Rakefile for LaTeX compile

Ippei KISHIDA

Last-modified: 2017/05/09 12:25:47.

目次

1 生成の流れと依存関係

(English version will be written in §2.)

Fig. 3 に生成の流れを示す。tex ファイルから dvi を生成し、dvi から pdf を生成を生成するというのがメインのプロセス。dvi を生成する際に eps が必要になる。eps は graphviz 系、 gnuplot、 ruby など様々な方法で生成される。LaTeX は印刷を最終目的とするものなので、ベクターデータである eps での出力が主となる。とはいえ eps だと閲覧が不便で、qiv などで簡便に表示するために png があったら便利。ということで LaTeX で扱う画像は、ベクターデータ (eps) が主であり、ラスターデータ (png) が補助的という位置付けとなる。[*1]

[*1] この点が Markdown/HTML と決定的に異なる。Markdown/HTML はディスプレイで表示するの

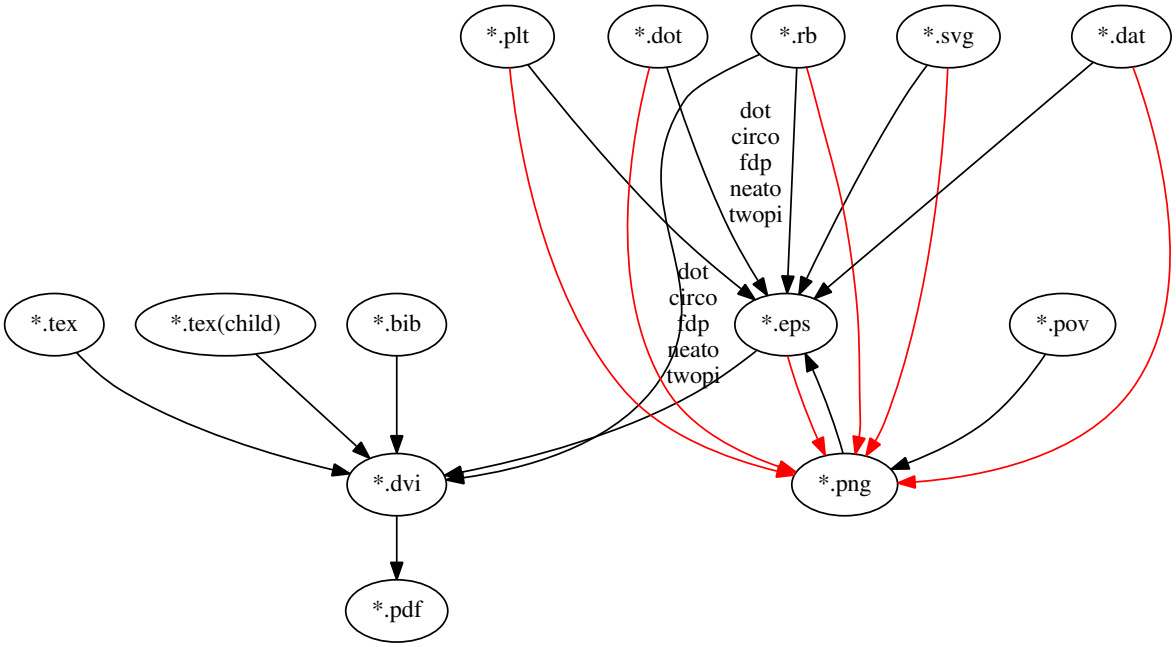

図 1　(fig010) 生成の流れ。*.rb は gnuplot をインクルードしてグラフを描画するためのスクリプトと、プログラミング文書でコードを埋め込むためのものがありうる。*.tex は、それぞれの tex ファイルがトップレベル tex ファイルとして機能する可能性がある。*.dat は ruby, gnuplot でのデータファイルとしての使用を想定している。pdf を作るために必要な流れを黒、補助的扱いの png を作るためのプロセスを赤としている。

■tex の問題　本来の LaTeX では、ディレクトリ内のあらゆる tex ファイルが dvi を生成する可能性がある。ファイルの中身を見て include 解析などすれば機械的に依存関係を把握できる筈だが、ファイル名のみからはその関係がわからない。しかも C のようにとりあえず単体が完結してコンパイルできるということは保証されておらず、documentclass 宣言や document 環境が含まれていなければコンパイルできない。ファイル名のみから判断される機械的なコンパイル動作を作るのは煩雑。ソースコード解析してまでやるべきことではなかろう。

　これらを解決するために自分ルールを導入する。1 つのディレクトリは 1 つの pdf ファイルを生成するためのものと位置付ける。latex で処理するのは唯一 main.tex のみとする。ただし他の `*.tex` ファイルも依存する可能性があり更新時刻チェックの対象とする。

を目的とするためラスターデータ (png) が主であり、ベクターデータ (eps) が補助的という位置付けになる。

同じディレクトリにあるのだから include などで取り込まれるのだろう、ということを前提とする。

■pdf の名前　「latex で処理するのは唯一 main.tex のみとする」ため、素直に作ると main.dvi から main.pdf を生成することになる筈だが、pdf を別の場所に移したりメール添付する場合には内容を表すファイル名にその都度変更すべきという事になる。

自分ルールとして「1 つのディレクトリは 1 つの pdf ファイルを生成するためのものと位置付ける」ことから、カレントディレクトリの名前を用いたファイル名で PDF を作ることにする。カレントディレクトリが foo ならば、foo.pdf である。

■pov の問題　pov の出力が png であり、eps には png を経由して生成するものであるということが問題。そのため、svg などでは eps から png を生成するが、pov では png から eps を生成することになり、ファイルタイプ (拡張子) だけからはどちらがどちらに依存しているか判別がつかないことになる。rake を実行するたびに相互に生成しあうこともありうる。

これを解決するために、rake の処理としては pov → eps に直接変換するように見えるようにする。すなわち、内部的には png を生成するものの中間ファイルを削除する。

■rb の問題　rb ファイルが本当に画像を生成させるためのものかも問題。プログラミングの教科書だと LIST 環境で表示するためのコードを記述していることもありうる。

この解決として、以下の自分ルールを課すことにする。

- 画像用の rb ファイルは fig から始まるファイル名にする。
- LIST 環境で使うものは list から始まるファイル名にする。
- これら以外の rb ファイルは無視する。

■dot の問題　dot は graphviz の dot 形式ファイル。ただし、 dot, circo, fdp, neato, twopi のいずれを使うかで描画され方が違う。絵の意図としてはその都度どれかを選択する筈だが、ファイル内でそれを指定できない。(dot ファイルは基本的に論理的な関係のみを記述する。) よって、ファイルごとに Rakefile に記述していく手間が生じることになる。

この解決として、circo, fdp といった拡張子だったらそれらのコマンドを使うというルールを自分ルールとして採用。

自分ルールを含めて改善した生成の流れ

上記の自分ルールの適用によって整理した生成の流れを Fig. 2 に示す。このフローに沿って Rakefile を作成した。

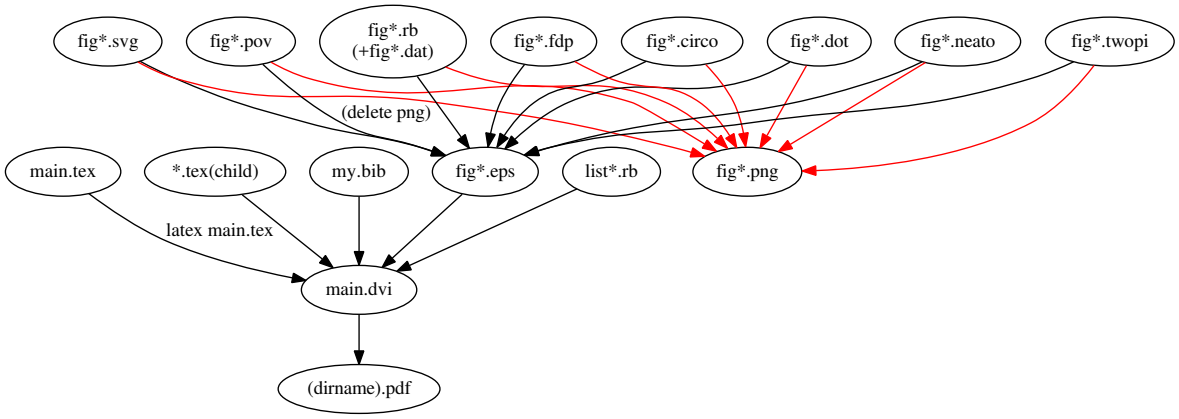

図 2　(fig015) 自分ルールを含めて改善した生成の流れ。png を作るプロセスを赤エッジで表している。

2　Compiling process and dependency(English Version of §1)

The flow of generation is shown in Fig. 3. The main process is to generate DVI from the TEX file and generate pdf from DVI. You need EPS files when generating DVI using figures. EPS files are generated by various methods such as graphviz, gnuplot, and ruby. The final objective of LaTeXis printing. Principal figure format should be EPS as vector data. However, PNG is more convenient to display and browse than EPS. [*2]

■TEX problem　In the original LaTeX, every TEX file in a directory may generate DVI. Although you can grasp dependency mechanically by reading and analyzing the contents of the file, it cannot be determined from the file name alone. Moreover,

[*2] This point is definitely different from Markdown/HTML system. Markdown/HTML is intended to show on the electronic display. Raster data (PNG) is the main, Vector data (EPS) is not needed.

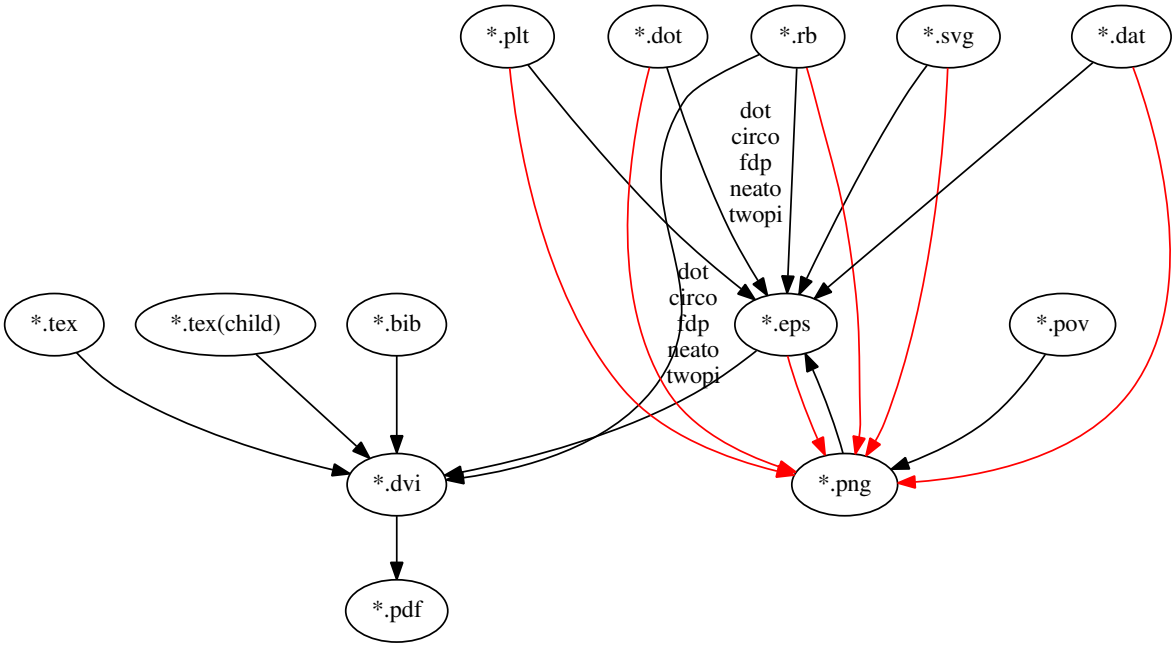

図 3　(fig010) Generation flow from TEX to PDF. Each TEX file, *.tex, may become a top level TEX file and an included TEX document. Ruby script file, *.rb, may be embedding code in a list environment, and may be a gnuplot script for drawing a graph. Data file, *.dat, is assumed to be used as a data file in ruby and gnuplot. The flow necessary to make pdf is black, the process to make png which is supplementary treatments is red.

not all tex file can be compiled unlike C. Documentclass declaration and document environment must be included in an entry point TEX file. It is cumbersome to make a mechanical compilation operation judged only from the file name. It should not be done until analyzing the source code because of the complication.

I introduce my own rules to solve these problems. One directory is positioned to generate one PDF file. Only `main.tex` is compiled by latex. However, other `*.tex` files may depend on them, and they are used for the update time check. It is based on the premise that it will be included because they are in the same directory,

■PDF name　”Only `main.tex` is compiled by latex.” Under the rule, `main.tex` will generate `main.dvi` and `main.pdf`. But when the PDF is transferred to another place or attaching e-mails, it should be renamed each time. As my rule ”Position one directory as one for generating one pdf file”. Therefore, we will make a PDF with the

file name using the name of the current directory. If the current directory is `foo/`, it is `foo.pdf` in the `foo/` directory.

■POV problem Many program can output EPS and PDF. But POV-Ray can generate PNG but cannot EPS. So EPS of POV-Ray figures are generated from PNG. It is a problem that it is difficult to determine the first of EPS and PNG from mechanically.

In order to solve this problem, a process is imitated like to make EPS directly from POV. That is, internally, it creates png but deletes the intermediate file.

■RB problem It is also a problem whether the RB file is really for generating images. The ruby code may be LIST environment material for programming textbooks. To solve this, we will impose our own rule.

- The RB file for images should be filenames starting with `fig`, e.g., `fig010.rb`.
- The one used in the LIST environment is the file name starting from the `list`, e.g., `list010.rb`.
- Ignore other RB files.

■DOT problem DOT is a file for graphviz. However, the figure is generated by various commands, e.g., dot, circo, fdp, neato, twopi, etc. As the intention of the picture, you should choose something every time, it can not be specified in the file. Therefore, it takes time and effort to describe it in the Rakefile for each file.

As this solution, I assume new extensions like `.circo, .fdp`, etc. We adopt the rule of using those commands as our rule.

Improved generation flow including my own rules

The flow of generation sorted out by applying the above rule of self is shown in Fig. 2. I made a Rakefile along this flow.

3 Requirement

- latex
- latexmk
- bibtex

- dvipdfmx
- gnuplot
- graphviz
- imagemagick
- inkscape
- povray
- (ruby/gnuplot)

4 Section for test

4.1 Other *.tex

「この文は input1.tex 内に記述されている。」

”This sentense is written in input1.tex”

4.2 list

List 1 (list010.rb)list010.rb

```
1 #! /usr/bin/env ruby
2
3 puts "Hello, world!"
```

4.3 Figures

Conversion from ’dot’ to ’eps’ can be confirmed at Fig. 3.

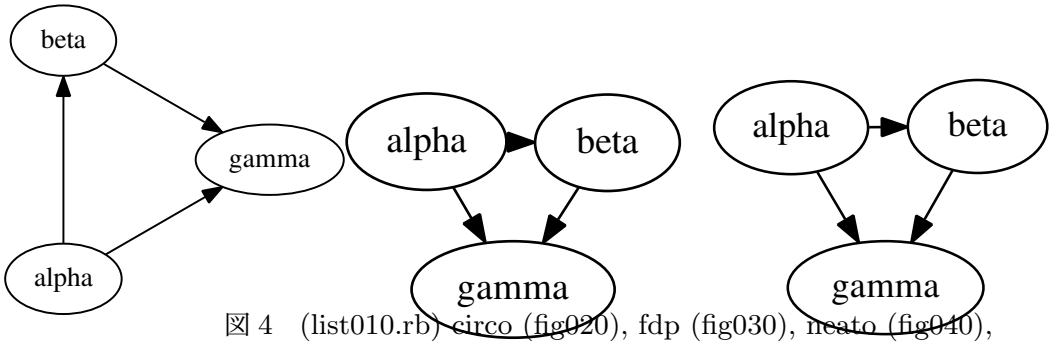

図 4　(list010.rb) circo (fig020), fdp (fig030), neato (fig040),

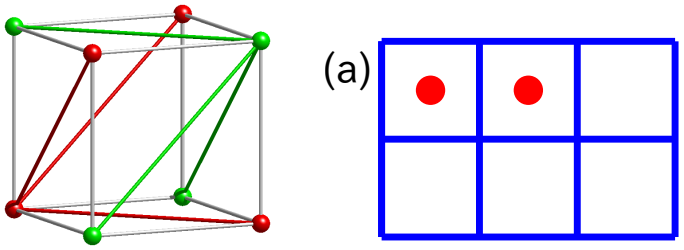

図 5　pov → eps (fig070), svg → eps (fig080)。

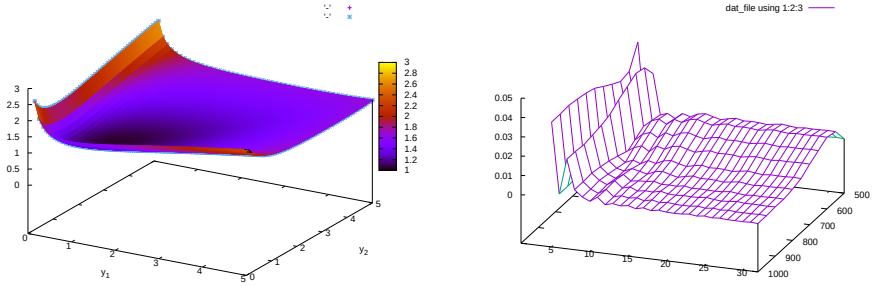

図 6　(list010.rb) rb + dat → eps (fig090), plt + dat → eps (fig100).