FLAM ®

FRANKENSTEIN-LIMES-ACCESS-METHOD

FLCC

User Manual

Edition September 2018 Version 5.1.9

phone (06172) 5919-0 • fax (06172) 5919-39
www.flam.de

User Manual FLAM® flcc

Contents

List of Figures

Preface

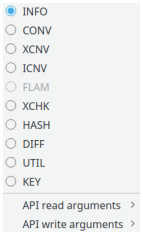
FLUC-Commander is a perspective of our new GUI (Frankenstein-Limes-Control-Center (FLCC)). By showing all available arguments together with their help text and manual page, it makes it easy to build complex command lines and is an interactive supplement of the manual at the same time. The objective of the commander is to make it easier for our customers learning to deal with our new command line.

All parameters are shown in a tree view and can be selected and/or edited in place. The parameter values can be stored in a property file or the respective command might be executed at once.

To ease the use of the FLUC byte API the argument strings needed there can be build in the same way. However the execution of API argument strings is not possible within FLCC.

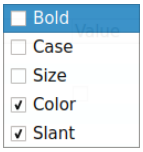
This manual covers the features and working of the GUI. For a detailed description of FLAM please see its manual.

1 Design



Commands menu

All arguments of a **flcl** command are shown in a tree structure of the main window. The argument names might be abbreviated on the command line. The characters needed to make an argument unique might be visualized if the option Aa is switched on. The available options are to show the required characters in **bold** font in the tree view of **flcc** (for example: **ENC**ODINGS) or the remaining characters with lower case, a smaller font, in gray color, or *slanted*. To edit argument values 2 modes of operation are available:



Abbreviation options

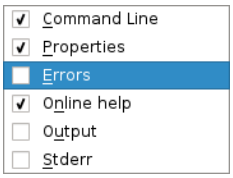
Properties In this mode the values of all arguments are mutable and can be stored in a property file. See section 2

Command line In this mode only the active arguments are mutable in order to make a valid **flcl** command line. See section 3

The mode is switched with a click on the labeled button in the tool bar at the top of the flcc window. Its label will show the currently active mode. Additionally the background color of every other line of the tree in Properties mode is a light blue while this changes to a light green in Command line mode.

All other parts of the main window are placed in so called docking windows. A docking window can be hidden, resized and dragged within the main window to its left, right, top and bottom edges. Or it might be removed from the main window and shown within its own frame. The size, position and state of all docking windows is saved on exit and restored on the next start. The windows menu contains a list of all docking windows. Hidden windows can be recalled here. The following windows support this features:

- Command line
- Properties
- Errors
- Online help
- Output



Window menu

By making one of the first 2 windows visible the operation mode is switched accordingly.

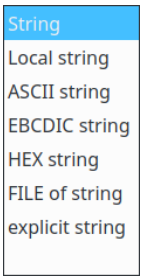
All **flcl** commands are listed in the *Commands* menu. As only one command can be used at runtime of **flcl**, only the arguments of one command are accessible in **flcc** at a time. The command selection might be done with the command menu, or by opening an existing property file. Property files normally contain the command name in the first line. This is used on open to select the command automatically.

Note: If the command name is removed from the first line of a property file, this automatic selection of the command will cease to work.

Each argument has one of the types listed in table 1. In **flcc** the modification of an overlay is done by selecting the appropriate entry from a list box. This list box appears with a click on the value column of an overlay type. Switches and Objects are switched ON/OFF by clicking on its check button. When a string is used as a file name its name might be selected with a file dialog. To open it the secondary mouse button must be clicked on its value column.

Type	**Values**	**Description**
Object	INIT, ""	Container of arguments
Overlay	Name of active argument	Container of arguments, but only *one* is used
String	any alphanumeric string	
Number	any valid number	
Switch	ON, OFF	

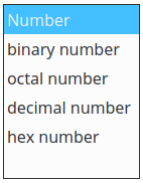
Table 1: Argument types



String formats

The format to use for a string on the command line is selectable in the type column. With the option **FILE of string** the name of a file containing the string must be specified. When this format option is used the filename might be selected within a file dialog.

The format to use for a number on the command line is selectable in the type column. With the **time** option the date and time value to use can be selected within a calender.



Number formats

Unsaved changes will be shown with a yellow background, or red if selected, in the tree structure to provide visual feedback of modifications.

The expand all button allows to open all child arguments and will keep them open.

The view all button switches the display of inactive arguments on or off. If switched off, inactive arguments are not show in the tree. If switched on they are shown in the tree but are disabled for editing.

2 Properties

The properties mode allows to modify the value of all arguments of the selected **flcl** command. A value set on the **flcl** command line takes precedence over a value set in the property file. The structuring elements of the **flcl** command line are objects and overlays. In the properties mode they are handled as follows.

Overlay Only the active child of an overlay is used by **flcl** at runtime, but a property file might contain values for each child of an argument. Which of them is used can be specified in the command line or, if none is given, a default might be selected in the property file. Unset overlays are not used at **flcl** runtime, but if specified on the command line its descendant arguments will use the values set in the property file, if any.

Objects Unset objects are not used at **flcl** runtime, but if specified on the command line its child arguments will use the values set in the property file, if any.

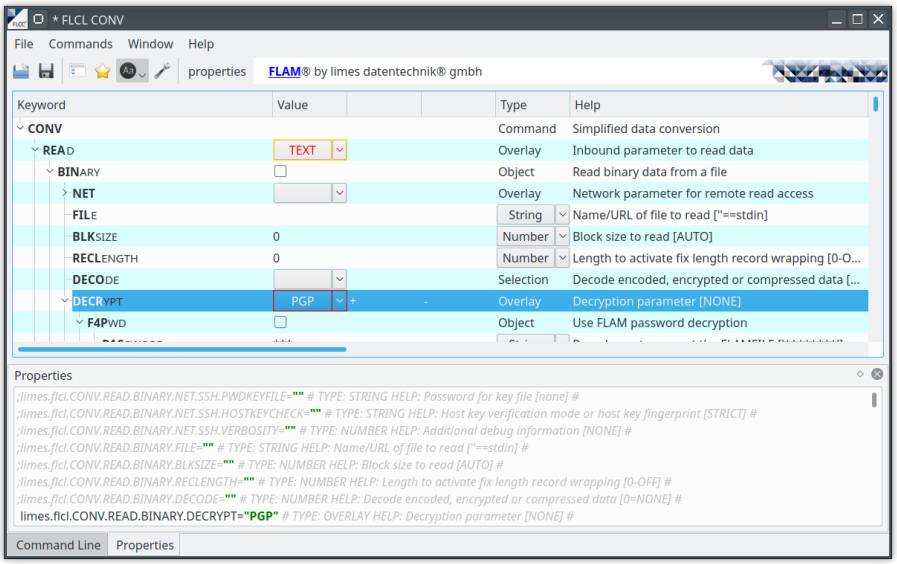


Figure 1: Properties

In the Properties window the currently selected property file is shown with syntax highlighting. If no existing property file was opened a property file is generated for the active **flcl** command. This property file might contain comments with explanations for certain arguments. This comments are not used by **flcc** or **flcl** and might be changed as desired within **flcc** or other text editors. The one exception is the first line where the owner Id and command name is used by **flcc** on open to set the corresponding values automatically. Comments in property files are shown in light gray in order to improve the visibility of the actually used arguments. A multi-line comment begins and ends with a single '#' character for **flcl**, but **flcc** needs a space *after* the beginning and a space *before* the ending '#'. A ';' starts a single line comment.

By selecting a line in the tree the corresponding line in the property file is selected if there is one. By setting a value an appropriate line is inserted if none is found in the property file. Otherwise the old value will be replaced with the new value and the line comment sign ';' is removed if necessary.

All argument values might also be entered directly in the Properties window. This input is inserted in the tree as it is typed, as long as the syntax is not broken. A broken syntax will be shown in the Errors window as it occurs while typing. The error window is forced to the front if necessary. The output shown in the Error window contains the position and description of the error to allow an immediate fix. A syntax error in a property file disables the save function. This makes it impossible to save a property file with syntax errors with **flcc**.

The contents of the Properties window is saved in the property file if the option to **Only save used properties** is unset. If this option is set all comments with the exception of the first line won't be written to the property file.

The name of the current property file is shown in the title bar. If there are unsaved changes in the property file a '*' is shown in front of the file name. In this case a message box is shown if an action is initiated which would result in data loss. This actions are:

- changing a new command from the command menu
- opening a new property file
- closing the **flcc** window

3 Command line

The main difference of the command line mode to the properties mode is the handling of overlays. If an overlay is set in the command line mode its active child is selected and all other childs are hidden or disabled in the tree, depending on the option to show inactive arguments or not. The entered values are used to build a valid **flcl** command line in the Command line window. To easier comprehend complex command lines it might be shown in a structured way by selecting this option in the view selection at the top of the Command Line window.

For users of FLAM on z/OS the view mode might be set to '**as JCL**'. In this mode the command line is shown within the required JCL code to run it on z/OS. The JCL code can be read from a user supplied template file which is configurable in the options dialog. If no template file is given a builtin default is used. If the  button is switched ON the command line will be checked for syntax errors when it is changed in the text field. If an error is encountered it is shown in the error display pane.

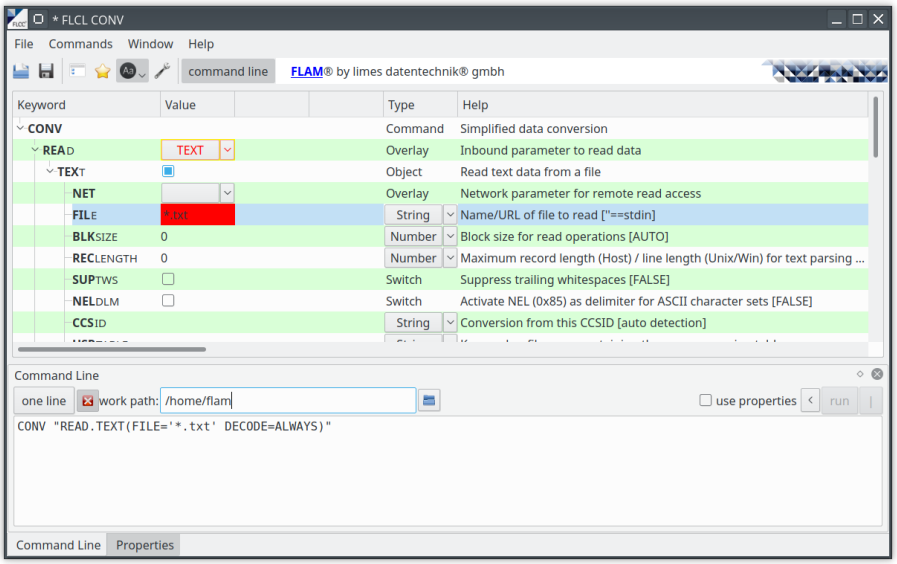


Figure 2: Command line

Arguments defined as arrays are shown with a  to the right of the value column. A click on this icon will add an additional element to the array after the clicked argument. A click on the symbol will remove the corresponding array element.

3.1 Command execution

To execute a **flcl** command from **flcc** any directory can be specified as work path in the entry field of the Command Line window. The directory might be selected in a file dialog opened when the  button is clicked. In this dialog its also possible to create a new directory.

A click on the **run** button will start **flcl** in a separate process. The output of **flcl** is shown in the Output window as it occurs, at the top of the output window the complete command as it was executed is shown. As long as **flcl** is executing the text label of the **run** button changes to **stop**, indicating that another click will terminate the **flcl** process without delay.

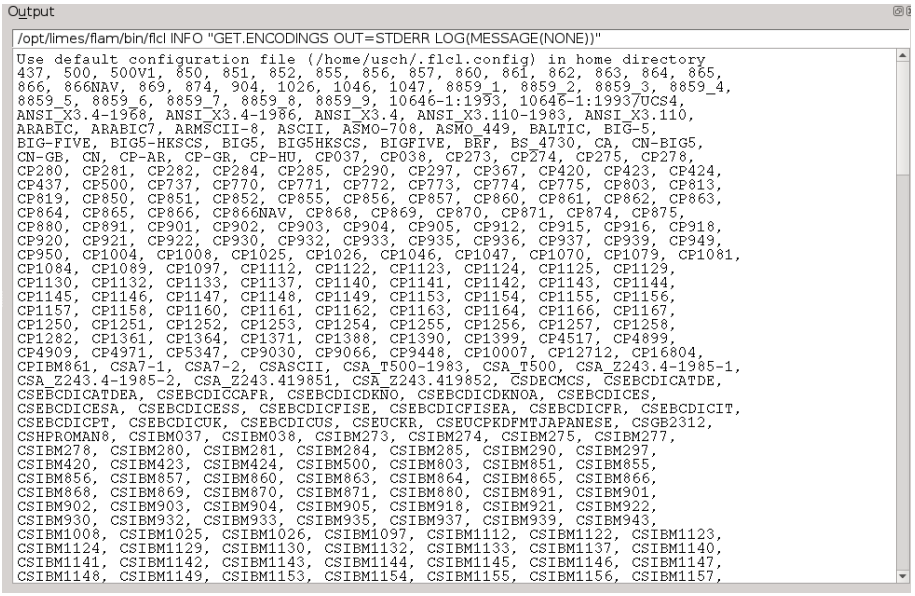


Figure 3: Output window

If the **use properties** feature is checked, the current property file is used to execute the **flcl** command. This is done by writing the name of the current property file to the **flcl** config file in the current working directory. The **flcl** config file can be modified with the edit dialog of **flcc**, see section 5.2.

3.2 History

Each successful execution of **flcl** is stored in the **flcc** command history. Previous commands are accessible with the 2 buttons near the **run** button. If they show arrows there are commands in the history to switch to by clicking on them. 20 commands are saved to the history, the oldest will be overwritten when this, currently fixed, limit is reached. **flcc** keeps a separate history for each of the **flcl** commands.

4 Online help

The extensive online help of **flcl** is also available in **flcc**. By clicking on an entry in the tree the appropriate help of **flcl** is shown in the Online help window. Above the help text the path of the **flcl** argument is shown. The following sections describe the different types of help selectable at the top of the Online help window.

The text is shown in a font with fixed width. Since **flcc** has to be work on different operating systems it uses a portable way to select a font like this. This design brings the small disadvantage that the text will not wrap around when a line becomes longer than the window width.

4.1 Manual

If Manual is selected, the help text from the **flcl** reference manual for the selected argument path is shown. In figure 4 the same text as **flcl** will print with the command

```
flcl manpage conv.read.text
```

is shown.

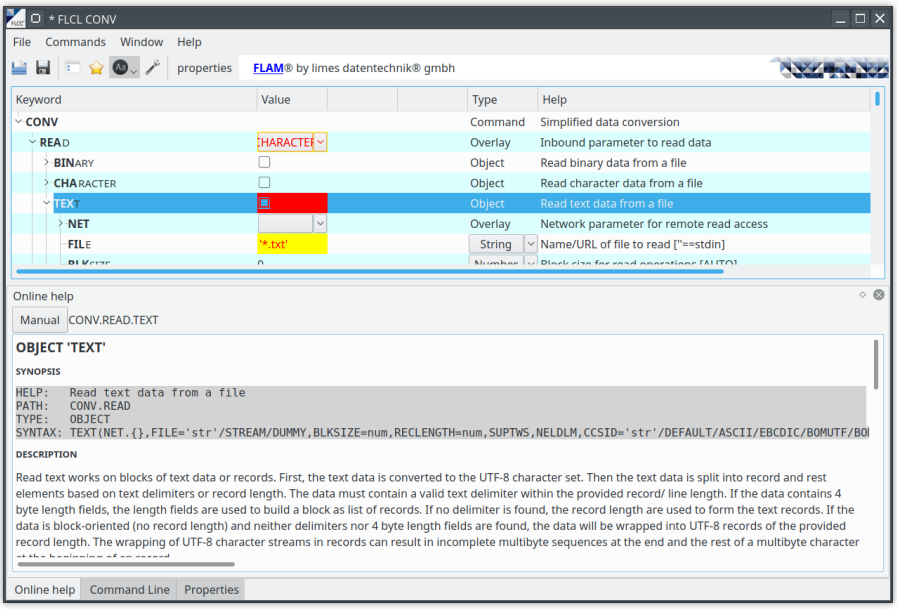


Figure 4: Manual

4.2 Help

If Help is selected, the help text of the selected argument path is shown. This is the same text as **flcl** will print with the command

```
flcl help conv.read.text all
```

4.3 Syntax

If Syntax is selected, the syntax help text of the selected argument path is shown. This is the same text as **flcl** will print with the command

```
flcl syntax conv.read.text all
```

4.4 Grammar

4.5 Lexeme

5 Configuration

5.1 Options

The dialog to modify the **flcc** options is shown in figure 5. All options are immediately used once they are modified.

The **flcl** install path is the directory where the **flam** binaries are installed. If a directory is entered where the **flcl** binary is not present an error is shown.

The owner is the content of the flcl.owner.id value from the **flcl** config file.

The option `Only save used properties` determines if a property file is saved as shown in the Properties window or if only the NOT commented arguments in the property file are saved.

The trace option allows to switch the **flcl** trace feature on or off. If switched on, **flcl** will write extensive internal information about its execution to the file given in the entry box of the dialog. Usually this is only needed for debugging **flcl**.

The template JCL code used when the view mode of the command line is set to '**as JCL**' can be set and modified here.

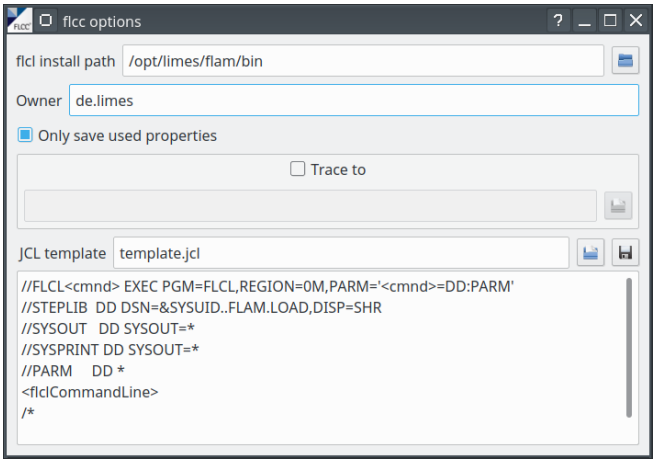


Figure 5: Options

5.2 flcl config

The **flcl** config can be modified with the dialog shown in figure 6. By typing a new variable name in the entry field at the top, a new environment variable can added to the config file. The action is initiated with the <Enter> Key after the new name is complete. This brings the keyboard focus to the value of the new key in order to type it, once finished it is saved by pressing the <Enter> Key again.

The **remove** button allows to delete the currently selected entry from the config file.

Note: No confirmation is needed to delete an entry.

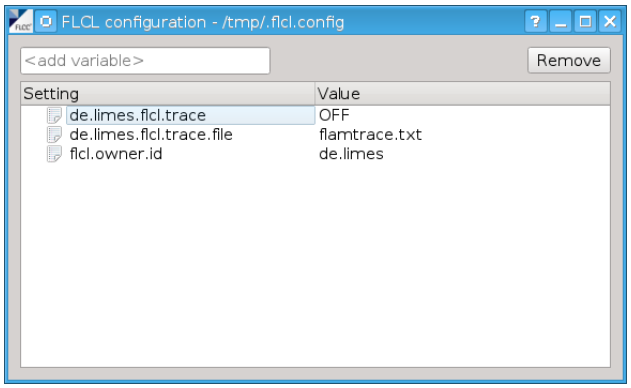


Figure 6: **flcl** configuration

If `use properties` is checked in the command line window an entry containing the name of the property file is added to the config file. This is used in the next execution of **flcl** with the matching command name.

5.3 flucFS config

On Linux systems FLAM provides the ability to use it as a file system which makes its features transparent to an application. Writing a file in a folder mounted with **flucFS** will result in a converted, encrypted, compressed version of the file as specified within the write configuration of **flucFS**.

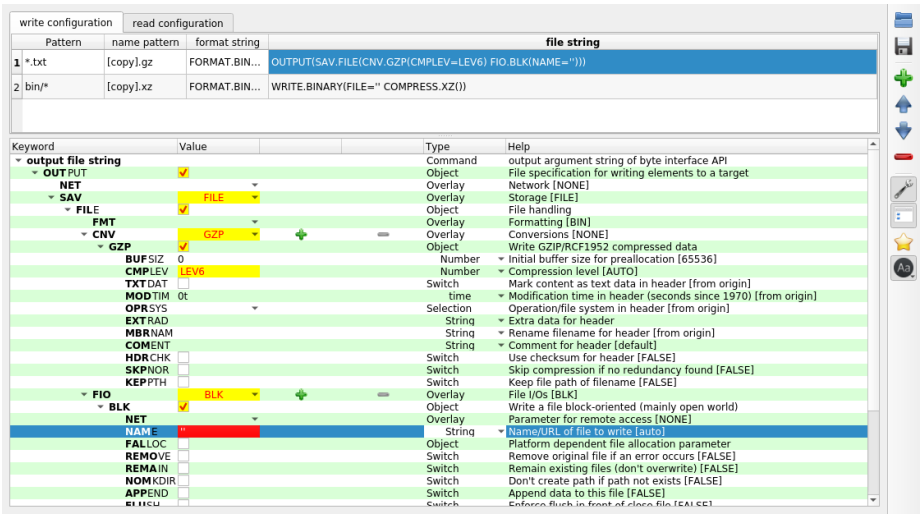


Figure 7: **flucFS** configuration

The configuration dialog shown in figure 7 allows to manage the **flucFS** configuration conveniently.

The upper half of the dialog shows the 2 config tables used by **flucFS**, one for write requests and one for read requests. A normal folder is used by **flucFS** as its *root* folder. Files written to the *mount* folder will be stored here after conversion according to its write configuration. Files written to the *root* folder are converted according to the read configuration when accessed from the *mount* folder. The read configuration is used by **flucFS** only if a file is opened which was not accessed by **flucFS** before. When reading files previously written to the mount folder **flucFS** knows how to read them by using the meta data stored on write. A fast memory mapped database is used to store configuration and meta data. This database is stored in a subdirectory of the *root* folder with the name `.flucIndex`.

The configuration is done with the argument strings of the FLUC byte interface API. However, since **flucFS** is a filesystem and uses only the read and write functions of the API, the element access features of the API are not useable with **flucFS**.

The 'pattern' value is applied to the filename of a request. If the filename is matching the config entry is used for the request. The first entry matching will be used. The order of the entries is controlled by the integer 'index' value. The matching is done in increasing 'index' order. If no matching entry is found the request is passed through without any data modification. The order can be defined in the editor by using the  and  buttons. New table entries will be inserted before the current row with a click on the  button. The current row can be removed from a table

with the button. A click on the button will save the current content of the write and read config tables. With a click on the button a new *root* folder can be selected in a file browser.

The file string can be given in simplified form as with the **flcl** CONV command or with the full flexibility as with the **flcl** XCNV command. To switch between this 2 modes the button can be checked/unchecked.

Attention: If this mode is switched the previous content of the file string is lost, it is replaced with an empty string in the new mode.

The editing of the format and file strings is done in the same way, with the same view options, as the **flcl** command line described in section 3.

The *name pattern* column is used to generate the physical filename for write requests from the given logical name. For read requests the *name pattern* is used to generate the logical name from the physical name only if the file was not accessed by **flucFS** before.

Please note: For this naming scheme to work the file string argument *must* contain an empty file name specification: `name=''` or `file=''`

For easy *name pattern* input they can be assembled with a specialized dialog as shown in figure 8. It is shown in the bottom half of the window when a pattern of a config table is selected.

Figure 8: pattern string input

The string shown as filename is just an example filename used as input to the pattern matching with the current pattern. The result of matching this filename with the current pattern is shown immediately. The available building blocks for this pattern matching are shown in the list on the left side. They can be dragged and dropped to the pattern too.

Appendix

6 Installation

Currently **flcc** is available for Linux and Windows running on x86 compatible processors. The cross-platform UI framework Qt from http://www.qt.io is used to build **flcc**.

6.1 Linux

flcc for Linux is provided in 2 variants: one is built with the current Qt5 version and one is built with the older Qt4 version. This is needed in order to run **flcc** on enterprise distributions like Red Hat or Suse which have a conservative upgrade policy.

The Qt5 Variant additionally is usable from 2 packages:

flamgui contains the required Qt5 shared libraries.

flamgui-qt5 depends on the Qt5 libraries from the distribution.

This means: if the **flamgui-qt5** package can be installed with the package manager of a distribution without disabling the dependency checking it should be usable on this system by using the Qt5 shared libraries from the distribution. In case this fails due to the fact that the distribution does not have packages which contain Qt5, **flamgui** can be installed. Unfortunately this does not work everywhere, Suse Linux Enterprise Server 11 for example is such a case.

The Qt4 variant of **flcc** is usable by installation of the **flamgui-qt4** package. The Qt4 libraries from the distribution are used which should have a version of at least 4.6.

Both variants might be installed in parallel. The Qt4 binary has the name **flcc-qt4**.

6.2 Windows

flcc for Windows is part of the **flam** msi package and does also contain the necessary Qt5 DLLs.