ</b>	<b>21</b>
image	21
rules	22
color	23

rotate	23
features	24
unichar	24
bidirectional	25
raiselower	25
grid	25
verbatim	27
他のパッケージにより利用されるパッケージ	27
footnotes	27
counters	28
pdf	28
frametricks	29
insertions	29
twoside	30
masters	30
infonode	30

# 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。ふた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ー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。<sup>1</sup>もしあなたが TeX に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。

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。

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 (xmlltex 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、(1) ボックスとグルー・モデル、(2) ハイフネーション・アルゴリズム、(3) 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、それは同じ入力を与えられたとき、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、<sup>2</sup>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。

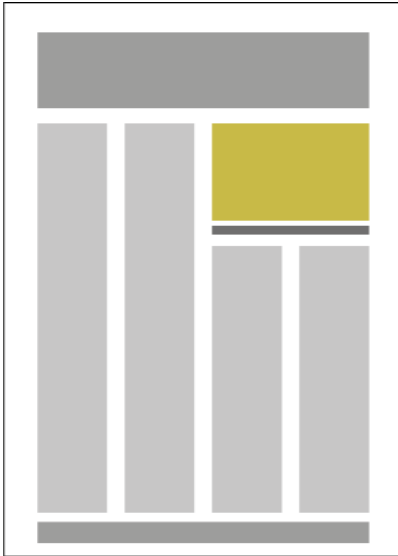
例えば TeX が苦手とすることのひとつとしてグリッド上での組版があります。この機能は聖書を組むような人にとっては重要なものです。これを TeX 上で行う試みはなされてきましたが、どれもひどいものでした。SILE では組版エンジンの動作を変更し、ごく簡単なアドオンパッケージを用意することでグリッド上での組版を可能にします。

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。

SILE とは？



# 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。

さあ始めよう

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。<sup>1</sup>

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**
- **luarocks install 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。

- **./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。

## 2.2.1 OS X ユーザのためのノート

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**

もしも OS X で Homebrew を利用しているのであれば、環境変数**PKG\_CONFIG\_PATH**を適切に設定する必要がありますかもしれません。(PKG\_CONFIG\_PATH=/usr/local/lib/pkgconfig:/usr/local/lib)あるいはライブラリ依存関係参照のための**pkgconfig**ファイルを提供する**brew link**を使用してください。configure スクリプトはどのライブラリが見つからなかったか警告してくれるでしょう。

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設定する必要があります。

**DYLD\_LIBRARY\_PATH=./libtexpdf/.libs ./sile examples/simple.sil**

## 2.2.2 Linux ユーザのためのノート

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。

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。

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

### 2.2.3 Windows ユーザのためのノート

Windows でもmingw32環境で SILE を動作させることができたとのユーザからの報告があります。現在のところ、確実な方法はありませんが、<https://github.com/simoncozens/sile/issues/82>での議論が参考になるでしょう。

## 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
```

(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 もうちょっとクールに

**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しましょう。

---

```
\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、quarto、ledger、tabloid。

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。<sup>1</sup>SILE はパラグラフ全体を取扱い、与えられた行長で可能な、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

1. 訳注： わかり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。

よって無視されます。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`%`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。リガチャやその他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

典型的な（この点に関しては後ほど再検討しましょう）SILE コマンドは、バックスラッシュで始まり、コマンド名が続く文字列です。そして文書は`\begin{document}`コマンドで始まり、`\end{document}`で終わります。

コマンドはまた、ふたつの必須でない部分を持ちます。それはパラメータと引数です。文書を開始する時の`\begin`コマンドはその良い例です。<sup>3</sup>

---

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

---

コマンドのパラメータは角括弧で囲まれ、`key=value`の形をとります。複数のパラメータを指定する場合は、コンマやセミコロンを使って、`[key1=value1,key2=value2,...]`のように続けます。`"key"`の前後のスペースは重要ではありません。`[key1 = value1; key2 = value2; ...]`のように書くこともできます。もしもコンマやセミコロンをパラメータの値に使いたければ、引用符で値全体を囲います。`[key1 = "value1, still value 1", key2 = value2; ...]`のように。

コマンドは引数をとるかもしれませんが、その場合は波括弧で囲います。<sup>4</sup>

以下にいくつかのSILEコマンドを示しましょう。

---

<code>\eject</code>	% A command with no parameters or argument
<code>\font[family=Times,size=10pt]</code>	% Parameters, but no argument
<code>\chapter{Introducing SILE}</code>	% Argument but no parameters

---

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. 厳密に言うと`\begin`はコマンドではありませんが、とりあえず今はそういうことにしましょう。
4. TeX ユーザはつい括弧を忘れてしまうかもしれませんが、それはいけません。SILE では括弧は必須です。

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

---

### 3.3 環境

`\chapter`や`\em` (イタリック体による強調)といったコマンドは、せいぜい数行の比較的短いテキストを囲むために用いられます。もっと長い、文書の一部を構成する部分を囲みたい場合は、環境を使います。環境は`\begin{name}`で始まり、対応する`\end{name}`までをその中に含みます。ひとつの例が既に出ていますね。document環境で、これは文書全体を囲みます。内緒ですが、コマンドと環境の間には全く違いはありません。いうなれば、以下のふたつは等価なのです。

---

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

---

しかしながら、いくつかの場面では、環境を用いたほうが読みやすく、どこからどこまでコマンドが影響するのか認識しやすくなります。

### 3.4 XML 書式

実際のところ、SILE はこれまで示したものとは完全に異なる入力フォーマットを受け付けます。これまで例示してきたものは「TeX 風書式」でしたが、もし入力ファイルの最初の文字が山括弧 (実際は不等号記号`<`) であった場合は、SILE は入力ファイルが XML 書式であると捉えます。[もしそれが整形式の (well-formed) XML 文書でなければ、SILE は非常に機嫌を損ねるでしょう]

入力ファイル中のすべての XML タグは、SILE コマンドであると解釈され、属性はパラメータであるとみなされます。このため、ふたつのファイルフォーマットは実際的には等価です。ただひとつの例外を除いては。XML 書式の場合は SILE 文書は任意のタグで始まってもよいのです。(習慣として SILE 文書には`<sile>`を用いるのが好ましいですが)

例えば、XML 形式で前述の例文を示すと、

---

```
<sile papersize="a4">
Hello SILE!
</sile>
```

---



引数を取らないコマンドはすべて整形形式の self-closing<sup>5</sup>タグ（例えば<break/>）でなければならず、パラメータ付のコマンドはその属性が整形形式でなければなりません。前に挙げた例を XML 書式で書くと、

---

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

---

XML 書式は人間が直接書くことを想定しているわけではありませんが—TeX 風書式のほうがそれには向いているでしょう—XML 書式に対応することは、コンピューターで SILE を扱うのをより容易にします。例えば SILE 文書を編集するための GUI インターフェイスを作ったり、他の XML 書式を SILE のそれに変換したり。

しかしながら、SILE においては XML 文書进行处理するためのよりスマートな方法が存在します。そのためには、あなたはあなた自身の SILE コマンド、それは非常に単純な文書整形用のものから SILE の動作を根本から変えるものまでを含む、を定義できることを知る必要があります。あなたがある特定の XML 形式のファイル—仮に DocBook としましょう—を持っているとします。あなたはすべての可能な DocBook タグに対する SILE コマンドを定義します。するとあなたの DocBook ファイルは SILE 入力ファイルとしてそのまま使えるようになるのです。

最後の 2 章では、SILE コマンドを定義と XML 文書进行处理する例を示しましょう。

5. 訳注：適切な訳語が分からないが、開始・終了のペアではなく、単体で存在するタグのこと。



# Chapter 4

## SILE コマンド

さて、それでは SILE の具体的な使用法について見ていきましょう。まずはあなたが SILE で文書を作成し始めるのに最も役立つコマンドから始め、次第により細かな点について進んでいきます。

### 4.1 フォント

テキストの見た目を変えるもっとも基本的なコマンドは `\font` コマンドです。これは次のような書式をとります。

- `\font[parameters...]{argument}`
- `\font[parameters...]`

最初の書式では引数として与えられたテキストを指定されたフォントで描画します。次の書式ではそれ以降のテキストすべてに影響します。

例として挙げると、

---

Small text

`\font[size=15pt]`Big text!

`\font[size=30pt]`{Bigger text}

Still big text!

---

は

---

Small text

Big text!

Bigger text

Still big text!

---

となります。

ここで見たように、属性として可能なものとして、`size`があります。これは、`<dimension>`で指定されます。ここで、`<dimension>`は以前登場した`<basic length>`のようなものですが、これは現在のフォントのサイズに対する相対的な値として指定可能です。例えば、ex ユニット (`ex`)、であったり、em ユニット (`em`)、あるいは en ユニット (`en`) です。

`\font` コマンドで指定可能な属性値は、

- `size` – 先に述べたとおりです。

- `family` – 使用するフォント名が来ます。フォントをその名前指定するには、SILE はシステムにインストールされたすべてのフォントについて知る必要があります。SILE の XML 書式では、フォントファミリーは CSS 形式のコンマで分離された‘スタック’として指定可能です。
- `style` – `normal` または `italic` です。
- `weight` – CSS 形式のウェイトを表す数値が来ます。有効な値は 100 と 200 から 300、400、500、600、700、800、900 までです。フォントによっては全てのウェイトがサポートされているとは限りませんが（ふたつ程度かもしれません）、SILE は最も近いものを選択します。
- `language` – 2 文字からなる (ISO639-1) 言語コードです。これはスペーシングとハイフネーションの両方に影響を与えます。
- `script` – スクリプト（文字体系、用字系）の指定です。後で述べる「言語とハイフネーション」の節を参照してください。

手で陽にフォント指定を行うのは非常に面倒ですね。後ほどこれを自動化する方法についても見てみましょう。SILE は `\em{...}` コマンドを `\font[style=italic]{...}` のショートカットとして提供します。ボールド体に対するショートカットはありません。なぜならそれはあまり良い習慣とは言えないからです。そのようなものを簡単に行う方法は与えないことにしましょう。

## 4.2 文書構造

SILE は様々な文書クラス (LaTeX のクラスと似た) を提供します。デフォルトでは、文書の構造化をごくわずかにサポートするのみの、`plain` クラスが用いられます。他には `book` クラスがあり、これは左右のページマスタ、ヘッダと脚注、章、節などのヘッディングをサポートします。

この節のコマンドを使うには、あなたの文書の `\begin{document}` コマンドで `book` クラスを指定する必要があります。あなたが今読んでいるこの文書は実際に、`\begin[papersize=a4,class=book]{document}` で始まります。

### 4.2.1 章と節

あなたは文書を `\chapter{...}`、`\section{...}`、そして `\subsection{...}` などのコマンドを使って分割することができます。これらのコマンドは引数として、その章や節のタイトルをとります。章は新たな左ページから始まり、章のタイトルは左ページのヘッダに表示されます。加えて、節のタイトルは右ページのヘッダに表示されます。

---

章や節は自動的に 1 から番号付けされて開始されます。この動作を変更するには、次の章の `counters` パッケージの解説を参照してください。番号付けを抑制したければ、パラメータ `[numbering=no]` を与えます。

---

この副節はコマンド `\subsection{章と節}` で開始されています。

### 4.2.2 脚注

脚注は`\footnote{...}`コマンドでつけることができます。<sup>1</sup>脚注コマンドに対する引数はページ下部に表示される脚注の内容です。これは各章ごとに、自動的に 1 から番号付けされます。

## 4.3 インデントとスペーシング

SILE では、パラグラフは通常インデントされます（デフォルトで 20 ポイント幅です）。これを抑制するには`\noindent`コマンドを、パラグラフの先頭に付与します。（このパラグラフのような、最初のパラグラフでは`\noindent`は必要ありません。なぜなら`\section`と`\chapter`は自動的に、章や節のタイトルに続く文章に対してそれを呼ぶからです）`\noindent`は`\indent`コマンドを続けて呼ぶことで打ち消すことができます。

パラグラフ間、あるいはパラグラフと他の要素との間の垂直方向のスペース分量を増やすには、`\smallskip`、`\medskip`および`\bigskip`が使えます。これらはそれぞれ、3pt、6pt、12pt のスペースに相当します。このパラグラフの後に`\bigskip`を入れてみましょう。

水平方向のスペースを行ないに挿入するには、小さなものから大きなものへ順に、`\thinspace`（em の 1/6）、`\enspace`（1em）、`\quad`（1em）、そして`\qquad`（2em）。

`center`環境中（`\begin{center} ... \end{center}`）では中央寄せとなります。例えばこのパラグラフのように。

## 4.4 分割

SILE は行とページの分割を自ら決定します。後の章ではこのプロセスを微調整する設定法を紹介しましょう。しかしながら、SILE の plain クラスにもそれを助けるためのいくつかの方法が存在します。

パラグラフ間に挿入された`\break`コマンドはフレーム分割を引き起こします。（`\framebreak`と`\eject`という同義のコマンドも存在します）もし、複数のフレームがページ内にあれば、—例えば、多段組みの文書—現在のフレームが終了し、次のフレームの先頭から処理は続けられます。`\pagebreak`（あるいは`\supereject`）はより強制力のあるもので、これはページ上に更なるフレームが残っていても新しいページを開始します。より穏やかな変種としては、`\goodbreak`、これは SILE にそこが良いページ分割点であると教えるもの、があります。それとは反対に、`\nobreak`は分割を抑止する働きがあります。これらの中間的なものとして、`\allowbreak`があり、SILE にページやフレームの分割に適さないかもしれないが、それを許可するよう指示するものとして利用できます。

パラグラフの中では、これらのコマンドは全く別の意味を持ちます。`\break`コマンドは改行を指示し、同様に、`\goodbreak`、`\nobreak`、および`\allowbreak`も行分割に対応します。もしもページ分割を特に禁止したければ、`\novbreak`を使います。

1. このように。 `\footnote{このように}`。

SILE は通常、両端揃えを行います—すなわち、SILE は一行がちょうど与えられた行長でぴったり収まるように単語間のスペースを調整します。<sup>2</sup>両端揃え以外には左揃えがあります。左揃えでは単語間のスペースは均等になるかわり、パラグラフの右端はきれいに揃いません。左揃えはしばしば子供向けの本に用いられたり、新聞のような一行の幅が狭い状況でも用いられます。左揃えを行うには、文章

を`\begin`

`{raggedright}`環境を囲います。このパラグラフは左揃えで組まれています。

同様に、`raggedleft`環境もあります。これはパラグラフの右側は揃え、逆に左はがたつきます。このパラグラフは右揃えで組まれています。

## 4.5 言語とハイフネーション

SILE は現在選択されている言語の設定に基づいてハイフネーションを行います。（言語設定は前に見たように`\font`コマンドで行います）SILE は様々な言語のハイフネーションをサポートしています。また、その言語特有の組版ルールについてもサポートすることを目的としています。

SILE はまた、`xx`という特別な「言語」、を理解します。これはなんのハイフネーションパターンもないものです。この言語に切り替えると、ハイフネーションは行われません。コマンド`\nohyphenation{...}`が`\font[language=xx]{...}`のショートカットとして利用できます。

ハイフネーション以外にも、言語ごとに組版上の規則は異なりますが、SILE はほとんどの言語とスクリプトに対する基本的なサポートを備えます。（もしも SILE が適切に処理出来ない言語やスクリプトがあれば知らせてください。対応します）

いくつかの言語では、同じ文字を使うが異なるように組まれるという状況が生じます。例えば、Sindhi と Urdu はアラビア文字`heh`を標準的なアラビア語とは異なるやり方で結合します。そのような場合は、あなたは`language`と`script`オプションを`\font`コマンド中で適切に指定しなければなりません。

---

Standard Arabic

```
\font[family=Scheherazade,language=ar,script=Arab]{\font{...}};
```

Sindi:

```
\font[family=Scheherazade,language=snd,script=Arab]{\font{...}};
```

Urdu:

```
\font[family=Scheherazade,language=urd,script=Arab]{\font{...}}.
```

---



---

Standard Arabic: ههه; Sindhi: ههه; Urdu: ههه.

---

（`script`オプションの完全なリストについては<http://www.simon-cozens.org/content/duffers-guide-fontconfig-and-harfbuzz>を参照）を一行に厳密に合うようにするということを意味しません。SILE はある程度の調整を行います、最善を尽くした後、最も悪くないと思われる結果を出力します。いくつかの語がわずかに余白に突き出る結果となることもあります。

## 4.6 ファイルの取り込みと Lua コード

長大な文書を作成するとき、あなたは SILE 文書を複数のファイルに分割して管理したくなるでしょう。例えば、それぞれの章を別のファイルに小分けしたり、ユーザー定義のコマンドを開発し（第 6 章を参照）、それをひとまとめのファイルにして文書の本文とは分けて管理したり。その場合、異なる SILE ファイルを取り込む必要があります。

その機能は `\include` コマンドにより提供されます。これには、必須の `src=<path>` パラメータによりファイルへのパスを示す必要があります。例えば、あなたは学位論文を次のように書きたくなるでしょう。

---

```
\begin[papersize=a4,class=thesis]{document}
\include[src=macros]
\include[src=chap1]
\include[src=chap2]
\include[src=chap3]
...
\include[src=endmatter]
\end{document}
```

---

`\include` は入れ子になっても構いません。ファイル A がファイル B を取り込み、それがまたファイル C を取り込んだり。

SILE は Lua プログラム言語で書かれており、Lua インタプリタが実行時に利用可能です。ちょうど HTML 文書中で Javascript コードを `<script>` タグで実行するように、SILE 文書中では Lua コードを `\script` コマンドを用いて実行可能です。（XML 書式ではちょうどよく見えるでしょう）このコマンドはふたつの形態をとります。ひとつは `\script[src=<filename>]` で Lua スクリプトをファイルごとに取り込み、もうひとつは `\script{...}` で、インラインの Lua コードです。

インラインで何か面白いことをやるには SILE の内部に関する知識が必要です（幸運なことにコードはそれほど複雑ではない）が、とりあえず手始めに、Lua 関数 `SILE.typesetter:typeset(...)` を使ってみましょう。これはページにテキストを加えます。`SILE.call("...")` は SILE コマンドを呼び出し、`SILE.typesetter:leaveHmode()` は現在のパラグラフを終了し、テキストを出力します。例として、

---

```
\begin{script}
  for i=1,10 do
    SILE.typesetter:typeset(i .. " x " .. i .. " = " .. i*i .. ". ")
    SILE.typesetter:leaveHmode()
    SILE.call("smallskip")
  end
\end{script}
```

---

## SILE コマンド

は以下を出力します。

---

```
1 x 1 = 1.  
2 x 2 = 4.  
3 x 3 = 9.  
4 x 4 = 16.  
5 x 5 = 25.  
6 x 6 = 36.  
7 x 7 = 49.  
8 x 8 = 64.  
9 x 9 = 81.  
10 x 10 = 100.
```

---



# Chapter 5

## SILE パッケージ

SILE には付加的な機能を提供する様々なパッケージが付属しています。事実、SILE の実際の「中核」(“core”)となる機能はかなりコンパクトで拡張性に富み、ほとんどの重要な機能はアドオンパッケージとして提供されています。SILE パッケージは Lua プログラミング言語で書かれており、新たなコマンドを定義したり、SILE の動作を変更したり、実際のところ、Lua でできることは何でもできます。

先に述べたとおり、パッケージのロードは`\script`コマンドで行われ、これは Lua コードを実行します。規約として、パッケージはあなたの作業ディレクトリ、または SILE のインストールディレクトリの、`packages/`サブディレクトリに入れることになります。例えば、すぐ後に述べる`grid`パッケージは通常は`/usr/local/lib/sile/packages/grid.lua`にあります。これをロードするには、

---

```
\script[src=packages/grid]
```

---

とします。

---

### SILE のパス検索

SILE は様々なディレクトリを検索します。まずはカレントディレクトリ、次に、もし環境変数`SILE_PATH`が設定されていれば、SILE はそのディレクトリを、そして標準的なインストールディレクトリ、`/usr/lib/sile`や`/usr/local/lib/sile`。TeX とは異なり、SILE はサブディレクトリを再帰的に検索しません。このためもしあなたが、あなたのマクロやクラス、パッケージファイルなどをサブディレクトリに置いたら、あなたはその完全な相対パスを指定しなければなりません。

---

## 5.1 image

テキスト以外にも SILE は画像を挿入することができます。

`image`パッケージをロードすることで、HTML のそれと同様の、`\img`コマンドが使えるようになります。`img`は次のふたつのパラメータを取ります。`src=...`は画像ファイルへのパスで、またオプションとして、`height=...`あるいは`width=...`パラメータを表示する画像のサイズとして指定します。もしもサイズが指定されなければ、画像はその「自然な」<sup>1</sup>ピクセルサイズで表示されます。

---

デフォルトの `libtexpdf` バックエンドでは、画像は JPEG、PNG、EPS、PDF フォーマットがサポートされます。`Pango/Cairo` バックエンドでは PNG のみです。

1. 訳注：原著は‘natural’です。どういう意味なのかよく分かりません。

---

それでは 200x243 ピクセルの画像を、`\img[src=documentation/gutenberg.png]`で表示させてみましょう。



それと、(それぞれ) `\img[src=documentation/gutenberg.png,width=120px]`、  
`\img[src=documentation/gutenberg.png,height=200px]`、  
`\img[src=documentation/gutenberg.png,width=120px,height=200px]`です。



画像は、あたかも非常に大きな文字のように、テキストのベースラインに沿って配置されることに注意してください。

## 5.2 rules

`rules` パッケージは罫線を描画します。これはふたつのコマンドを提供します。

まずは `\hrule` で、これは与えられた太さ（高さ）と長さ（幅）の線分を描画します。

罫線は他のテキストと全く同じように取り扱われます。このため、パラグラフの途中で、このように——描画することも可能です。（これは`\hrule[width=20pt, height=0.5pt]`で生成されました）

画像と同じく、罫線はテキストのベースラインに沿って配置されます。

`rules`パッケージで提供される、ふたつ目のコマンドは`\underline`で、これは下線を引くものです。

---

「下線」は文書作成上の良い習慣とはとても言えません。決して使わないでください。

---

（これは`\underline{決して}`で生成されました）

## 5.3 color


`color`パッケージは、テキストや罫線の色を一時的に変えるためのものです。このパッケージはひとつのパラメータを取る`\color`コマンドを提供します。パラメータは`color=<color specification>`で引数として与えられたものをその色で描画します。色の指定法はHTMLと同じです。16進数値`x`を使ったRGB値で`#xxx`または`#xxxxxx`の形式（例えば、`#000`は黒で`#fff`は白、`#f00`は赤）、またはHTMLとCSSにおける名前付きの色指定が可能です。

---

HTMLとCSSの名前付き色のリストは<http://dev.w3.org/csswg/css-color/#named-colors>にあります。

---

例として挙げると、このテキストは`\color{color=red}{...}`での出力です。

罫線の例も挙げておきましょう。`\color{color=#22dd33}`: 

## 5.4 rotate

`rotate`パッケージは回転機能を提供します。これにより、`rotate=<angle>`をフレーム宣言に加えることで、フレーム全体を回転させることができます。また、`\rotate[angle=<angle>]{...}`コマンドであらゆるものを回転させることができます。ここで`<angle>`は角度を「度」で表したものです。

回転される描画物はボックスに配置され、回転されます。その高さや幅は計算され、組版のために通常の水平リストに加えられます。このため、回転される内容の周囲には余白が確保されます。このことを理解するために実例をいくつか示しましょう。 here is some text rotated by *ten*, *twenty* and *forty* degrees.

前行は以下のコードで生成されました。

---

```
here is some text rotated by
\rotate[angle=10]{ten}, \rotate[angle=20]{twenty} and \rotate[angle=40]{forty}
degrees.
```

---

## 5.5 features

Chapter 3 で述べたように、SILE は自動的にフォントで定義されたリガチャを適用します。これらのリガチャはフォントファイルのフィーチャ<sup>2</sup>テーブルで定義されています。リガチャ（複数のグリフがひとつのグリフとして表示される）の他にも、フィーチャテーブルでは様々なグリフ置換が宣言されています。

**features** パッケージは SILE があなたが選択するフォントで有効にするフィーチャを選択する機構を提供します。どのようなフィーチャが存在するかはフォントに依存します。いくつかのフォントでは、それがどのようなフィーチャをサポートしているか説明する文書が付属しています。OpenType フィーチャに関する議論はこのマニュアルの範疇を超えているのでここでは行いません。

フィーチャは `\font` コマンドのオプションでそのまのフィーチャ名を渡すことで有効・無効にできます。

---

```
\font[features="+dlig,+hlig"]... % turn on discretionary and historic ligatures
```

---

しかしながら、この方法は扱いづらく、フィーチャコードを覚えておく必要があります。**features** はふたつのコマンドを提供します。`\add-font-feature` と `\remove-font-feature` で、OpenType フィーチャへのより簡単なアクセスを実現します。インターフェイスは TeX パッケージの **fontspec** に由来しています。サポートされている OpenType フィーチャの完全な解説については、**fontspec** パッケージの文書を参照してください。<sup>3</sup>

以下に **features** パッケージで任意の (discretionary) リガチャと歴史的 (historic) リガチャを制御する例を示します。

---

```
\add-font-feature[Ligatures=Rare]\add-font-feature[Ligatures=Discretionary]
...
\remove-font-feature[Ligatures=Rare]\remove-font-feature[Ligatures=Discretionary]
```

---

## 5.6 unichar

SILE は Unicode 対応であり、その入力には UTF-8 エンコーディングです。（サポートされる Unicode の範囲に関しては、使用するフォントに依存します）Unicode で定義されたいくつかの文字はキーボードから直接入力するのは困難です。このため、**unichar** パッケージはこの問題に対処するための方法、Unicode コードを直接指定するコマンド、を提供します。**unichar** をロードすると、`\unichar` コマンドが利用可能となります。

2. 訳注：原文は単に features。OpenType などの高度な組版拡張機能のこと
3. <http://texdoc.net/texmf-dist/doc/latex/fontspec/fontspec.pdf>

---

`\unichar{U+263A} % produces ☺`

---

`\unichar`の引数がU+、u+、0xあるいは0Xで始まる場合は、それは16進数値だとみなされます。そうでなければ10進数であると仮定されます。

## 5.7 bidi

ラテン文字などの用字系では文章は左から右へ（LTR）進みます。しかしながら、いくつかの用字系、特にアラビア語やヘブライ語では、右から左に（RTL）進みます。`bidi`パッケージ、これはデフォルトでロードされます、は右から左へ書き進める場合の文書や、LTRとRTLを混在させるような文書を、正しく組むための機能を提供します。これはデフォルトで読み込まれるため、パラグラフ中でLTRとRTLテキストの両方を書くことができ、SILEは正しい順序でそれらの文字が出力されることを保証します。

`bidi`パッケージはふたつのコマンド、`\thisframeLTR`と`\thisframeRTL`、これらは現在のフレームのデフォルトの書字方向を設定する、を提供します。すなわち、もしあなたがSILEにフレームがRTLであると指示するならば、文章は右端から始まり、左に進みます。このパッケージはまた、`\bidi-off`と`\bidi-on`コマンドを提供します。bidirectionalサポートを無効にすることで、もしかしたら処理速度を向上させることができるかもしれません。

## 5.8 raiselower

もしあなたが、画像や罫線、テキストなどがベースライン上に沿って配置させてくれないならば、`raiselower`パッケージを使ってそれらを上げ下げすることができます。（`footnote`パッケージは脚注の参照番号を、上付き数字として表示するため、このパッケージを利用しています。）

これはふたつの単純なコマンドを提供します。`\raise`と`\lower`で、どちらも`height=<dimension>`をパラメータとして取ります。それぞれ引数となるものを、与えられた量だけ持ち上げたり、下げたりします。これらは行の高さや深さに影響しません。

Here is some text raised by three points; here is some text lowered by four points.

前のパラグラフは以下のコードで生成されます。

```
Here is some text raised by \raise[height=3pt]{three points}; here is
some text lowered by \lower[height=4pt]{four points}.
```

## 5.9 grid

SILEは通常、以下に示すふたつのルールに従って行と行の間のスペースを決めます。

- SILEはベースライン間が、`baselineskip`と呼ばれるある固定された距離となるように、連続したふたつの行の間にスペースの挿入を試みる。

- もしも最初のルールによる結果によって、行間の距離が2ポイント以下になるならば、それを2ポイントとなるように強制する。（この量は`lineskip`の設定で調整可能）

2番目のルールは、前の行が大きなディセンダー（例えば、g、q、j、pなどの文字が含まれる）場合に、高いアセンダーを持つ（k、l、と大文字など）次の行とぶつかるのを防ぐためにあります。

これに加えて、`baselineskip`はある量の「伸張性」(‘stretch’)を持ちます。これはページ分割処理を最適化させるために役に立つ場合に、行間を広げたり、同様にパラグラフ間の距離を伸縮させたりします。

これらのルールの組み合わせにより、行はページの様々な位置で開始することになります。

これとは別の組版の流儀では、行は規則正しいグリッド上の決まった位置で始まることを要求します。いくらかの人々はグリッド上の組版によって生じる、ページの「カラー」(‘color’)を好みます。そしてこの手法は、しばしば非常に薄い紙に印字するときに、インクが裏から滲まないように工夫する目的として、行の位置をきっちり揃えるために用いられます。次の例を比べてみてください。左のものは行が紙の両面で同じ位置にくるようにしたもので、右はそうになっていないものです。

loremol	lorem
ipsumqi	ipsum
dolorob	dolor
sit amet	sit amet

`grid`パッケージはSILEの組版エンジンが、上記ふたつのルールを適用しないように動作を変更させます。その結果、行は常にグリッド上に整列され、パラグラフ間のスペースなどは常に、グリッド上行が規則正しく並ぶように調整されます。パッケージをロードすることで、ふたつの新たなSILEコマンドが追加されます。`\grid[spacing=<dimension>]`と`\no-grid`です。最初のものはグリッド上の組版を、それが実行された時点以降から、有効にして、次のものはそれを無効にします。

このセクションのはじめで、`\grid[spacing=15pt]`によって、15ポイントのグリッドが設定されています。グリッドが有効になったときの例文を示しましょう。

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.

そして以下は、`\no-grid`にした後のものです。

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.

## 5.10 verbatim

**verbatim**パッケージはプログラムのコードを引用したりするなど、フォーマットが重要な文章を表示したりする際に役に立ちます。これは SILE の設定を、左揃え、ハイフネーションなし、インデントなしの規則的なスペーシングとなるようにします。これはまた、SILE にスペースの数を勝手に減らしたりしないように伝え、また等幅フォントを使うように設定します。

---

この名前にも関わらず、**verbatim**は SILE が特殊文字を扱うやり方を変えません。あなたは**verbatim**中でもバックスラッシュと波括弧をエスケープする必要があります。例えば、\\のように。

---

それでは verbatim 環境の例を示してみましょう。

```
function SILE.repl()
  if not SILE._repl then SILE.initRepl() end
  SILE._repl:run()
end
```

verbatim 環境の中で使用するフォントを指定したければ、**verbatim:font**コマンドで再定義することができます。この文書では、

```
<define command="verbatim:font">
  <font family="Consolas" size="10pt"/>
</define>
```

です。

## 5.11 他のパッケージにより利用されるパッケージ

これらに加えて、おそらくあなたが直接は利用しないであろうパッケージもあります。それらのパッケージは、他のパッケージやクラスにより基本的な機能を提供するという目的のために存在します。例えばbookクラスは、他の補助的なパッケージからの機能で構成されています。

### 5.11.1 footnotes

例えば、bookクラスでは、**\footnote**コマンドで脚注を加えることができることをみました。このコマンドは実際には、**footnotes**パッケージにより提供されています。bookクラスはこのパッケージをロー

ドし、どこに脚注を置くべきか伝え、そして`footnotes`パッケージは脚注を整形し表示させます。これは他にも、以下に述べる数々のパッケージを利用しながら行われます。

### 5.11.2 counters

SILE の様々な部分、例えば`footnotes`パッケージや章・節用のコマンド、はカウンタを利用します。現在の脚注番号、章番号、などに使うためです。`counters`パッケージは、カウンタを設定し、カウンタを増加させ、表示させたりすることに使えます。これは以下のようなコマンドを提供します。

- `\set-counter[id=<counter-name>,value=<value>]` — `<counter-name>` という名前のカウンタを与えられた値で設定します。
- `\increment-counter[id=<counter-name>]` — `\set-counter` と同様に、しかし `value` パラメータがなければ、カウンタを 1 だけ増加させます。
- `\show-counter[id=<counter-name>]` — これはカウンタの値を、宣言された表示形式で整形し、表示します。

---

カウンタパッケージのすべてのコマンドはオプションとして、`display=<display-type>` パラメータを取り、これはカウンタの表示形式を設定します。

可能な表示形式は、デフォルトで `arabic`、アルファベットのカウンタとして `alpha`、小文字のローマ数字 `roman`、そして大文字のローマ数字 `Roman` です。

---

例えば、次のような SILE コードは、

---

```
\set-counter[id=mycounter, value=2]
\show-counter[id=mycounter]
\increment-counter[id=mycounter]
\show-counter[id=mycounter, display=roman]
```

---

以下のようになります。

---

```
2
iii
```

---

### 5.11.3 pdf

`pdf` パッケージは（基本的な）PDF リンクや目次の機能を実現します。これは 3 つのコマンドを提供します。`\pdf:destination`、`\pdf:link`、そして `\pdf:bookmark` です。

コマンド `\pdf:destination` はリンク先（ターゲット）を生成します。これはパラメータとして `name` を取り、リンク先を特定するための一意的な名前を指定します。文書中のある場所にリンクを張るには、`\pdf:link[dest=name]{内容}` を使います。

もし、`pdf` パッケージが `tableofcontents` パッケージのあとにロードされたら（例えば、`book` クラスを用いた文書の中）、PDF 文書はしおり（アウトライン）付のものとなります。



### 5.11.4 frametricks

最初の章で述べたように、SILE はページのどの部分にテキストを置くか指定するために、フレームを使います。frametricksパッケージはパッケージの著者に、フレームを扱うための数々のコマンドを提供します。

とにかく有用なのはshowframeです。これは出力エンジンに、フレームを線で囲いラベルを付けるように指示します。これはオプションでパラメータid=<frame id>をとります。もしこのオプションが与えられなければ、現在のフレームが使用されます。もし ID がallであれば、現在のクラスの中で宣言されたすべてのフレームが表示されます。

コマンド\breakframeverticalは現在のフレームを、与えられた地点で上下2分割にします。現在のフレームが ID mainを持つとしましょう。フレームが分割されると、mainは上のフレーム（コマンド挿入以前）となり、下のフレーム（コマンド挿入以降）はmain\_という ID となります。このパラグラフの先頭では、\breakframeverticalコマンドを実行しています。そしてここで、コマンド\showframeを実行してみましょう。見てわかる通り、現在のフレームはcontent\_ で、ちょうどこのパラグラフの開始位置で始まります。

同様に、\breakframehorizontalコマンドは、フレームを左右に分割します。このコマンドは必須でない引数として、offset=<dimension>をとり、これはどの位置でフレームを分割するかを指定します。もしこれが与えられなければ、フレームはその行の現在地で分割されます。

コマンド\shiftframeedgeは、現在のフレームを左右に再配置します。これはleft=と（または）right=パラメータをとり、パラメータの値は正か負の長さです。このコマンドはフレームの先頭で使用されなければなりません。それはこのコマンドが組版エンジンを再初期化するからです。

それらをすべて組み合わせた\floatコマンドは、現在のフレームを分割し、フロートオブジェクトを保持する小さなフレーム（この文章のはじめのドロップキャップのように）を生成し、文章を周囲のフレームに流し込み、文章がフロートオブジェクトの脇を過ぎ去れば、フレームを元のように戻します。floatコマンドは必須でないふたつのパラメータを取ります。bottomboundary=<dimension>とrightboundary=<dimension>で、フレームの周りに余白を付与します。このパラグラフのはじめでは、コマンド\float[bottomboundary=5pt]{\font[size=50pt]{こ}}が実行されています。

最後に、本文とは分離された、サイドバーのようなフレームを定義する方法を示しましょう。後の章でどのようにしてそれが実現されるか見るでしょう。<sup>4</sup> frametricksは\typeset-intoコマンドを提供します。これはテキストを指定されたフレームに描画します。

---

```
\typeset-into[frame=sidebar]{ ... frame content here ... }
```

---

### 5.11.5 insertions

footnotesパッケージは付加的なもの（脚注の内容）を内容として取り、現在のフレームを縮小し、それを脚注フレームに挿入します。このような作業はinsertionsパッケージの力で実現されています。

これはユーザに見える SILE コマンドを提供しませんが、他のパッケージに Lua 機能を提供します。<sup>4</sup> 脚注：原文は We'll see how to do that in a later chapter, but this raises the obvious question: if they're not part of the text flow, how do we get stuff into them?

TeXnician たちは、これが SILE のコアではなく、外部のアドオンパッケージとして実装されていることに興味を持つかもしれません。

### 5.11.6 twoside

Chapter 4 で述べた **book** クラスは左右の対となるページマスタ<sup>5</sup>を設定します。**twoside** パッケージはヘッダやその他の位置の入れ替えなどを行います。これは一般ユーザからは利用されません。

### 5.11.7 masters

The masters functionality is also itself an add-on package. It allows a class to define sets of frames and switch between them either temporarily or permanently. It defines the commands `\define-master-template` (which is pattern on the `\pagetemplate` function we will meet in chapter 8), `\switch-master` and `\switch-master-one-page`. See `tests/masters.sil` for more about this package.

### 5.11.8 infonode

---

This package is only for class designers.

---

While typesetting a document, SILE first breaks a paragraph into lines, then arranges lines into a page, and later outputs the page. In other words, while it is looking at the text of a paragraph, it is not clear what page the text will eventually end up on. This makes it difficult to produce indexes, tables of contents and so on where one needs to know the page number for a particular element.

To get around this problem, the **infonode** allows you to insert information nodes into the text stream; when a page is outputted, these nodes are collected into a list, and a class's output routine can examine this list to determine which nodes fell on a particular page. **infonode** provides the `\info` command to put an information node into the text stream; it has two required parameters, **category=** and **value=**. Categories are used to group similar sets of node together.

As an example, when typesetting a Bible, you may wish to display which range of verses are on each page as a running header. During the command which starts a new verse, you would insert an information node with the verse reference:

---

```
SILE.Commands["info"](category = "references", value = ref , )
```

---

During the `endPage` method which is called at the end of every page, we look at the list of “references” information nodes:

---

```
local refs = SILE.scratch.info.thispage.references
local runningHead = SILE.shaper.shape(refs[1] .. " - " .. refs[#refs])
SILE.typesetNaturally(rhFrame, runningHead);
```

---

5. 訳注 : 適切な訳語がわかりません。

---

### 5.11.9 inputfilter

The **inputfilter** package provides ways for class authors to transform the input of a SILE document after it is parsed but before it is processed. It does this by allowing you to rewrite the abstract syntax tree representing the document.

Loading **inputfilter** into your class with `class:loadPackage("inputfilter")` provides you with two new Lua functions: **transformContent** and **createCommand**. **transformContent** takes a content tree and applies a transformation function to the text within it. See `examples/inputfilter.sil` for a simple example, and `packages/chordmode.sil` for a more complete one.