

LuaL^AT_EX with Debian

環境構築と作業手順

© [ru_museum](#)(GitHub)

2024 年 9 月 26 日

目次

1	Debian における Lua ^A T _E X の使用	1
1.1	環境構築	1
1.2	作業環境	1
1.2.1	gedit	1
1.2.2	VSCode	3
2	SVG 画像の表示	4
3	PythonT _E X の利用	5
4	Bib ^A T _E X + biber の導入	6
5	編集作業の流れ	7
6	コードサンプル	8
6.1	Build LuaLatex	8
6.2	Open PDF	9
7	TIPS	10
7.1	python3.12 へのアップグレードにおける不具合	10

1 Debian における Lua_AT_EX の使用

1.1 環境構築

- これは Debian 標準添付のエディタ **gedit** を使用した作業手順です。

- インストールパッケージ

※ texlive-full は容量大なので必要なもののみを選択。

texlive

texlive-lang-japanese

texlive-luatex

texlive-bibtex-extra % BibLaTeX

texlive-extra-utils % PythonTeX

inkscape % SVG 画像の表示

- PDF Viewer (Atril、evince 等) が必要となります。

Lua_AT_EX は直接 PDF を作成するので pdftex 等は不要です。

1.2 作業環境

1.2.1 gedit

1. プラグインパッケージをインストール

```
# apt-get install gedit-latex-plugin
```


参照: [TeX Wiki gedit](https://texwiki.texjp.org/?gedit) (<https://texwiki.texjp.org/?gedit>)

2. プラグイン設定

「設定」→「プラグイン」において以下にチェックを入れます:

LaTeX Plugin、外部ツール、組み込み端末、コードスニペット、
コードコメント

3. 外部ツール設定

「Manage External Tools(外部ツールの設定)」→「外部ツールの管理」で新規ツールを登録します(画面左下の  をクリックします)。

ツール 1: Build LuaLatex

sh スクリプト: `#!/bin/sh`

`lualatex $GEDIT_CURRENT_DOCUMENT_NAME`

ショートカットキー: **Alt + F5**



図 1 外部ツールの設定: Build LuaLatex

- 「ショートカットキー」は設定が重複していなければ自由に登録出来ます。
- 設定は ” /.config/gedit/tools/build-lualatex” に登録されます。
- ビルド (Alt + F5) は処理状況が gedit 画面下部の **Tool Output** に表示され **Done** と表示されれば正常終了です。

ツール 2 : Open PDF

sh スクリプト :

```
#!/bin/sh
```

```
FILENAME="$GEDIT_CURRENT_DOCUMENT_NAME"
```

```
PDF_FILE="`basename "$FILENAME" .tex`.pdf"
```

```
# Open with PDFViewer(atril or evince)
```

```
atril $PDF_FILE
```

ショートカットキー : **Shift + Alt + P**

- 初回もし PDF ファイルが生成されていれば **Shift + Alt + P** で開くことが出来ます。
- 2 度目のビルド以降は変更が自動で反映されます。

4. Build LuaLatex の修正 : biber

- ビルド時 **biber** 用の **.bcf** ファイルが既に生成されていれば同時にコンパイルする様に **Build LuaLatex** を修正します。
- この設定は「BibLaTeX + biber」導入済の場合です。
⇒ 詳しくは「4 BibLaTeX + biber」をご覧ください。
- **biber** は BibLaTeX 用のコンパイラです。

ツール名: **Build LuaLatex**

sh スクリプト:

```
#!/bin/sh
# .tex ファイル名
FILENAME="$GEDIT_CURRENT_DOCUMENT_NAME"
# .bcf ファイル名 (BibLatex)
FILE_BCF="`basename "$FILENAME" .tex`.bcf"
# .tex をビルド
lualatex $FILENAME
# biber での .bcf コンパイル (BibLatex)
if [ -e $FILE_BCF ]; then
    echo "BCF File をコンパイルしています -----"
    # "File exists."
    biber $FILE_BCF
    echo "BCF File のコンパイルを終了しました -----"
    lualatex $FILENAME
fi
```

1.2.2 VSCode

1. プラグイン LaTeX Workshop をインストール
2. settings.json に追記します。

```
"latex-workshop.latex.recipes": [
{
    "name": "lualatex",
    "tools": ["lualatex"]
}
],
"latex-workshop.latex.tools": [
{
    "name": "lualatex",
    "command": "lualatex",
    "args": [
        "--cmdx",
```

```

        "-file-line-error",
        "-synctex=1",
        "-interaction=nonstopmode", % エラー中断の回避 (重要)
        "-halt-on-error",
        "\%DOC\%"
    ],
    "env": {}
}
]

```

2 SVG 画像の表示

※ "Inkscape" のインストールが必要です。

1. Build LuaLatex に追加します。

```

# Inkscape export で変換
lualatex \textbf{-shell-escape} $FILENAME

```

2. 画像の表示

```

\usepackage{svg} % 必須
\usepackage{float} % [H]
\begin{document}
  \begin{figure}[H]
    \centering
    \includesvg[width=6cm]{./filename.svg}
    \caption{figure 1}
  \end{figure}

```

3. Example SVG:



図2 L^AT_EX Logo: ウィキメディア・コモンズ

3 Python_TE_X の利用

- pythontex により Python スクリプトを動作させます。
※ Ruby や Javascript その他も可能。
- Python_TE_X は texlive-extra-utils に含まれています。
- ビルドは Lualatex ⇒ pythontex ⇒ Lualatex で行います。
- 初回のビルド lualatex .tex を行うとファイル **.pytxcode** が生成されます。
- ビルド時 **pythontex** 用の .pytxcode ファイルが生成されていれば Lualatex のコンパイルが動作する様に「外部コマンド」に追加します。

ツール名: **Build LuaLatex**

sh スクリプト追加部分:

```
FILE_PYTHON=`basename "$FILENAME" .tex`.pytxcode"

# PYTHONTEX
if [ -e $FILE_PYTHON ]; then

    echo "PYTHONTEX File をコンパイルしています -----"
    # "File exists."
    pythontex $FILE_PYTHON

    echo "PYTHONTEX File のコンパイルを終了しました -----"

    lualatex $FILENAME

fi
```

- 「Alt + F5」のビルドで Python スクリプトが PDF へ反映されます。

4 Bib \LaTeX + biber の導入

- Debian ではパッケージ texlive-bibtex-extra に含まれます。
/usr/share/texlive/texmf-dist/tex/latex/biblatex
- biber は参考文献処理コンパイラです。

【使用例】

```
\usepackage[
    backend=biber,
    bibstyle=ieee,
]{biblatex}
\nocite{*}
\addbibresource{./lib/sample.bib} % .bib データの読込
\begin{document}
    \printbibliography[title=参考文献]
```

biber でのコンパイルの流れ

```
backend=biber の設定で .bcf ファイルが生成
lualatex test.tex    \% 初回ビルド
biber test.bcf        \% biber でコンパイル
lualatex test.tex    \% 再ビルド
```

- 外部ツール **Build Lua \LaTeX** に登録し、.bcf ファイルが存在していれば biber を動作させます。⇒参照：1.2.1 4. Build Lua \LaTeX の修正: biber

データファイル (.bib) の作成

※「文献参照名」は重複するとエラーとなります。

% 書籍

```
@book{
    文献参照名,
    author = "著者名",
    title = "タイトル",
    year = "出版年",
    publisher = "出版社",
}
```



```
% 小冊子
@booklet{
  文献参照名,
  author = "著者名",
  title = "タイトル",
  publisher = "出版社",
  note = "",
  month = "",
  year = "出版年",
  type= ""
}
```

5 編集作業の流れ

1. .tex ファイルを編集保存し、「Alt + F5」でビルドします。
2. .pdf, .aux, .out, .log (biblatex: .bcf, .bbl, .blg) が生成されます。
3. 「Shift + Alt + P」で PDF ファイルを開きます（初回のみ）。
4. .tex ファイルを再編集保存し、再ビルドします。
5. PDF ファイルに更新が反映されます。

※「Alt + F5」と「Shift + Alt + P」の設定は **1.2.1 3. 外部ツール設定**を参照して下さい。

6 コードサンプル

- 以下は最終的な動作コードのサンプルです。

6.1 Build LuaLatex

```
#!/bin/sh

FILENAME="$GEDIT_CURRENT_DOCUMENT_NAME"
FILE_BCF=`basename "$FILENAME" .tex`.bcf"
FILE_PYTHON=`basename "$FILENAME" .tex`.pytxcode"

# Incscape export で変換
lualatex -shell-escape $FILENAME

# biber : .bcf ファイルが既に生成されていれば
if [ -e $FILE_BCF ]; then

    echo "BCF File をコンパイルしています -----"
    biber $FILE_BCF

    echo "BCF File のコンパイルを終了しました -----"
    lualatex $FILENAME

fi

# PYTHONTEX : ファイルが既に生成されていれば
if [ -e $FILE_PYTHON ]; then

    echo "PYTHONTEX File をコンパイルしています -----"
    pythontex $FILE_PYTHON

    echo "PYTHONTEX File のコンパイルを終了しました -----"

    lualatex $FILENAME

fi
```

6.2 Open PDF

```
#!/bin/sh
FILENAME="$GEDIT_CURRENT_DOCUMENT_NAME"
PDF_FILE=`basename "$FILENAME" .tex`.pdf

# PDFViewer: evince, atril
atril $PDF_FILE
```

7 TIPS

7.1 python3.12 へのアップグレードにおける不具合

- python3 (3.12.6-1) へのアップグレードにより **gedit-latex-plugins** において不具合^{*1}が生じています (2024-09-16)。
- これは **gedit-latex-plugins** 側で何れ修正されると思いますが、もし問題が生じた場合は下記の対処法を試して下さい。

処理内容

```
python3 (3.12.6-1) を設定しています ...
running python rtupdate hooks for python3.12...
/usr/lib/gedit/plugins/latex/latex/actions.py:254: SyntaxWarning: invalid escape sequence '\e'
  snippet_source = "\ensuremath{\mathbb{$0}}"
/usr/lib/gedit/plugins/latex/latex/actions.py:261: SyntaxWarning: invalid escape sequence '\e'
  snippet_source = "\ensuremath{\mathcal{$0}}"
/usr/lib/gedit/plugins/latex/latex/actions.py:269: SyntaxWarning: invalid escape sequence '\e'
  snippet_source = "\ensuremath{\mathfrak{$0}}"
/usr/lib/gedit/plugins/latex/latex/editor.py:146: SyntaxWarning: invalid escape sequence '\e'
  """
/usr/lib/gedit/plugins/latex/latex/expander.py:33: SyntaxWarning: invalid escape sequence '\i'
  """
/usr/lib/gedit/plugins/latex/latex/expander.py:58: SyntaxWarning: invalid escape sequence '\i'
  """
/usr/lib/gedit/plugins/latex/latex/parser.py:233: SyntaxWarning: invalid escape sequence '\e'
  """
/usr/lib/gedit/plugins/latex/latex/parser.py:509: SyntaxWarning: invalid escape sequence '\w'
  _PATTERN = compile("(TODO|FIXME)\w?\.?(?<text>.*)")
/usr/lib/gedit/plugins/latex/latex/validator.py:40: SyntaxWarning: invalid escape sequence '\['
  """
/usr/lib/gedit/plugins/latex/preferences/_init_.py:145: SyntaxWarning: invalid escape sequence '\s'
  self._re = re.compile("^\\s*%+\\s*gedit:(.*)\\s*=\\s*(.*)")
/usr/lib/x86_64-linux-gnu/gedit/plugins/externaltools/library.py:212: SyntaxWarning: invalid escape sequence '\-'
  RE_KEY = re.compile('([a-zA-Z_][a-zA-Z0-9_\\-\\.]*)\\([([a-zA-Z_0-9\\-\\.]*)\\)?$')
/usr/lib/x86_64-linux-gnu/gedit/plugins/snippets/substitutionparser.py:162: SyntaxWarning: invalid escape sequence '\s'
  match = re.match('\\\\?%s*' % self.REG_GROUP, tokens)
/usr/share/scribus/scripts/importcsv2table.py:3: SyntaxWarning: invalid escape sequence '\o'
  """
running python post-rtupdate hooks for python3.12...
```

原因

python3.12 において **escape sequence** の扱いが変更されたことに起因します。

参照：[python3.12](https://docs.python.org/3/whatsnew/3.12.html) (<https://docs.python.org/3/whatsnew/3.12.html>)

Other Language Changes

The parser now raises **SyntaxError** when parsing source code containing null bytes.

A backslash-character pair that is not a valid escape sequence now generates a **SyntaxWarning**, instead of **DeprecationWarning**. For

example, `re.compile("\\d+\\.\\d+")` now emits a **SyntaxWarning** ("**d**" is an invalid escape sequence, use raw strings for regular expression: `re.compile(r"\\d+\\.\\d+")`). In a future Python version, **SyntaxError** will eventually be raised, instead of **SyntaxWarning**.

事例 1：「」で囲まれた文字列の先頭に「r」^{*2}を記述するか、「\\」とします。

```
/usr/lib/gedit/plugins/latex/latex/actions.py:254: SyntaxWarning: invalid es-
cape sequence '\e'
```

```
snippet_source = "\ensuremath{\mathbb{$0}}"
```

修正後：r"\ensuremath{\mathbb{\$0}}" 又は "\\ensuremath{\\mathbb{\$0}}"

^{*1} SyntaxWarning: invalid escape sequence

^{*2} raw string：バックスラッシュによるエスケープシーケンスを無視し文字通りに解釈される。

事例 2：「'''」で囲まれたコメント部全体を「#」によりコメントアウトします。
(上記「\\」でも可)

```
/usr/lib/gedit/plugins/latex/latex/editor.py:146: SyntaxWarning: invalid escape  
sequence '\e'  
'''
```

```
修正後：# '''  
# (略)  
# '''
```