SlimExc

User Manual

A deterministic exception handling implementation for C++ with GCC

Documentation version: 0.9.0
for SlimExc-plugin version: 0.9.0
and SlimExc-library version: 0.9.0

Philipp Rimmele - developer@philipp-rimmele.de Valentin Felder - fritz-valentin@web.de

June 16, 2024

Contents

1. Environment

The Slim-Exception-System consists of the compiled GNU Compiler Collection (GCC) plugin (binary ”SlimExc”), the Slim-exception handling library (SlimExcLib) and 3 user defined functions (subsection 1.4).

1.1 GCC and Platform compatibility

The current version of the Slim-Exception-Plugin (0.9.0) is successfully tested with the following versions of the GCC on the following platforms:

Development platform / Target platform	x64 Linux*	x64 Windows
x64 Linux*	10.1.0, 10.2.0, 10.3.0, 10.4.0, 11.1.0, 11.2.0, 11.3.0	-
x64 Windows	-	-
ARM-none-eabi	10.3.1	-

* Tested on Ubuntu18.04 and OpenSUSE Tumbleweed

The same plugin-binary was used on all platforms. It is not unlikely the Slim-Exception-Plugin will also work on other target platforms. However, it is confirmed that GCC-Versions 9.* and 12.* do not work properly to full extent. For testing it with unverified GCC-versions the version check can be disabled as shown in Listing 5.

1.2 Toolchain

The g++ call scheme to compile your source-File with the Slim-Exception-Plugin is shown in Listing 1.

- The parameter ”-fplugin” specifies the Path to the Slim-Exception-Plugin Executable and enables its usage.
- The plugin parameters are listed in subsection 2.1. If not specified, default values will be used.
- C++-17 or newer must be used.
- The optional GCC parameter ”-fno-rtti” is used to disable the C++ runtime type information system. It may only be used if the Slim-Exception-Plugin is configured to use its own SlimRTTI mechanism instead (section 2).
- In order for the Slim-Exception-System to work properly it is necessary to keep the C++ standard exception handling enabled. The flag ”-fno-exceptions” must not be used.
- All other GCC-Flags and GCC-Parameters as well as the linker can be used as usual.

```
g++ -fplugin={pathToPlugin}/SlimExc [plugin parameters] -std=c++17 [-fno-rtti] [other gcc parameters]
    {pathToCppFile}
```

Listing 1: Call scheme of the C++-Compiler with the Slim-Exception-Plugin

1.3 Library

The Slim-Exception handling library needs to be included in all project files which use exception handling mechanisms. It may be convenient to include the library in form of a git submodule (Listing 2).

```
git submodule add https://oko.spdns.de/git/admin/projects/oko/DeterministicExceptionHandlingLibrary
```

Listing 2: Command to add the library as git submodule to your project

1.4 User defined functions

In order for the system to function, the user is required to provide an implementation of the functions "ExceptionState::getCurrentExceptionState", "ExceptionState::setCurrentExceptionState" in the namespace "SlimExcLib" and "std::terminate" if not already defined.
The implementation must provide an initial ExceptionState object as well as an updatable pointer which initially points to the initial ExceptionState object for each thread. A minimal example implementation for a single threaded system is shown in Listing 3.

- "SlimExcLib::ExceptionState::getCurrentExceptionState" must return a pointer to the active ExceptionState object of the current thread.
- "SlimExcLib::ExceptionState::setCurrentExceptionState" must update the pointer to the active ExceptionState object of the current thread.
- "std::terminate" see https://en.cppreference.com/w/cpp/error/terminate

```
#include "SlimExcLib.hpp"

namespace SlimExcLib
{
  ExceptionState baseExceptionState(NULL);
  ExceptionState* pCurrentExceptionState = &baseExceptionState;

  ExceptionState* ExceptionState::getCurrentExceptionState(void) noexcept
  {
    return pCurrentExceptionState;
  }

  void ExceptionState::setCurrentExceptionState(ExceptionState* newInstance) noexcept
  {
    pCurrentExceptionState = newInstance;
  }
}

[[noreturn]] void std::terminate() noexcept
{
  while(1){};
}
```

Listing 3: Minimal example implementation of the necessary user defined functions

1.5 Other dependencies

The Slim-Exception-System depends on a definition of the placement-new and placement-delete operator. Those are declared as "extern" in the Slim-Exception-Library. Other than that there are no dependencies to the C++-standard-library and the code can be compiled with the "-nodefaultlibs"-flag.

2. Configuration

It is possible to change the behavior of the Slim-Exception-System by using either commandline parameters or a XML-Configuration File. The default configuration is set to offer a maximal standard-compatibility as described in section 3. It is possible to restrict the features of the System for a possibly better performance and code size.

2.1 Parameters

XML	**CMD-Parameter**	**Values/Description**
	*-config=	Path to XML-Configuration-File
Exceptions/ SlimExceptions	*-f-SlimExc *-fno-SlimExc	**<u>Enabled</u>/Disabled** Enables/Disables the usage of the Slim-Exceptions (The Plugin itself still remains enabled but the default Exception-Code is not modified)
Exceptions/ ThrowableTypes	*-ThrowableTypes=	**<u>All</u>** - Maximal Functionality, allows throwing all Instances (most memory and runtime costs) **Fundamental** - Restricts the system to only allow throwing non-class-Types (reduced memory and runtime costs) **Single** - Restricts the system to only allow throwing a single Type, which is specified by ”SingleType”. (Minimal memory and runtime costs)
Exceptions/ SingleType	*-SingleType=	**char, schar, uchar, ushort, short, <u>uint</u>, int, ulong, long, ulonglong, longlong, float, double** Defines the type which is allowed to be thrown if ”ThrowableTypes”=”Single”.
Exceptions/ Buffersize	*-Buffersize=	Range: **1 - n**, Default: **<u>10</u>** The size of the Exceptionbuffer in Bytes.
RTTI/ SlimRTTI	*-f-SlimRTTI *-fno-SlimRTTI	**<u>Enabled</u>/Disabled** Enables/Disables the usage of the SlimRTTI-System for the Exception-matching. If Disabled, the default RTTI-System must be enabled instead (not using -fno-rtti).
RTTI/ PointerTypes	*-f-PointerTypes *-fno-PointerTypes	**<u>Enabled</u>/Disabled** Enables/Disables Pointer-Types in the SlimRTTI-System. (Reduced memory and runtime costs if Disabled)
RTTI/ Qualifiers	*-f-Qualifiers *-fno-Qualifiers	**<u>Enabled</u>/Disabled** Enables/Disables Qualifiers (const) in the SlimRTTI-System. If Disabled, const-Qualifiers are not taken into account on Exception-matching! (Reduced memory and runtime costs if Disabled)
Filters/ InitIgnoreFile	*-f-InitIgnoreFile *-fno-InitIgnoreFile	**Enabled/<u>Disabled</u>** Enables/Disables the initialization of the IgnoreFile with a list of used Namespaces, Classes and Functions/Methods. If Enabled, the Slim-Exception-System is automatically disabled. This can be used as a starting point for a custom Blacklist/Whitelist to exclude partial code from being handled with the Slim-Exception-System.

Filters/ InitFiltered	*-f-InitFiltered *-fno-InitFiltered	**Enabled/Disabled** Enables/Disables filtering for the initialization of the IgnoreFile. If Enabled, the current Blacklist/Whitelist is already applied to initialization.

Filters/ WhitelistStrategy	*-f-WhitelistStrategy
	*-fno-WhitelistStrategy
		Enabled/Disabled
		Enabled = Whitelist
		Disabled = Blacklist

Filters/ IgnoreFile	*-IgnoreFile=	Default: **empty** Path to the XML-Filter-File which is used as Blacklist/Whitelist to exclude partial code from being handled with the Slim-Exception-System. This path is also used for initializing the File when ”InitIgnoreFile” is enabled. If empty, filtering is disabled completely.
Diagnostics/ WrongGCCVersion	*-WrongGCCVersion=	**Ignore, Info, Warning, Error** Message when the GCC-Version is not compatible with the plugin binary. It might still work, so it is possible to decrease the Diagnostic-Level. But this can lead to undocumented problems!
Diagnostics/ StdTerminateOn Exception	*-StdTerminateOn Exception=	**Ignore, Info, Warning, Error** Message when an exception is thrown which is never caught, leading to a guaranteed call of ”std::terminate”.
Diagnostics/ PossiblyUncaught Exception	*-PossiblyUncaught Exception=	**Ignore, Info, Warning, Error** Message when an exception could propagate out of a non-throwing region. This can be the case when a ”try-catch”-block within a ”noexcept” section doesn’t include a catch-all handler.
Debugging/ Verbose	*-f-Verbose *-fno-Verbose	**Enabled/Disabled** Enables/Disables additional outputs while traversing the Abstract syntax tree (AST).
Debugging/ GenerateAST	*-f-GenerateAST *-fno-GenerateAST	**Enabled/Disabled** Enables/Disables the printing of the AST. Two files in the Debugging-Directory are generated for each function, one with the unmodified and one with the modified AST.
Debugging/ Directory	*-Debugging-Directory=	Default: **empty** Path to the Debugging-Directory, is used by ”GenerateAST”.

* Commandline Parameter Prefix: -fplugin-arg-SlimExc
_ underlined: Default Values.

2.2 Examples

The following examples show the usage of some of the command line parameters:

```
g++ -fplugin={pathToPlugin}/SlimExc -fplugin-arg-SlimExc-config={pathToPluginConfig}/pluginConfig.xml
    -std=c++17 -O3 -g3 -Wall -fno-rtti -c -o "myFile.o" myFile.cpp
```

Listing 4: Using the XML-Config-File

```
g++ -fplugin={pathToPlugin}/SlimExc -fplugin-arg-SlimExc-WrongGCCVersion=Warning -std=c++17 -c -o
    "myFile.o" myFile.cpp
```

Listing 5: Using a different GCC-Version (be aware, this can result in undocumented error-Messages)

```
g++ -fplugin={pathToPlugin}/SlimExc -fplugin-arg-SlimExc-IgnoreFile={pathToMyFilterFile}/ignore.xml
    -fplugin-arg-SlimExc-f-InitIgnoreFile -std=c++17 -c -o "myFile.o" myFile.cpp
```

Listing 6: Generating found Entries in a Filter-File

```
g++ -fplugin={pathToPlugin}/SlimExc -fplugin-arg-SlimExc-IgnoreFile={pathToMyFilterFile}/ignore.xml
    -fplugin-arg-SlimExc-f-WhitelistStrategy -std=c++17 -c -o "myFile.o" myFile.cpp
```

Listing 7: Using a Filter-File as Whitelist

```
g++ -fplugin={pathToPlugin}/SlimExc -fplugin-arg-SlimExc-ThrowableTypes=Single
    -fplugin-arg-SlimExc-SingleType=short -std=c++17 -c -o "myFile.o" myFile.cpp
```

Listing 8: Allow only throwing "short"-Types

```
g++ -fplugin={pathToPlugin}/SlimExc -fplugin-arg-SlimExc-Debugging-Directory={pathToDirectory}
    -fplugin-arg-SlimExc-f-GenerateAST -std=c++17 -c -o "myFile.o" myFile.cpp
```

Listing 9: Print the ASTs for Debugging in a Directory

2.3 Using filtering to exclude Code sections

In order to exclude precompiled libraries or legacy code without ”noexcept”-declarations it is possible to use an ignorefile (as shown in Listing 7).
To avoid having to write the ignorefile manually, it is possible to use the ”initIgnoreFile”-Option as shown in Listing 6. This initializes the ignorefile's XML-Structure and generates a list of all Namespaces, Classes and Functions which can be copy-pasted into the blacklist/whitelist section of the ignorefile.
Adding an empty XML-Tag for a Namespace or a Class selects everything inside of this Namespace or Class as well.

3. Deviations from C++-Standard

The following deviations are based on the assumption that Slim-Exception-Plugin is configured with its default values which represent the full function range. It is possible to reduce the function range and thus the C++ standard conformity in favor of better performance in the configuration (section 2).

3.1 Working with precompiled code

If standard exception are thrown from precompiled code (without the Slim-Exception-Plugin), those can not be handled by the Slim-Exception-System! Equally no exceptions from the Slim-Exception-System can be handled by precompiled code! Further it is highly recommended to exclude all precompiled code by the ”ignoreList” configuration in order to avoid unnecessary runtime overhead (subsection 2.1).

3.2 Multiple and virtual inheritance

Multiple and virtual inheritance is currently not supported for classes which get thrown as exception objects by the system.

3.3 Noexcept keyword

To minimize the runtime overhead it is highly recommended to use the official C++ keyword "noexcept" on all function and method definitions which do not throw. Unlike the C++ standard behavior, the Slim-Exception-System does not allow exceptions to propagate through functions or methods which are marked as "noexcept".

3.4 "Extern C" is treated as "noexcept"

All functions declared as "extern C" are treated as if declared "noexcept(true)". Throwing from "extern C"-Functions is not allowed with the Slim-Exception-System.

3.5 Stack unwinding in case of std::terminate

When a condition is met which leads to a call of "std::terminate", the call is immediately executed without a preceding stack unwinding. Thus, the objects on the stack don't get properly destructed. In practice this should not be a problem, because a call to "std::terminate" ends the execution of the program. However, if the Destructors are used to disable some physical periphery this can be a critical deviation from the standard. It is recommended to implement all necessary hardware shutdowns in the "std::terminate"-routine.

3.6 Exception Size limit

The size of all thrown exception object must be lower or equal to the configured Buffer Size (buffersize in configuration subsection 2.1).

3.7 Maximal pointer depth

If the SlimRTTI-System is used, the maximal pointer depth of thrown exceptions is 7.

4. Known issues

- On arm-none-eabi systems, an Implementation of the functions "__aeabi_unwind_cpp_pr0" and "__aeabi_unwind_cpp_pr1" is required but may be left empty.

5. Bug reporting

If any bugs are encountered, please follow these instructions to file a report:

1. Compile the program without the Slim-Exception-System and verify it's correctness to make sure the error actually is in the Slim-Exception-System.
2. Extract a minimal example which reproduces the bug from your code.
3. Send an E-Mail including
   - the minimal code example
   - a description of the bug

- the version numbers of SlimExc and GCC
- the target- and build platforms
- any possibly occurring compiler- or runtime errors

to developer@philipp-rimmele.de