gedit のバージョンアップに伴う対応について]

ru_museum: GitHub)

2025 年 5 月 13 日

目次

1 概要 1

2 検証動作環境 1

3 snippets のインストール手順 1

4 externaltools への対処法 2

4.1 組み込み端末の導入 2

4.2 Latex 用ファイルのビルド 3

4.3 SOURCE 4

1 概要

gedit は最近の v48 から組込 plugin の整理を行い、Python で書かれたプラグインを除外することとしています[*1]。それは、次の理由によります。

・多言語によるメンテナンスの煩雑さ故に Python を非推奨とし C 言語による実装へと統一する。
・PyGObject がローダライブラリ (libpeas) による Python の使用を止めた。

一方かなり以前からメンテナーの不在によりメンテナンスが行われていないという問題があり、そこにも遠因があると思われます[*2]。又、GTK4 対応の「GNOME Text Editor」をデフォルト・エディタとする移行の流れもあります [*3]。

排除されたプラグインの中には、開発に重要な **externaltools** と **snippets** も含まれ作業に支障を来している次第です。旧アーカイブからの借用を試みた所、**snippets** は正常に動作しましたが **externaltools** はインストール及び設定自体までは問題ないものの実行されませんでした [*4] 。

再実装の可能性は否定されてはいませんが、ここでは一応 **snippets** のインストール方法とその動作しない **externaltools** への暫定的な対処法を示します。

2 検証動作環境

- OS: Debian GNU/Linux(sid: trixie, Kernel: x86_64 Linux 6.12.27-amd64)
- gedit: version 48.1

3 snippets のインストール手順

1. ソースのダウンロード：
   gedit-48.0.tar.xz // Index of /sources/gedit/48/(Gnome)
   https://download.gnome.org/sources/gedit/
2. 解凍し plugins ディレクトリ内の snippets を取り出します。
3. システム内に配置：meson.build はほぼ正常に動かないので使用しません。
   主な注意点
   - meson.build は削除。
   - snippet.ui を/usr/share/gedit/plugins/ui(新規作成) にコピー。
   - snippets.plugin.desktop.in を snippets.plugin に名前変更してコピー。

[*1] 基本的に v48 から全ての Python 製のプラグインを使用出来なくしています。
「PyGObject の最新バージョンは libpeas1 の Python 3 ローダーを使用不能にした（libpeas は gedit やその他のアプリ用のプラグインエンジンを提供)」「libpeas1 リリースで問題が修正される可能性は有り」としています。
No more Pythons in gedit // Gedit Technology blog。
Are Third party plugins still going to be a thing ?

[*2] Gedit Text Editor is No Longer Maintained [gedit-list] gedit is unmaintained, some thoughts

[*3] デフォルトのテキストエディターを GNOME Text Editor に変更する提案 // kledgeb

[*4] 選択肢として gedit から派生した Pluma がありますが、こちらは逆に snippets の方が動作しませんでした。

```
/usr/lib/x86_64-linux-gnu/gedit/plugins/
    └── snippets
    │    ├── __init__.py
    │    ├── appactivatable.py
    │    ├── completion.py
    │    ├──
    │    ├── ... 略 ...
    │    ├──
    │    ├── snippet.py
    │    ├── substitutionparser.py
    │    └── windowactivatable.py
    └── snippets.plugin <= /plugins/snippets/snippets.plugin.desktop.in

/usr/share/gedit/plugins/
    └── snippets
       ├── lang <= /plugins/snippets/data/lang
       │    └── snippets.lang
       ├── ui
       │    └── snippets.ui <= /plugins/snippets/snippets/snippets.ui
       ├── c.xml
       ├──
       ├── ... 略 ...
       ├──
       ├── snippets.xml
       ├── tcl.xml
       ├── xml.xml
       └── xslt.xml
```

4. gedit を再起動：「設定」⇒「プラグイン」で「Snippets」にチェックを入れます。
5. 変更及び追加は、メニューの「Manage Snippets」で出来ます。

4 externaltools への対処法

- externaltools においては設定までは問題はありませんが、スクリプトの実行段階で動作不良となっています。そこでスクリプトを使ったコマンドラインで行います。

4.1 組み込み端末の導入

1. $ apt-get install gedit-plugins
   組み込み端末の **gedit-plugin-terminal** がインストールされます [*5] 。
2.「設定」⇒「プラグイン」で「組み込み端末」にチェックを入れます。
3. gedit 画面下部に端末が表示されます。
4. 端末画面でマウスの右クリックから「ディレクトリを変更」を選択すると、現在開いているファイルが存在しているディレクトリに移動（cd）されます。

[*5] 単独でもインストールは可能です。

4.2 Latex 用ファイルのビルド

4.2.1 使用手順

1. 添付の **lualatex-build** (sh) ファイルを project フォルダへ配置します。
   ※ snippet へ登録することで作業フォルダへのスクリプト設置を容易にします。
2. スクリプトに実行権を与えます：
   $ chmod 744 lualatex-build
3. 処理の実行：Terminal
   $ ./lualatex-build **sample.tex**
4. 処理終了と同時に生成された PDF ファイルが開かれます。

※ default の PDF ビューワとして evince がプリインストールされていることが多いですが、問題[*6]がありますので Atril その他の使用をお薦めします。
$ apt-get install atril

[*6] **auto-pst-pdf-lua** との関連で evince-thumbnai が実行されたままになり膨大なリソースを奪い「CPU 使用率が 100% 近くになり戻らない」バグがかなり以前に報告されています。
- [Bug 1386120] Re: evince and/or evince-thumbnailer stuck with high cpu load on specific dvi file // The Mail Archive: https://www.mail-archive.com/ubuntu-bugs@lists.ubuntu.com/msg5738211.html
- evince-thumbnailer with a 100% CPU usage // Ubuntu evince package https://bugs.launchpad.net/ubuntu/+source/evince/+bug/384062

4.3 SOURCE

※ PDF ビューワとして Atril が設定されていますが、他のビューワを使用する場合は各々置き換えて下さい。
atril $FILE_PDF &

4.3.1 SOURCE：端末用

```
#!/bin/sh
# 実行するファイル
FILENAME="$1" # 引数による実行ファイル名
LUA_CMD="lualatex -shell-escape"

# 生成されるファイル
FILE_PDF="`basename "$FILENAME" .tex`.pdf"
FILE_BCF="`basename "$FILENAME" .tex`.bcf"

# LUALATEX の実行
if [ -e $1 ]; then
  echo "\n=====" $1 "の処理を開始します："
  $LUA_CMD $FILENAME
  echo "処理は正常に終了しました。PDF ファイルが開かれます：\n"
else
  echo "----------------------------------------"
  echo " ファイル名エラー：[" $1 "]"
  echo "   処理に失敗しました。"
  echo "   ⇒ ファイルが存在するかを確認して下さい。"
  echo "----------------------------------------\n"
  return
fi

# BIBER .BCF が存在する場合：
if [ -e $FILE_BCF ]; then
   echo "BCF File のコンパイルを開始します ------"
   biber $FILE_BCF
   echo "BCF File のコンパイルを終了しました ------"

   echo "処理を反映させる為、再度のコンパイルを行います ------"
   $LUA_CMD $FILENAME
```

```
   echo "処理は反映されました ------"
fi

# OPEN .PDF
if [ -e $FILE_PDF ]; then
  atril $FILE_PDF &

fi

exit
```

4.3.2 SOURCE：snippets 用

※ snippet へはシステム文字へのエスケープ処理が必要です。

1. 新規文書を「LaTeX」にし、アクティベーションで設定したタブ名で読込みます。
2. 作業フォルダに「lualatex-build」として保存し実行します。
   $ ./lualatex-build sample.tex

```
#!/bin/sh
# 実行するファイル
FILENAME="\$1"
LUA_CMD="lualatex -shell-escape"

# 生成されるファイル
FILE_PDF="\`basename "\$FILENAME" .tex\`.pdf"
FILE_BCF="\`basename "\$FILENAME" .tex\`.bcf"

# LUALATEX の実行
if [ -e \$1 ]; then

  echo "\n=====" \$1 "の処理を開始します："
  \$LUA_CMD \$FILENAME
  echo "処理は正常に終了しました。PDF ファイルが開かれます：\n"

else

  echo "----------------------------------------"
  echo " ファイル名エラー：[" \$1 "]"
  echo "   処理に失敗しました。"
  echo "   ⇒ ファイルが存在するかを確認して下さい。"
```

```
  echo "----------------------------------------\n"
  return

fi

# BIBER .BCF が存在する場合：
if [ -e \$FILE_BCF ]; then

   echo "BCF File のコンパイルを開始します ------"
   biber \$FILE_BCF
   echo "BCF File のコンパイルを終了しました ------"

   echo "処理を反映させる為、再度のコンパイルを行います ------"
   \$LUA_CMD \$FILENAME
   echo "処理は反映されました ------"

fi

# OPEN .PDF
if [ -e \$FILE_PDF ]; then

  atril \$FILE_PDF &

fi

exit
```