LogLib

Generated by Doxygen 1.8.17

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

Chapter 2

File Documentation

2.1 include/trace.hh File Reference

Enumerations

- enum LogLvl { TLL_ERR, TLL_WARN, TLL_LOG }

Functions

- int tll_verbose (const char *format,...)
- int tll_warning (const char *format,...)
- int tll_error (const char *format,...)
- int tll_exit_code (bool leave=true)

  returns exit code
- void tll_set_log_lvl (LogLvl log_lvl)

  function to set logging level

2.1.1 Enumeration Type Documentation

2.1.1.1 LogLvl

`enum LogLvl`

Enumerator

TLL_ERR	
TLL_WARN	
TLL_LOG	

Definition at line 4 of file trace.hh.

2.1.2 Function Documentation

2.1.2.1 tll_verbose()

```
int tll_verbose (
            const char * format,
             ...  )
```

Parameters

format	format string
...	

Returns

upon success, return the number of characters printed

Definition at line 15 of file trace.cc.

2.1.2.2 tll_warning()

```
int tll_warning (
            const char * format,
             ...  )
```

Parameters

format	format string
...	

Returns

upon success, return the number of characters printed

Definition at line 32 of file trace.cc.

2.1.2.3 tll_error()

```
int tll_error (
            const char * format,
             ...  )
```

Parameters

format	format string
...	

Returns

upon success, return the number of characters printed

Definition at line 49 of file trace.cc.

2.1.2.4 tll_exit_code()

```
int tll_exit_code (
            bool leave = true )
```

returns exit code

Returns

0 if no errors occured, -1 otherwise

Definition at line 68 of file trace.cc.

2.1.2.5 tll_set_log_lvl()

```
void tll_set_log_lvl (
            LogLvl log_lvl )
```

function to set logging level

Parameters

log↩ _lvl	logging level

Definition at line 79 of file trace.cc.

2.2 trace.hh

```
00001 #ifndef TRACE_HH
00002 #define TRACE_HH
00003
00004 enum LogLvl
00005 {
00006   TLL_ERR,
```

```
00007   TLL_WARN,
00008   TLL_LOG
00009 };
00010
00011 /**
00012  * @brief
00013  *
00014  * @param format format string
00015  * @param ...
00016  * @return upon success, return the number of characters printed
00017  */
00018 int tll_verbose(const char *format, ...);
00019
00020 /**
00021  * @brief
00022  *
00023  * @param format format string
00024  * @param ...
00025  * @return upon success, return the number of characters printed
00026  */
00027 int tll_warning(const char *format, ...);
00028
00029 /**
00030  * @brief
00031  *
00032  * @param format format string
00033  * @param ...
00034  * @return upon success, return the number of characters printed
00035  */
00036 int tll_error(const char *format, ...);
00037
00038 /**
00039  * @brief returns exit code
00040  * @return 0 if no errors occured, -1 otherwise
00041  */
00042 int tll_exit_code(bool leave = true);
00043
00044 /**
00045  * @brief function to set logging level
00046  * @param log_lvl logging level
00047  */
00048 void tll_set_log_lvl(LogLvl log_lvl);
00049
00050 #endif // TRACE_HH
```

2.3 lib/trace.cc File Reference

TLL - Tako's Logging Library.

```
#include <stdarg.h>
#include <stdio.h>
#include "trace.hh"
```

Functions

- int tll_verbose (const char *format,...)
- int tll_warning (const char *format,...)
- int tll_error (const char *format,...)
- int tll_exit_code (bool leave)

  returns exit code
- void tll_set_log_lvl (LogLvl log_lvl)

  function to set logging level

2.3.1 Detailed Description

TLL - Tako's Logging Library.

Author

Tako-San

Definition in file trace.cc.

2.3.2 Function Documentation

2.3.2.1 tll_verbose()

```
int tll_verbose (
            const char * format,
             ...  )
```

Parameters

format	format string
...	

Returns

upon success, return the number of characters printed

Definition at line 15 of file trace.cc.

2.3.2.2 tll_warning()

```
int tll_warning (
            const char * format,
             ...  )
```

Parameters

format	format string
...	

Returns

upon success, return the number of characters printed

Definition at line 32 of file trace.cc.

2.3.2.3 tll_error()

```
int tll_error (
             const char * format,
              ...  )
```

Parameters

format	format string
...	

Returns

upon success, return the number of characters printed

Definition at line 49 of file trace.cc.

2.3.2.4 tll_exit_code()

```
int tll_exit_code (
             bool leave = true )
```

returns exit code

Returns

0 if no errors occured, -1 otherwise

Definition at line 68 of file trace.cc.

2.3.2.5 tll_set_log_lvl()

```
void tll_set_log_lvl (
             LogLvl log_lvl )
```

function to set logging level

Parameters

log↩ _lvl	logging level

Definition at line 79 of file trace.cc.

2.4 trace.cc

```
00001 /**
00002  * @file trace.cc
00003  * @author Tako-San
00004  * @brief TLL - Tako's Logging Library
00005  */
00006
00007 #include <stdarg.h>
00008 #include <stdio.h>
00009
00010 #include "trace.hh"
00011
00012 static bool IS_FAILED = false;
00013 static LogLvl LOG_LVL = TLL_ERR;
00014
00015 int tll_verbose(const char *format, ...)
00016 {
00017   if (nullptr == format)
00018     return -1;
00019
00020   if (LOG_LVL < TLL_LOG)
00021     return 0;
00022
00023   va_list args;
00024   va_start(args, format);
00025
00026   printf("[VERBOSE] ");
00027   int res = vprintf(format, args);
00028   va_end(args);
00029   return res;
00030 }
00031
00032 int tll_warning(const char *format, ...)
00033 {
00034   if (nullptr == format)
00035     return -1;
00036
00037   if (LOG_LVL < TLL_WARN)
00038     return 0;
00039
00040   va_list args;
00041   va_start(args, format);
00042
00043   fprintf(stderr, "[WARNING] ");
00044   int res = vfprintf(stderr, format, args);
00045   va_end(args);
00046   return res;
00047 }
00048
00049 int tll_error(const char *format, ...)
00050 {
00051   IS_FAILED = true;
00052
00053   if (nullptr == format)
00054     return -1;
00055
00056   if (LOG_LVL < TLL_ERR)
00057     return 0;
00058
00059   va_list args;
00060   va_start(args, format);
00061
00062   fprintf(stderr, "[ERROR] ");
00063   int res = vfprintf(stderr, format, args);
00064   va_end(args);
00065   return res;
00066 }
00067
00068 int tll_exit_code(bool leave)
00069 {
00070   if (!IS_FAILED)
00071     return 0;
00072
00073   if (leave)
00074     tll_error("Exiting...\n");
00075
00076   return -1;
00077 }
00078
00079 void tll_set_log_lvl(LogLvl log_lvl)
```

```
00080 {
00081   LOG_LVL = log_lvl;
00082 }
```

Index