Autor: Sergei Smyslov
in: https://www.linkedin.com/in/smyslov-sergei/
United Kingdom

Data: April 2024 | revision 1.

Code Analyzer

Annotation
This application was created for analyzing including files and creating graphics with dependencies
It can help you create more independent code or understand and minimize dependencies
For analysis was added functionality – analyze only user files (don't create dependencies by standard C and C++ libraries), add this analyze in graphics for visualization and minimization of usage libs, and also for different architectures (like embedded) was added possibility edit user libs and mard them like "standard libraries" for analyze.

For big parts of the project (more than 25 files or difficult dependencies) was added possibility to analyze without drawing image (only text information in the log).

How to install
Application ver CodeAnalyzer_ver1.0.deb was tested with Ubuntu 22.04, installations process was written for this OS.

Download file CodeAnalyzer_ver X.Y.deb
in terminal input

```
cd ~Download
sudo apt update
sudo apt install ./CodeAnalyzer_ver X.Y.deb
```

confirm installation necessary libs

Start work
In Menu with application press on icon (see Figure 1)

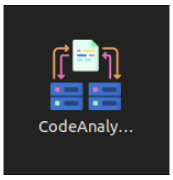
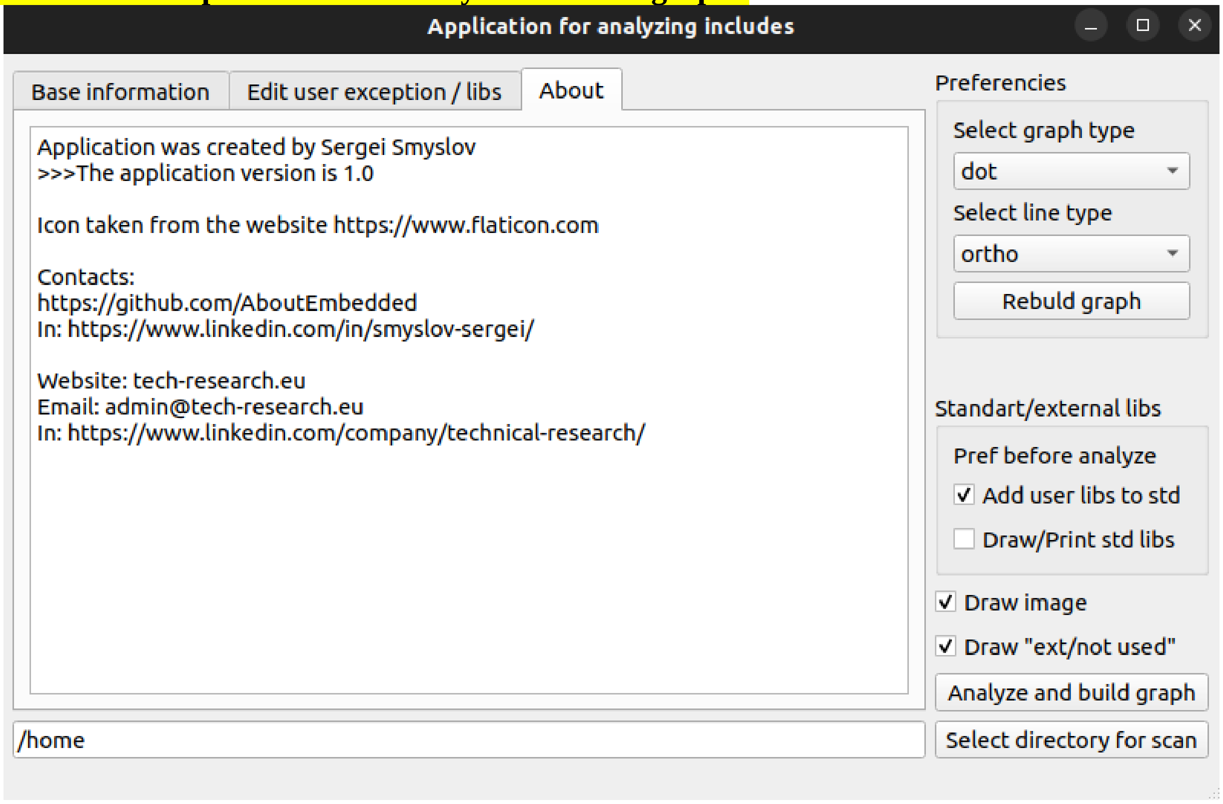


Figure 1 – icon CodeAnalyzer in app menu

In opened window (see Figure 2)
Press button "Select directory for scan" and select folder with your project (or part of them). For start work – press button "Analyze and build graph"



Application was created by Sergei Smyslov
>>>The application version is 1.0

Icon taken from the website https://www.flaticon.com

Contacts:
https://github.com/AboutEmbedded
In: https://www.linkedin.com/in/smyslov-sergei/

Website: tech-research.eu
Email: admin@tech-research.eu
In: https://www.linkedin.com/company/technical-research/

Select graph type: dot
Select line type: ortho
✓ Add user libs to std
☐ Draw/Print std libs
✓ Draw image
✓ Draw "ext/not used"

/home

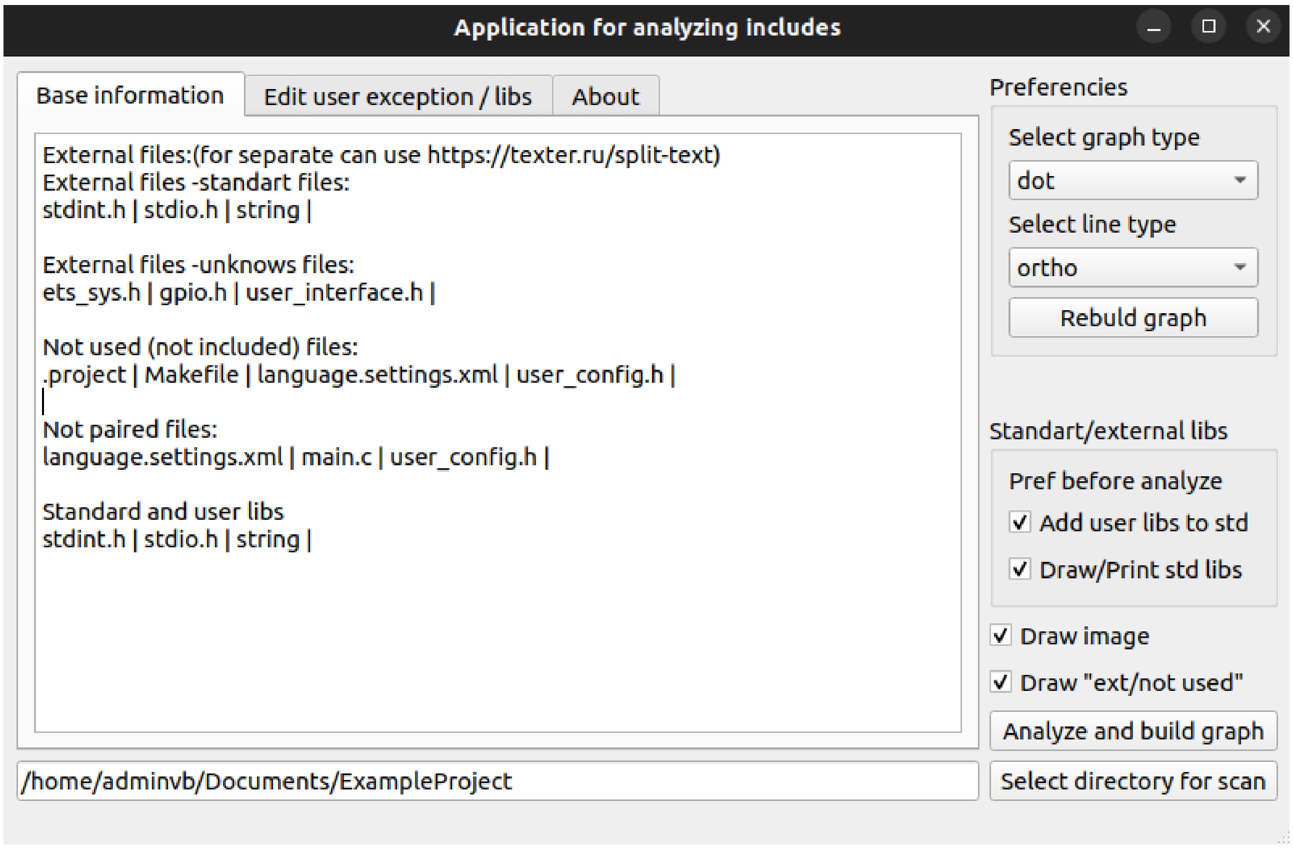
Figure 2 – Application base view

Description of functionality:

Type	Name	Description
Menu	Preferences	Menu with graph and line type – change visualization for image After "Analyze and build graph" you can use button "Rebuild graph" for a fast-changing view.
Checkbox	Add user libs to std	Add user libraries to "standard libraries"
Checkbox	Draw/Print std libs	Add user libraries to output files (in image or on base information) **Note:** on graph std libs drawing in rectangle, other libs in ellipse
Checkbox	Draw image	if selected, the app will draw image, **Note:** for analyzing big folders/projects can be selected
Checkbox	Draw "ext/not used"	if selected, the app will draw image with all external libraries

Example
We can try analyze folder with project (ExampleProject) and see all dependencies
On tab “Base information” (see Figure 3) you can see next data about your project:

Point of menu	Description
External files – standart files:	Print all standard and user libraries (which you define like the standard on the tab “Edit user …./libs”)
External files -unknows files:	Potential problems and dependencies in your project
Not used (Not included) files:	Its external files – which necessary for your project. files which exist in folders, but not used in code (for example trash file user_config.h on Figure 3)
Not paired files:	Usually, all files .c and .h must-have pairs with the same name – if not, then the project has potential difficulties.

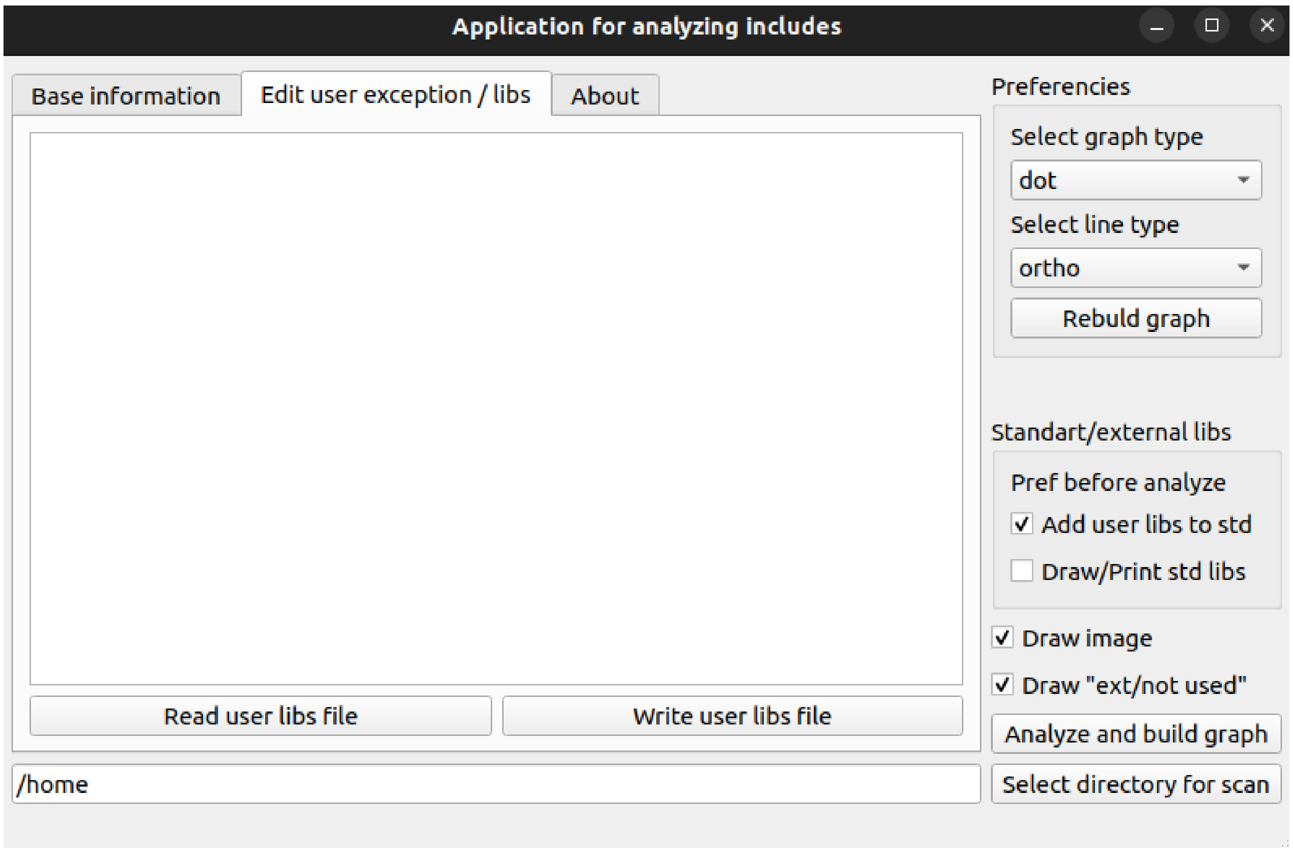


```
External files:(for separate can use https://texter.ru/split-text)
External files -standart files:
stdint.h | stdio.h | string |

External files -unknows files:
ets_sys.h | gpio.h | user_interface.h |

Not used (not included) files:
.project | Makefile | language.settings.xml | user_config.h |

Not paired files:
language.settings.xml | main.c | user_config.h |

Standard and user libs
stdint.h | stdio.h | string |
```

Select graph type: dot
Select line type: ortho
✓ Add user libs to std
✓ Draw/Print std libs
✓ Draw image
✓ Draw "ext/not used"
/home/adminvb/Documents/ExampleProject

Figure 3 – tab Base information

Edit user libraries / exceptions
For your code with different architecture and standard libraries, you can add your exception (the program in analyzing will think that its “standard libraries”)
For edit libraries – open the tab “Edit user ...lins” (see Figure 4)
Press the button “Read user lib file” for read the current user exception
after you can modify these exceptions
Press button “Write user lib file” for write data to user standard libs.



Preferencies
Select graph type: dot
Select line type: ortho

Standart/external libs
Pref before analyze
[✓] Add user libs to std
[ ] Draw/Print std libs
[✓] Draw image
[✓] Draw "ext/not used"

/home

Figure 4 – tab “Edit user libs”

Graph for simple example project WITH SELECTED CHECKBOX “DRAW EXT/NOT USED” you can see in Figure 5 and WITH DESELECTED CHECKBOX in Figure 6.
Yellow color – external libraries
Red color – not used (partly trash files)
Purple color – files without pairs

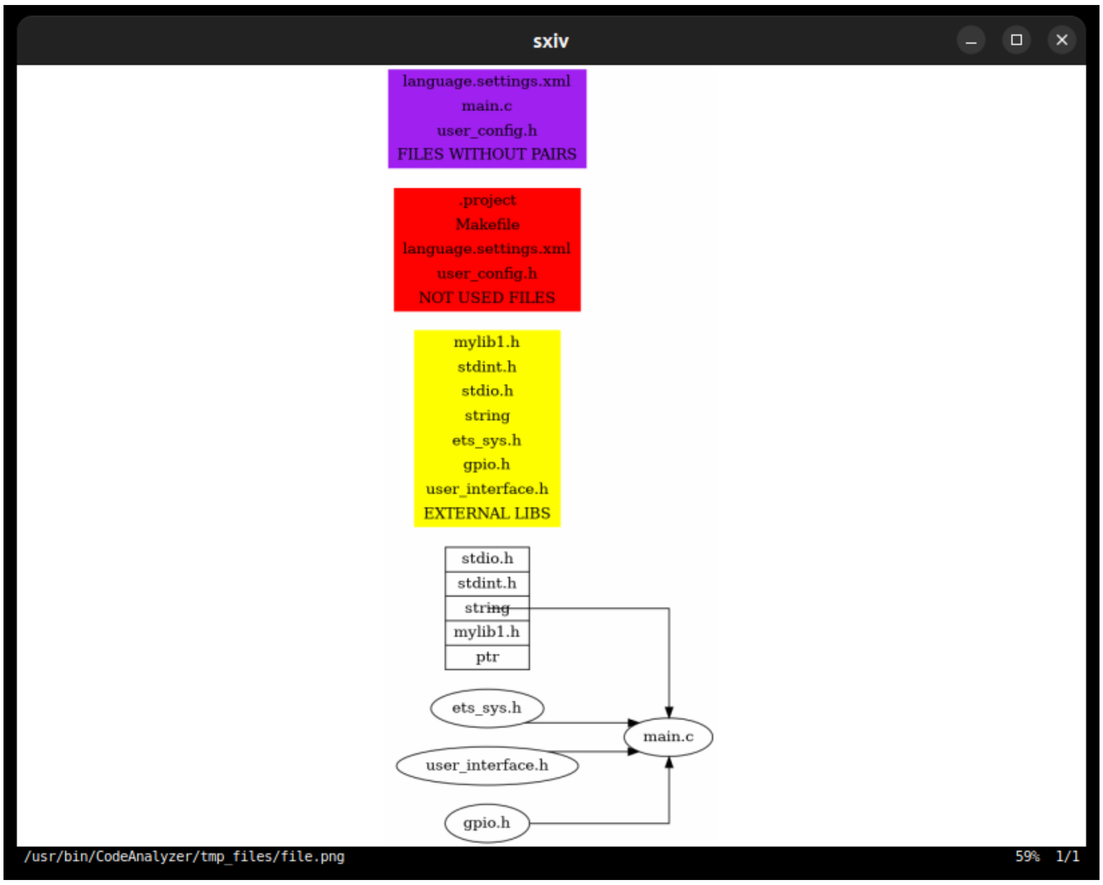


Figure 5 – graphic with “draw ext/not used” option

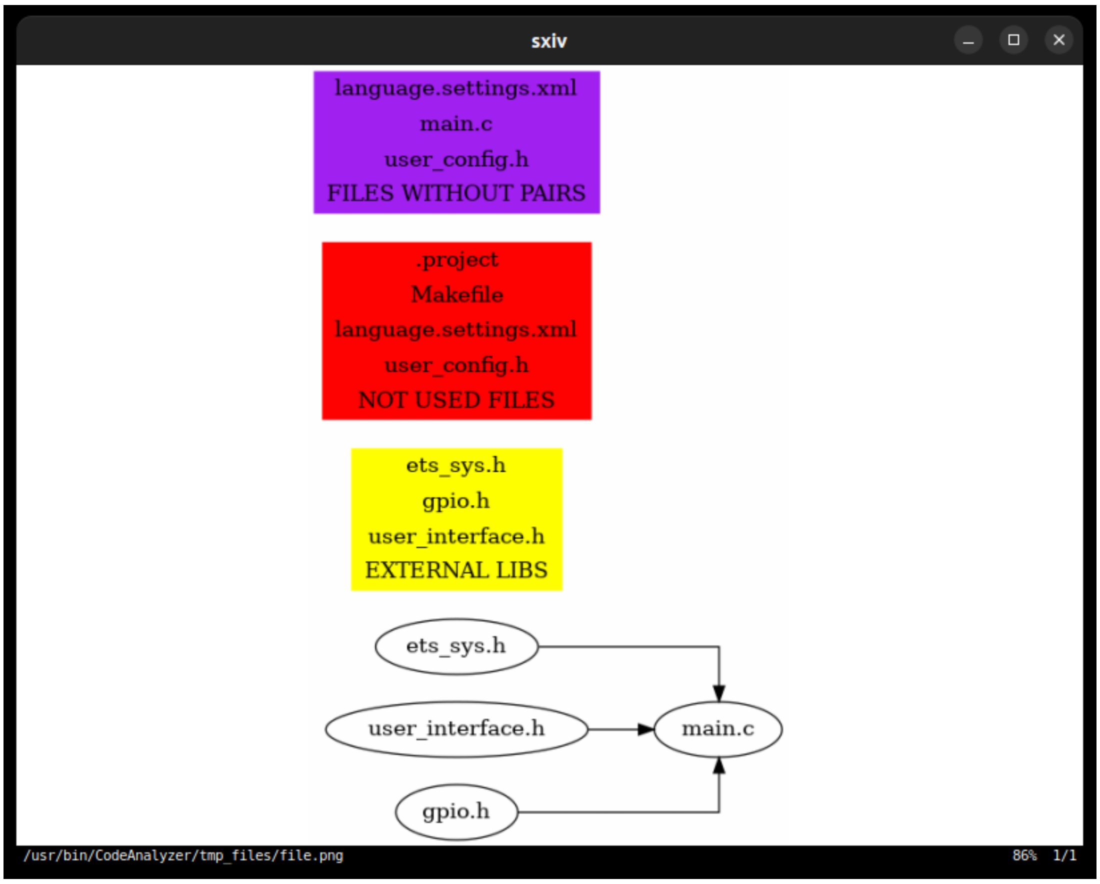


Figure 6 – graphic with no “draw ext/not used” option