Source to Flow

0.2

Generated by Doxygen 1.9.8

Chapter 1

Specification

Programozás 1 - Nagy Házi Specifikáció - Szihalmi Botond L1U7KJ

1.1 Source to Flow specifikáció

Én egy saját ötlet alapján kezdtem el dolgozni a nagy házimon. \ A célja hogy egy beolvasott c source fileból egy megjeleníthető folyamat ábrát hozzon létre.\ A programot mind parancssorból mind grafikus felülettel lehet irányítani.

1.1.1 Parancssori irányítás

Itt nem tényleges irányítás történik, csak adott lehetőségek vannak a meghívás alatt:

- help
- theme
- output file
- input file

Ha semmilyen meghívási paraméter nincs megadva csak megnyitja a program grafikus felületét. \ Ha meg van adva a bemeneti file, azt a file-ot nyitja meg a grafikus felületen \ Ha bemeneti és kimeneti file is meg van adva, rögtön kimenti a folyamat ábra képét. \ A téma paraméter ezeknek a működését nem érinti. \ A help paraméter csak kiírja hogyan kell a parancssori irányítást használni.

1.1.2 Grafikus Irányítás

`Felső menü\ Folyamat ábra`

Egér irányítás:

- görgővel lehet a folyamat ábrán belül nagyítani, kisebbíteni.
- lenyomva tartva lehet vele mozogni jobbra, balra, fel, le.
- bal kattintással lehet mozgatni a folyamat ábra pontjait.

1.1.3 Témák

Saját témát lehet megadni `.ini` file-ként. \ Mindegyik típusú objektumhoz (beleértve a fő képernyőt is) külön témát kell megadni. \ Egy objektumhoz két színérték tartozik:

- háttér
- szöveg

1.1.4 File mentés

A létrehozott folyamati ábrát vagy `.png` vagy `.jpg`-ként lehet elmenteni. \ A mentett file nevét és helyét a felhasználó adja meg. \ A mentett file a használt téma alapján legyen színezve.

1.1.5 Kilépés

Kilépésnél rákérdezünk a felhasználóra hogy biztos meg szeretné e tenni, de automatikusan nem mentünk semmit.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 colour_t Struct Reference

colouring struct for theme

`#include <ini_reader.h>`

Data Fields

- SDL_Colour background
- SDL_Colour text

4.1.1 Detailed Description

colouring struct for theme

Definition at line 34 of file ini_reader.h.

4.1.2 Field Documentation

4.1.2.1 background

`SDL_Colour background`

Definition at line 35 of file ini_reader.h.

4.1.2.2 text

`SDL_Colour text`

Definition at line 36 of file ini_reader.h.

The documentation for this struct was generated from the following file:

- ini_reader.h

4.2 conditional_type_t Struct Reference

`#include <source_reader.h>`

Data Fields

- char * condition

4.2.1 Detailed Description

Definition at line 24 of file source_reader.h.

4.2.2 Field Documentation

4.2.2.1 condition

`char* condition`

Definition at line 25 of file source_reader.h.

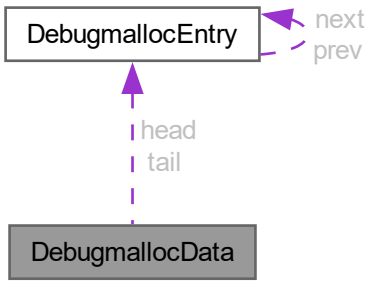
The documentation for this struct was generated from the following file:

- source_reader.h

4.3 DebugmallocData Struct Reference

`#include <debugmalloc.h>`

Collaboration diagram for DebugmallocData:



Data Fields

- char logfile [256]
- long max_block_size
- long alloc_count
- long long alloc_bytes
- long all_alloc_count
- long long all_alloc_bytes
- DebugmallocEntry head [debugmalloc_tablesize]
- DebugmallocEntry tail [debugmalloc_tablesize]

4.3.1 Detailed Description

Definition at line 64 of file debugmalloc.h.

4.3.2 Field Documentation

4.3.2.1 all_alloc_bytes

```
long long all_alloc_bytes
```

Definition at line 70 of file debugmalloc.h.

4.3.2.2 all_alloc_count

```
long all_alloc_count
```

Definition at line 69 of file debugmalloc.h.

4.3.2.3 alloc_bytes

```
long long alloc_bytes
```

Definition at line 68 of file debugmalloc.h.

4.3.2.4 alloc_count

```
long alloc_count
```

Definition at line 67 of file debugmalloc.h.

4.3.2.5 head

```
DebugmallocEntry head[debugmalloc_tablesize]
```

Definition at line 71 of file debugmalloc.h.

4.3.2.6 logfile

```
char logfile[256]
```

Definition at line 65 of file debugmalloc.h.

4.3.2.7 max_block_size

```
long max_block_size
```

Definition at line 66 of file debugmalloc.h.

4.3.2.8 tail

```
DebugmallocEntry tail[debugmalloc_tablesize]
```

Definition at line 71 of file debugmalloc.h.

The documentation for this struct was generated from the following file:

- debugmalloc.h

4.4 DebugmallocEntry Struct Reference

```
#include <debugmalloc.h>
```

Collaboration diagram for DebugmallocEntry:

Data Fields

- void * real_mem
- void * user_mem
- size_t size
- char file [64]
- unsigned line
- char func [32]
- char expr [128]
- struct DebugmallocEntry * prev
- struct DebugmallocEntry * next

4.4.1 Detailed Description

Definition at line 49 of file debugmalloc.h.

4.4.2 Field Documentation

4.4.2.1 expr

```
char expr[128]
```

Definition at line 57 of file debugmalloc.h.

4.4.2.2 file

```
char file[64]
```

Definition at line 54 of file debugmalloc.h.

4.4.2.3 func

```
char func[32]
```

Definition at line 56 of file debugmalloc.h.

4.4.2.4 line

```
unsigned line
```

Definition at line 55 of file debugmalloc.h.

4.4.2.5 next

```
struct DebugmallocEntry * next
```

Definition at line 59 of file debugmalloc.h.

4.4.2.6 prev

```
struct DebugmallocEntry* prev
```

Definition at line 59 of file debugmalloc.h.

4.4.2.7 real_mem

```
void* real_mem
```

Definition at line 50 of file debugmalloc.h.

4.4.2.8 size

```
size_t size
```

Definition at line 52 of file debugmalloc.h.

4.4.2.9 user_mem

```
void* user_mem
```

Definition at line 51 of file debugmalloc.h.

The documentation for this struct was generated from the following file:

- debugmalloc.h

4.5 func_type_t Struct Reference

```
#include <source_reader.h>
```

Data Fields

- char * return_type
- char ** args

4.5.1 Detailed Description

Definition at line 11 of file source_reader.h.

4.5.2 Field Documentation

4.5.2.1 args

```
char** args
```

Definition at line 13 of file source_reader.h.

4.5.2.2 return_type

```
char* return_type
```

Definition at line 12 of file source_reader.h.

The documentation for this struct was generated from the following file:

- source_reader.h

4.6 loop_type_t Struct Reference

```
#include <source_reader.h>
```

Data Fields

- char * condition

4.6.1 Detailed Description

Definition at line 28 of file source_reader.h.

4.6.2 Field Documentation

4.6.2.1 condition

```
char* condition
```

Definition at line 29 of file source_reader.h.

The documentation for this struct was generated from the following file:

- source_reader.h

4.7 mapping_t Struct Reference

hash-map like struct for mapping strings to anything (not very safe)

```
#include <types.h>
```

Data Fields

- const char * key
- const void * value

4.7.1 Detailed Description

hash-map like struct for mapping strings to anything (not very safe)

Definition at line 21 of file types.h.

4.7.2 Field Documentation

4.7.2.1 key

```
const char* key
```

Definition at line 22 of file types.h.

4.7.2.2 value

```
const void* value
```

Definition at line 23 of file types.h.

The documentation for this struct was generated from the following file:

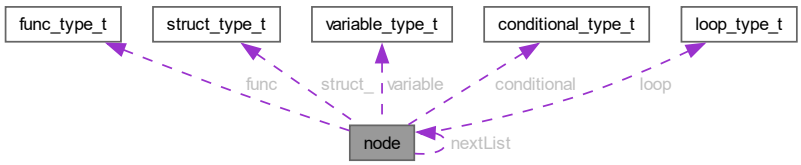
- types.h

4.8 node Struct Reference

linked list structure

```
#include <source_reader.h>
```

Collaboration diagram for node:



Data Fields

- context_e type
- char name [100]
- union {
  func_type_t func
  struct_type_t struct_
  variable_type_t variable
  conditional_type_t conditional
  loop_type_t loop
  };
- int list_size
- struct node ** nextList

4.8.1 Detailed Description

linked list structure

Definition at line 34 of file source_reader.h.

4.8.2 Field Documentation

4.8.2.1 [union]

```
union { ... }
```

4.8.2.2 conditional

```
conditional_type_t conditional
```

Definition at line 41 of file source_reader.h.

4.8.2.3 func

```
func_type_t func
```

Definition at line 38 of file source_reader.h.

4.8.2.4 list_size

```
int list_size
```

Definition at line 44 of file source_reader.h.

4.8.2.5 loop

```
loop_type_t loop
```

Definition at line 42 of file source_reader.h.

4.8.2.6 name

```
char name[100]
```

Definition at line 36 of file source_reader.h.

4.8.2.7 nextList

```
struct node** nextList
```

Definition at line 45 of file source_reader.h.

4.8.2.8 struct_

```
struct_type_t struct_
```

Definition at line 39 of file source_reader.h.

4.8.2.9 type

```
context_e type
```

Definition at line 35 of file source_reader.h.

4.8.2.10 variable

```
variable_type_t variable
```

Definition at line 40 of file source_reader.h.

The documentation for this struct was generated from the following file:

- source_reader.h

4.9 struct_type_t Struct Reference

```
#include <source_reader.h>
```

Data Fields

- char * args

4.9.1 Detailed Description

Definition at line 16 of file source_reader.h.

4.9.2 Field Documentation

4.9.2.1 args

```
char* args
```

Definition at line 17 of file source_reader.h.

The documentation for this struct was generated from the following file:

- source_reader.h

4.10 test Struct Reference

Data Fields

- int a

4.10.1 Detailed Description

Definition at line 5 of file test.c.

4.10.2 Field Documentation

4.10.2.1 a

```
int a
```

Definition at line 6 of file test.c.

The documentation for this struct was generated from the following file:

- test.c

4.11 test_t Struct Reference

Data Fields

- int b

4.11.1 Detailed Description

Definition at line 8 of file test.c.

4.11.2 Field Documentation

4.11.2.1 b

```
int b
```

Definition at line 9 of file test.c.

The documentation for this struct was generated from the following file:

- test.c

4.12 theme_t Struct Reference

struct for theme

```
#include <ini_reader.h>
```

Collaboration diagram for theme_t:

Data Fields

- colour_t functions
- colour_t structs
- colour_t variables
- colour_t conditionals
- colour_t loops
- colour_t main_

4.12.1 Detailed Description

struct for theme

Definition at line 42 of file ini_reader.h.

4.12.2 Field Documentation

4.12.2.1 conditionals

```
colour_t conditionals
```

Definition at line 46 of file ini_reader.h.

4.12.2.2 functions

```
colour_t functions
```

Definition at line 43 of file ini_reader.h.

4.12.2.3 loops

```
colour_t loops
```

Definition at line 47 of file ini_reader.h.

4.12.2.4 main_

```
colour_t main_
```

Definition at line 48 of file ini_reader.h.

4.12.2.5 structs

```
colour_t structs
```

Definition at line 44 of file ini_reader.h.

4.12.2.6 variables

```
colour_t variables
```

Definition at line 45 of file ini_reader.h.

The documentation for this struct was generated from the following file:

- ini_reader.h

4.13 variable_type_t Struct Reference

```
#include <source_reader.h>
```

Data Fields

- char * value

4.13.1 Detailed Description

Definition at line 20 of file source_reader.h.

4.13.2 Field Documentation

4.13.2.1 value

```
char* value
```

Definition at line 21 of file source_reader.h.

The documentation for this struct was generated from the following file:

- source_reader.h

Chapter 5

File Documentation

5.1 cmake-build-debug/CMakeFiles/3.26.4/CompilerIdC/CMakeCCompilerId.c File Reference

Macros

- #define __has_include(x) 0
- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X) #X
- #define STRINGIFY(X) STRINGIFY_HELPER(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define C_VERSION

Functions

- int main (int argc, char *argv[ ])

Variables

- char const * info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const * info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const * info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char * info_language_standard_default
- const char * info_language_extensions_default

5.1.1 Macro Definition Documentation

5.1.1.1 __has_include

```
#define __has_include(
              x ) 0
```

Definition at line 17 of file CMakeCCompilerId.c.

5.1.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

Definition at line 716 of file CMakeCCompilerId.c.

5.1.1.3 C_VERSION

```
#define C_VERSION
```

Definition at line 805 of file CMakeCCompilerId.c.

5.1.1.4 COMPILER_ID

```
#define COMPILER_ID ""
```

Definition at line 427 of file CMakeCCompilerId.c.

5.1.1.5 DEC

```
#define DEC(
              n )
```

Value:

```
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

Definition at line 720 of file CMakeCCompilerId.c.

5.1.1.6 HEX

```
#define HEX(
              n )
```

Value:

```
  ('0' + ((n)>>28 & 0xF)), \
  ('0' + ((n)>>24 & 0xF)), \
  ('0' + ((n)>>20 & 0xF)), \
  ('0' + ((n)>>16 & 0xF)), \
  ('0' + ((n)>>12 & 0xF)), \
  ('0' + ((n)>>8  & 0xF)), \
  ('0' + ((n)>>4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

Definition at line 731 of file CMakeCCompilerId.c.

5.1.1.7 PLATFORM_ID

```
#define PLATFORM_ID
```

Definition at line 558 of file CMakeCCompilerId.c.

5.1.1.8 STRINGIFY

```
#define STRINGIFY(
              X ) STRINGIFY_HELPER(X)
```

Definition at line 448 of file CMakeCCompilerId.c.

5.1.1.9 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
              X ) #X
```

Definition at line 447 of file CMakeCCompilerId.c.

5.1.2 Function Documentation

5.1.2.1 main()

```
int main (
             int argc,
             char * argv[] )
```

Definition at line 839 of file CMakeCCompilerId.c.

5.1.3 Variable Documentation

5.1.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

Definition at line 797 of file CMakeCCompilerId.c.

5.1.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

Definition at line 434 of file CMakeCCompilerId.c.

5.1.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Initial value:

```
= "INFO" ":" "extensions_default["



  "OFF"

"]"
```

Definition at line 821 of file CMakeCCompilerId.c.

5.1.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Initial value:

```
=
  "INFO" ":" "standard_default[" C_VERSION "]"
```

Definition at line 818 of file CMakeCCompilerId.c.

5.1.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

Definition at line 796 of file CMakeCCompilerId.c.

5.2 CMakeCCompilerId.c

Go to the documentation of this file.

```
00001 #ifdef __cplusplus
00002 # error "A C++ compiler has been selected for C."
00003 #endif
00004
00005 #if defined(__18CXX)
00006 # define ID_VOID_MAIN
00007 #endif
00008 #if defined(__CLASSIC_C__)
00009 /* cv-qualifiers did not exist in K&R C */
00010 # define const
00011 # define volatile
00012 #endif
00013
00014 #if !defined(__has_include)
00015 /* If the compiler does not have __has_include, pretend the answer is
00016    always no.  */
00017 #  define __has_include(x) 0
00018 #endif
00019
00020
00021 /* Version number components: V=Version, R=Revision, P=Patch
00022    Version date components:   YYYY=Year, MM=Month,   DD=Day  */
00023
00024 #if defined(__INTEL_COMPILER) || defined(__ICC)
00025 # define COMPILER_ID "Intel"
00026 # if defined(_MSC_VER)
00027 #  define SIMULATE_ID "MSVC"
00028 # endif
00029 # if defined(__GNUC__)
```

```
00030 #  define SIMULATE_ID "GNU"
00031 # endif
00032   /* __INTEL_COMPILER = VRP prior to 2021, and then VVVV for 2021 and later,
00033      except that a few beta releases use the old format with V=2021.  */
00034 # if __INTEL_COMPILER < 2021 || __INTEL_COMPILER == 202110 || __INTEL_COMPILER == 202111
00035 #  define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER/100)
00036 #  define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER/10 % 10)
00037 #  if defined(__INTEL_COMPILER_UPDATE)
00038 #   define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER_UPDATE)
00039 #  else
00040 #   define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER   % 10)
00041 #  endif
00042 # else
00043 #  define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER)
00044 #  define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER_UPDATE)
00045    /* The third version component from --version is an update index,
00046       but no macro is provided for it.  */
00047 #  define COMPILER_VERSION_PATCH DEC(0)
00048 # endif
00049 # if defined(__INTEL_COMPILER_BUILD_DATE)
00050    /* __INTEL_COMPILER_BUILD_DATE = YYYYMMDD */
00051 #  define COMPILER_VERSION_TWEAK DEC(__INTEL_COMPILER_BUILD_DATE)
00052 # endif
00053 # if defined(_MSC_VER)
00054    /* _MSC_VER = VVRR */
00055 #  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00056 #  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00057 # endif
00058 # if defined(__GNUC__)
00059 #  define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
00060 # elif defined(__GNUG__)
00061 #  define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
00062 # endif
00063 # if defined(__GNUC_MINOR__)
00064 #  define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
00065 # endif
00066 # if defined(__GNUC_PATCHLEVEL__)
00067 #  define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00068 # endif
00069
00070 #elif (defined(__clang__) && defined(__INTEL_CLANG_COMPILER)) || defined(__INTEL_LLVM_COMPILER)
00071 # define COMPILER_ID "IntelLLVM"
00072 #if defined(_MSC_VER)
00073 # define SIMULATE_ID "MSVC"
00074 #endif
00075 #if defined(__GNUC__)
00076 # define SIMULATE_ID "GNU"
00077 #endif
00078 /* __INTEL_LLVM_COMPILER = VVVVRP prior to 2021.2.0, VVVVRRPP for 2021.2.0 and
00079  * later.  Look for 6 digit vs. 8 digit version number to decide encoding.
00080  * VVVV is no smaller than the current year when a version is released.
00081  */
00082 #if __INTEL_LLVM_COMPILER < 1000000L
00083 # define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/100)
00084 # define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/10 % 10)
00085 # define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER    % 10)
00086 #else
00087 # define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/10000)
00088 # define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/100 % 100)
00089 # define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER     % 100)
00090 #endif
00091 #if defined(_MSC_VER)
00092   /* _MSC_VER = VVRR */
00093 # define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00094 # define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00095 #endif
00096 #if defined(__GNUC__)
00097 # define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
00098 #elif defined(__GNUG__)
00099 # define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
00100 #endif
00101 #if defined(__GNUC_MINOR__)
00102 # define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
00103 #endif
00104 #if defined(__GNUC_PATCHLEVEL__)
00105 # define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00106 #endif
00107
00108 #elif defined(__PATHCC__)
00109 # define COMPILER_ID "PathScale"
00110 # define COMPILER_VERSION_MAJOR DEC(__PATHCC__)
00111 # define COMPILER_VERSION_MINOR DEC(__PATHCC_MINOR__)
00112 # if defined(__PATHCC_PATCHLEVEL__)
00113 #  define COMPILER_VERSION_PATCH DEC(__PATHCC_PATCHLEVEL__)
00114 # endif
00115
00116 #elif defined(__BORLANDC__) && defined(__CODEGEARC_VERSION__)
```

```
00117 # define COMPILER_ID "Embarcadero"
00118 # define COMPILER_VERSION_MAJOR HEX(__CODEGEARC_VERSION__»24 & 0x00FF)
00119 # define COMPILER_VERSION_MINOR HEX(__CODEGEARC_VERSION__»16 & 0x00FF)
00120 # define COMPILER_VERSION_PATCH DEC(__CODEGEARC_VERSION__     & 0xFFFF)
00121
00122 #elif defined(__BORLANDC__)
00123 # define COMPILER_ID "Borland"
00124   /* __BORLANDC__ = 0xVRR */
00125 # define COMPILER_VERSION_MAJOR HEX(__BORLANDC__»8)
00126 # define COMPILER_VERSION_MINOR HEX(__BORLANDC__ & 0xFF)
00127
00128 #elif defined(__WATCOMC__) && __WATCOMC__ < 1200
00129 # define COMPILER_ID "Watcom"
00130    /* __WATCOMC__ = VVRR */
00131 # define COMPILER_VERSION_MAJOR DEC(__WATCOMC__ / 100)
00132 # define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
00133 # if (__WATCOMC__ % 10) > 0
00134 #  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
00135 # endif
00136
00137 #elif defined(__WATCOMC__)
00138 # define COMPILER_ID "OpenWatcom"
00139    /* __WATCOMC__ = VVRP + 1100 */
00140 # define COMPILER_VERSION_MAJOR DEC((__WATCOMC__ - 1100) / 100)
00141 # define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
00142 # if (__WATCOMC__ % 10) > 0
00143 #  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
00144 # endif
00145
00146 #elif defined(__SUNPRO_C)
00147 # define COMPILER_ID "SunPro"
00148 # if __SUNPRO_C >= 0x5100
00149    /* __SUNPRO_C = 0xVRRP */
00150 #  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_C»12)
00151 #  define COMPILER_VERSION_MINOR HEX(__SUNPRO_C»4 & 0xFF)
00152 #  define COMPILER_VERSION_PATCH HEX(__SUNPRO_C    & 0xF)
00153 # else
00154    /* __SUNPRO_CC = 0xVRP */
00155 #  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_C»8)
00156 #  define COMPILER_VERSION_MINOR HEX(__SUNPRO_C»4 & 0xF)
00157 #  define COMPILER_VERSION_PATCH HEX(__SUNPRO_C    & 0xF)
00158 # endif
00159
00160 #elif defined(__HP_cc)
00161 # define COMPILER_ID "HP"
00162   /* __HP_cc = VVRRPP */
00163 # define COMPILER_VERSION_MAJOR DEC(__HP_cc/10000)
00164 # define COMPILER_VERSION_MINOR DEC(__HP_cc/100 % 100)
00165 # define COMPILER_VERSION_PATCH DEC(__HP_cc     % 100)
00166
00167 #elif defined(__DECC)
00168 # define COMPILER_ID "Compaq"
00169   /* __DECC_VER = VVRRTPPPP */
00170 # define COMPILER_VERSION_MAJOR DEC(__DECC_VER/10000000)
00171 # define COMPILER_VERSION_MINOR DEC(__DECC_VER/100000  % 100)
00172 # define COMPILER_VERSION_PATCH DEC(__DECC_VER         % 10000)
00173
00174 #elif defined(__IBMC__) && defined(__COMPILER_VER__)
00175 # define COMPILER_ID "zOS"
00176   /* __IBMC__ = VRP */
00177 # define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
00178 # define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
00179 # define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)
00180
00181 #elif defined(__open_xl__) && defined(__clang__)
00182 # define COMPILER_ID "IBMClang"
00183 # define COMPILER_VERSION_MAJOR DEC(__open_xl_version__)
00184 # define COMPILER_VERSION_MINOR DEC(__open_xl_release__)
00185 # define COMPILER_VERSION_PATCH DEC(__open_xl_modification__)
00186 # define COMPILER_VERSION_TWEAK DEC(__open_xl_ptf_fix_level__)
00187
00188
00189 #elif defined(__ibmxl__) && defined(__clang__)
00190 # define COMPILER_ID "XLClang"
00191 # define COMPILER_VERSION_MAJOR DEC(__ibmxl_version__)
00192 # define COMPILER_VERSION_MINOR DEC(__ibmxl_release__)
00193 # define COMPILER_VERSION_PATCH DEC(__ibmxl_modification__)
00194 # define COMPILER_VERSION_TWEAK DEC(__ibmxl_ptf_fix_level__)
00195
00196
00197 #elif defined(__IBMC__) && !defined(__COMPILER_VER__) && __IBMC__ >= 800
00198 # define COMPILER_ID "XL"
00199   /* __IBMC__ = VRP */
00200 # define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
00201 # define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
00202 # define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)
00203
```

```
00204 #elif defined(__IBMC__) && !defined(__COMPILER_VER__) && __IBMC__ < 800
00205 # define COMPILER_ID "VisualAge"
00206   /* __IBMC__ = VRP */
00207 # define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
00208 # define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
00209 # define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)
00210
00211 #elif defined(__NVCOMPILER)
00212 # define COMPILER_ID "NVHPC"
00213 # define COMPILER_VERSION_MAJOR DEC(__NVCOMPILER_MAJOR__)
00214 # define COMPILER_VERSION_MINOR DEC(__NVCOMPILER_MINOR__)
00215 # if defined(__NVCOMPILER_PATCHLEVEL__)
00216 #  define COMPILER_VERSION_PATCH DEC(__NVCOMPILER_PATCHLEVEL__)
00217 # endif
00218
00219 #elif defined(__PGI)
00220 # define COMPILER_ID "PGI"
00221 # define COMPILER_VERSION_MAJOR DEC(__PGIC__)
00222 # define COMPILER_VERSION_MINOR DEC(__PGIC_MINOR__)
00223 # if defined(__PGIC_PATCHLEVEL__)
00224 #  define COMPILER_VERSION_PATCH DEC(__PGIC_PATCHLEVEL__)
00225 # endif
00226
00227 #elif defined(_CRAYC)
00228 # define COMPILER_ID "Cray"
00229 # define COMPILER_VERSION_MAJOR DEC(_RELEASE_MAJOR)
00230 # define COMPILER_VERSION_MINOR DEC(_RELEASE_MINOR)
00231
00232 #elif defined(__TI_COMPILER_VERSION__)
00233 # define COMPILER_ID "TI"
00234   /* __TI_COMPILER_VERSION__ = VVVRRRPPP */
00235 # define COMPILER_VERSION_MAJOR DEC(__TI_COMPILER_VERSION__/1000000)
00236 # define COMPILER_VERSION_MINOR DEC(__TI_COMPILER_VERSION__/1000   % 1000)
00237 # define COMPILER_VERSION_PATCH DEC(__TI_COMPILER_VERSION__        % 1000)
00238
00239 #elif defined(__CLANG_FUJITSU)
00240 # define COMPILER_ID "FujitsuClang"
00241 # define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
00242 # define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
00243 # define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
00244 # define COMPILER_VERSION_INTERNAL_STR __clang_version__
00245
00246
00247 #elif defined(__FUJITSU)
00248 # define COMPILER_ID "Fujitsu"
00249 # if defined(__FCC_version__)
00250 #   define COMPILER_VERSION __FCC_version__
00251 # elif defined(__FCC_major__)
00252 #   define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
00253 #   define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
00254 #   define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
00255 # endif
00256 # if defined(__fcc_version)
00257 #   define COMPILER_VERSION_INTERNAL DEC(__fcc_version)
00258 # elif defined(__FCC_VERSION)
00259 #   define COMPILER_VERSION_INTERNAL DEC(__FCC_VERSION)
00260 # endif
00261
00262
00263 #elif defined(__ghs__)
00264 # define COMPILER_ID "GHS"
00265 /* __GHS_VERSION_NUMBER = VVVVRP */
00266 # ifdef __GHS_VERSION_NUMBER
00267 # define COMPILER_VERSION_MAJOR DEC(__GHS_VERSION_NUMBER / 100)
00268 # define COMPILER_VERSION_MINOR DEC(__GHS_VERSION_NUMBER / 10 % 10)
00269 # define COMPILER_VERSION_PATCH DEC(__GHS_VERSION_NUMBER      % 10)
00270 # endif
00271
00272 #elif defined(__TASKING__)
00273 # define COMPILER_ID "Tasking"
00274   # define COMPILER_VERSION_MAJOR DEC(__VERSION__/1000)
00275   # define COMPILER_VERSION_MINOR DEC(__VERSION__ % 100)
00276 # define COMPILER_VERSION_INTERNAL DEC(__VERSION__)
00277
00278 #elif defined(__TINYC__)
00279 # define COMPILER_ID "TinyCC"
00280
00281 #elif defined(__BCC__)
00282 # define COMPILER_ID "Bruce"
00283
00284 #elif defined(__SCO_VERSION__)
00285 # define COMPILER_ID "SCO"
00286
00287 #elif defined(__ARMCC_VERSION) && !defined(__clang__)
00288 # define COMPILER_ID "ARMCC"
00289 #if __ARMCC_VERSION >= 1000000
00290   /* __ARMCC_VERSION = VRRPPPP */
```

```
00291   # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/1000000)
00292   # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 100)
00293   # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION     % 10000)
00294 #else
00295   /* __ARMCC_VERSION = VRPPPP */
00296   # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/100000)
00297   # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 10)
00298   # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION    % 10000)
00299 #endif
00300
00301
00302 #elif defined(__clang__) && defined(__apple_build_version__)
00303 # define COMPILER_ID "AppleClang"
00304 # if defined(_MSC_VER)
00305 #  define SIMULATE_ID "MSVC"
00306 # endif
00307 # define COMPILER_VERSION_MAJOR DEC(__clang_major__)
00308 # define COMPILER_VERSION_MINOR DEC(__clang_minor__)
00309 # define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
00310 # if defined(_MSC_VER)
00311   /* _MSC_VER = VVRR */
00312 #  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00313 #  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00314 # endif
00315 # define COMPILER_VERSION_TWEAK DEC(__apple_build_version__)
00316
00317 #elif defined(__clang__) && defined(__ARMCOMPILER_VERSION)
00318 # define COMPILER_ID "ARMClang"
00319   # define COMPILER_VERSION_MAJOR DEC(__ARMCOMPILER_VERSION/1000000)
00320   # define COMPILER_VERSION_MINOR DEC(__ARMCOMPILER_VERSION/10000 % 100)
00321   # define COMPILER_VERSION_PATCH DEC(__ARMCOMPILER_VERSION     % 10000)
00322 # define COMPILER_VERSION_INTERNAL DEC(__ARMCOMPILER_VERSION)
00323
00324 #elif defined(__clang__)
00325 # define COMPILER_ID "Clang"
00326 # if defined(_MSC_VER)
00327 #  define SIMULATE_ID "MSVC"
00328 # endif
00329 # define COMPILER_VERSION_MAJOR DEC(__clang_major__)
00330 # define COMPILER_VERSION_MINOR DEC(__clang_minor__)
00331 # define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
00332 # if defined(_MSC_VER)
00333   /* _MSC_VER = VVRR */
00334 #  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00335 #  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00336 # endif
00337
00338 #elif defined(__LCC__) && (defined(__GNUC__) || defined(__GNUG__) || defined(__MCST__))
00339 # define COMPILER_ID "LCC"
00340 # define COMPILER_VERSION_MAJOR DEC(__LCC__ / 100)
00341 # define COMPILER_VERSION_MINOR DEC(__LCC__ % 100)
00342 # if defined(__LCC_MINOR__)
00343 #  define COMPILER_VERSION_PATCH DEC(__LCC_MINOR__)
00344 # endif
00345 # if defined(__GNUC__) && defined(__GNUC_MINOR__)
00346 #  define SIMULATE_ID "GNU"
00347 #  define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
00348 #  define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
00349 #  if defined(__GNUC_PATCHLEVEL__)
00350 #   define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00351 #  endif
00352 # endif
00353
00354 #elif defined(__GNUC__)
00355 # define COMPILER_ID "GNU"
00356 # define COMPILER_VERSION_MAJOR DEC(__GNUC__)
00357 # if defined(__GNUC_MINOR__)
00358 #  define COMPILER_VERSION_MINOR DEC(__GNUC_MINOR__)
00359 # endif
00360 # if defined(__GNUC_PATCHLEVEL__)
00361 #  define COMPILER_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00362 # endif
00363
00364 #elif defined(_MSC_VER)
00365 # define COMPILER_ID "MSVC"
00366   /* _MSC_VER = VVRR */
00367 # define COMPILER_VERSION_MAJOR DEC(_MSC_VER / 100)
00368 # define COMPILER_VERSION_MINOR DEC(_MSC_VER % 100)
00369 # if defined(_MSC_FULL_VER)
00370 #  if _MSC_VER >= 1400
00371     /* _MSC_FULL_VER = VVRRPPPPP */
00372 #   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 100000)
00373 #  else
00374     /* _MSC_FULL_VER = VVRRPPPP */
00375 #   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 10000)
00376 #  endif
00377 # endif
```

```
00378 # if defined(_MSC_BUILD)
00379 #  define COMPILER_VERSION_TWEAK DEC(_MSC_BUILD)
00380 # endif
00381
00382 #elif defined(_ADI_COMPILER)
00383 # define COMPILER_ID "ADSP"
00384 #if defined(__VERSIONNUM__)
00385   /* __VERSIONNUM__ = 0xVVRRPPTT */
00386 #  define COMPILER_VERSION_MAJOR DEC(__VERSIONNUM__ » 24 & 0xFF)
00387 #  define COMPILER_VERSION_MINOR DEC(__VERSIONNUM__ » 16 & 0xFF)
00388 #  define COMPILER_VERSION_PATCH DEC(__VERSIONNUM__ » 8 & 0xFF)
00389 #  define COMPILER_VERSION_TWEAK DEC(__VERSIONNUM__ & 0xFF)
00390 #endif
00391
00392 #elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
00393 # define COMPILER_ID "IAR"
00394 # if defined(__VER__) && defined(__ICCARM__)
00395 #  define COMPILER_VERSION_MAJOR DEC((__VER__) / 1000000)
00396 #  define COMPILER_VERSION_MINOR DEC(((__VER__) / 1000) % 1000)
00397 #  define COMPILER_VERSION_PATCH DEC((__VER__) % 1000)
00398 #  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
00399 # elif defined(__VER__) && (defined(__ICCAVR__) || defined(__ICCRX__) || defined(__ICCRH850__) ||
      defined(__ICCRL78__) || defined(__ICC430__) || defined(__ICCRISCV__) || defined(__ICCV850__) ||
      defined(__ICC8051__) || defined(__ICCSTM8__))
00400 #  define COMPILER_VERSION_MAJOR DEC((__VER__) / 100)
00401 #  define COMPILER_VERSION_MINOR DEC((__VER__) - (((__VER__) / 100)*100))
00402 #  define COMPILER_VERSION_PATCH DEC(__SUBVERSION__)
00403 #  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
00404 # endif
00405
00406 #elif defined(__SDCC_VERSION_MAJOR) || defined(SDCC)
00407 # define COMPILER_ID "SDCC"
00408 # if defined(__SDCC_VERSION_MAJOR)
00409 #  define COMPILER_VERSION_MAJOR DEC(__SDCC_VERSION_MAJOR)
00410 #  define COMPILER_VERSION_MINOR DEC(__SDCC_VERSION_MINOR)
00411 #  define COMPILER_VERSION_PATCH DEC(__SDCC_VERSION_PATCH)
00412 # else
00413   /* SDCC = VRP */
00414 #  define COMPILER_VERSION_MAJOR DEC(SDCC/100)
00415 #  define COMPILER_VERSION_MINOR DEC(SDCC/10 % 10)
00416 #  define COMPILER_VERSION_PATCH DEC(SDCC    % 10)
00417 # endif
00418
00419
00420 /* These compilers are either not known or too old to define an
00421   identification macro.  Try to identify the platform and guess that
00422   it is the native compiler.  */
00423 #elif defined(__hpux) || defined(__hpua)
00424 # define COMPILER_ID "HP"
00425
00426 #else /* unknown compiler */
00427 # define COMPILER_ID ""
00428 #endif
00429
00430 /* Construct the string literal in pieces to prevent the source from
00431    getting matched.  Store it in a pointer rather than an array
00432    because some compilers will just produce instructions to fill the
00433    array rather than assigning a pointer to a static array.  */
00434 char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]";
00435 #ifdef SIMULATE_ID
00436 char const* info_simulate = "INFO" ":" "simulate[" SIMULATE_ID "]";
00437 #endif
00438
00439 #ifdef __QNXNTO__
00440 char const* qnxnto = "INFO" ":" "qnxnto[]";
00441 #endif
00442
00443 #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
00444 char const *info_cray = "INFO" ":" "compiler_wrapper[CrayPrgEnv]";
00445 #endif
00446
00447 #define STRINGIFY_HELPER(X) #X
00448 #define STRINGIFY(X) STRINGIFY_HELPER(X)
00449
00450 /* Identify known platforms by name.  */
00451 #if defined(__linux) || defined(__linux__) || defined(linux)
00452 # define PLATFORM_ID "Linux"
00453
00454 #elif defined(__MSYS__)
00455 # define PLATFORM_ID "MSYS"
00456
00457 #elif defined(__CYGWIN__)
00458 # define PLATFORM_ID "Cygwin"
00459
00460 #elif defined(__MINGW32__)
00461 # define PLATFORM_ID "MinGW"
00462
```

```
00463 #elif defined(__APPLE__)
00464 # define PLATFORM_ID "Darwin"
00465
00466 #elif defined(_WIN32) || defined(__WIN32__) || defined(WIN32)
00467 # define PLATFORM_ID "Windows"
00468
00469 #elif defined(__FreeBSD__) || defined(__FreeBSD)
00470 # define PLATFORM_ID "FreeBSD"
00471
00472 #elif defined(__NetBSD__) || defined(__NetBSD)
00473 # define PLATFORM_ID "NetBSD"
00474
00475 #elif defined(__OpenBSD__) || defined(__OPENBSD)
00476 # define PLATFORM_ID "OpenBSD"
00477
00478 #elif defined(__sun) || defined(sun)
00479 # define PLATFORM_ID "SunOS"
00480
00481 #elif defined(_AIX) || defined(__AIX) || defined(__AIX__) || defined(__aix) || defined(__aix__)
00482 # define PLATFORM_ID "AIX"
00483
00484 #elif defined(__hpux) || defined(__hpux__)
00485 # define PLATFORM_ID "HP-UX"
00486
00487 #elif defined(__HAIKU__)
00488 # define PLATFORM_ID "Haiku"
00489
00490 #elif defined(__BeOS) || defined(__BEOS__) || defined(_BEOS)
00491 # define PLATFORM_ID "BeOS"
00492
00493 #elif defined(__QNX__) || defined(__QNXNTO__)
00494 # define PLATFORM_ID "QNX"
00495
00496 #elif defined(__tru64) || defined(_tru64) || defined(__TRU64__)
00497 # define PLATFORM_ID "Tru64"
00498
00499 #elif defined(__riscos) || defined(__riscos__)
00500 # define PLATFORM_ID "RISCos"
00501
00502 #elif defined(__sinix) || defined(__sinix__) || defined(__SINIX__)
00503 # define PLATFORM_ID "SINIX"
00504
00505 #elif defined(__UNIX_SV__)
00506 # define PLATFORM_ID "UNIX_SV"
00507
00508 #elif defined(__bsdos__)
00509 # define PLATFORM_ID "BSDOS"
00510
00511 #elif defined(_MPRAS) || defined(MPRAS)
00512 # define PLATFORM_ID "MP-RAS"
00513
00514 #elif defined(__osf) || defined(__osf__)
00515 # define PLATFORM_ID "OSF1"
00516
00517 #elif defined(_SCO_SV) || defined(SCO_SV) || defined(sco_sv)
00518 # define PLATFORM_ID "SCO_SV"
00519
00520 #elif defined(__ultrix) || defined(__ultrix__) || defined(_ULTRIX)
00521 # define PLATFORM_ID "ULTRIX"
00522
00523 #elif defined(__XENIX__) || defined(_XENIX) || defined(XENIX)
00524 # define PLATFORM_ID "Xenix"
00525
00526 #elif defined(__WATCOMC__)
00527 # if defined(__LINUX__)
00528 #  define PLATFORM_ID "Linux"
00529
00530 # elif defined(__DOS__)
00531 #  define PLATFORM_ID "DOS"
00532
00533 # elif defined(__OS2__)
00534 #  define PLATFORM_ID "OS2"
00535
00536 # elif defined(__WINDOWS__)
00537 #  define PLATFORM_ID "Windows3x"
00538
00539 # elif defined(__VXWORKS__)
00540 #  define PLATFORM_ID "VxWorks"
00541
00542 # else /* unknown platform */
00543 #  define PLATFORM_ID
00544 # endif
00545
00546 #elif defined(__INTEGRITY)
00547 # if defined(INT_178B)
00548 #  define PLATFORM_ID "Integrity178"
00549
```

```
00550 # else /* regular Integrity */
00551 #  define PLATFORM_ID "Integrity"
00552 # endif
00553
00554 # elif defined(_ADI_COMPILER)
00555 #  define PLATFORM_ID "ADSP"
00556
00557 #else /* unknown platform */
00558 # define PLATFORM_ID
00559
00560 #endif
00561
00562 /* For windows compilers MSVC and Intel we can determine
00563    the architecture of the compiler being used.  This is because
00564    the compilers do not have flags that can change the architecture,
00565    but rather depend on which compiler is being used
00566 */
00567 #if defined(_WIN32) && defined(_MSC_VER)
00568 # if defined(_M_IA64)
00569 #  define ARCHITECTURE_ID "IA64"
00570
00571 # elif defined(_M_ARM64EC)
00572 #  define ARCHITECTURE_ID "ARM64EC"
00573
00574 # elif defined(_M_X64) || defined(_M_AMD64)
00575 #  define ARCHITECTURE_ID "x64"
00576
00577 # elif defined(_M_IX86)
00578 #  define ARCHITECTURE_ID "X86"
00579
00580 # elif defined(_M_ARM64)
00581 #  define ARCHITECTURE_ID "ARM64"
00582
00583 # elif defined(_M_ARM)
00584 #  if _M_ARM == 4
00585 #   define ARCHITECTURE_ID "ARMV4I"
00586 #  elif _M_ARM == 5
00587 #   define ARCHITECTURE_ID "ARMV5I"
00588 #  else
00589 #   define ARCHITECTURE_ID "ARMV" STRINGIFY(_M_ARM)
00590 #  endif
00591
00592 # elif defined(_M_MIPS)
00593 #  define ARCHITECTURE_ID "MIPS"
00594
00595 # elif defined(_M_SH)
00596 #  define ARCHITECTURE_ID "SHx"
00597
00598 # else /* unknown architecture */
00599 #  define ARCHITECTURE_ID ""
00600 # endif
00601
00602 #elif defined(__WATCOMC__)
00603 # if defined(_M_I86)
00604 #  define ARCHITECTURE_ID "I86"
00605
00606 # elif defined(_M_IX86)
00607 #  define ARCHITECTURE_ID "X86"
00608
00609 # else /* unknown architecture */
00610 #  define ARCHITECTURE_ID ""
00611 # endif
00612
00613 #elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
00614 # if defined(__ICCARM__)
00615 #  define ARCHITECTURE_ID "ARM"
00616
00617 # elif defined(__ICCRX__)
00618 #  define ARCHITECTURE_ID "RX"
00619
00620 # elif defined(__ICCRH850__)
00621 #  define ARCHITECTURE_ID "RH850"
00622
00623 # elif defined(__ICCRL78__)
00624 #  define ARCHITECTURE_ID "RL78"
00625
00626 # elif defined(__ICCRISCV__)
00627 #  define ARCHITECTURE_ID "RISCV"
00628
00629 # elif defined(__ICCAVR__)
00630 #  define ARCHITECTURE_ID "AVR"
00631
00632 # elif defined(__ICC430__)
00633 #  define ARCHITECTURE_ID "MSP430"
00634
00635 # elif defined(__ICCV850__)
00636 #  define ARCHITECTURE_ID "V850"
```

```
00637
00638 # elif defined(__ICC8051__)
00639 #  define ARCHITECTURE_ID "8051"
00640
00641 # elif defined(__ICCSTM8__)
00642 #  define ARCHITECTURE_ID "STM8"
00643
00644 # else /* unknown architecture */
00645 #  define ARCHITECTURE_ID ""
00646 # endif
00647
00648 #elif defined(__ghs__)
00649 # if defined(__PPC64__)
00650 #  define ARCHITECTURE_ID "PPC64"
00651
00652 # elif defined(__ppc__)
00653 #  define ARCHITECTURE_ID "PPC"
00654
00655 # elif defined(__ARM__)
00656 #  define ARCHITECTURE_ID "ARM"
00657
00658 # elif defined(__x86_64__)
00659 #  define ARCHITECTURE_ID "x64"
00660
00661 # elif defined(__i386__)
00662 #  define ARCHITECTURE_ID "X86"
00663
00664 # else /* unknown architecture */
00665 #  define ARCHITECTURE_ID ""
00666 # endif
00667
00668 #elif defined(__TI_COMPILER_VERSION__)
00669 # if defined(__TI_ARM__)
00670 #  define ARCHITECTURE_ID "ARM"
00671
00672 # elif defined(__MSP430__)
00673 #  define ARCHITECTURE_ID "MSP430"
00674
00675 # elif defined(__TMS320C28XX__)
00676 #  define ARCHITECTURE_ID "TMS320C28x"
00677
00678 # elif defined(__TMS320C6X__) || defined(_TMS320C6X)
00679 #  define ARCHITECTURE_ID "TMS320C6x"
00680
00681 # else /* unknown architecture */
00682 #  define ARCHITECTURE_ID ""
00683 # endif
00684
00685 # elif defined(__ADSPSHARC__)
00686 #  define ARCHITECTURE_ID "SHARC"
00687
00688 # elif defined(__ADSPBLACKFIN__)
00689 #  define ARCHITECTURE_ID "Blackfin"
00690
00691 #elif defined(__TASKING__)
00692
00693 # if defined(__CTC__) || defined(__CPTC__)
00694 #  define ARCHITECTURE_ID "TriCore"
00695
00696 # elif defined(__CMCS__)
00697 #  define ARCHITECTURE_ID "MCS"
00698
00699 # elif defined(__CARM__)
00700 #  define ARCHITECTURE_ID "ARM"
00701
00702 # elif defined(__CARC__)
00703 #  define ARCHITECTURE_ID "ARC"
00704
00705 # elif defined(__C51__)
00706 #  define ARCHITECTURE_ID "8051"
00707
00708 # elif defined(__CPCP__)
00709 #  define ARCHITECTURE_ID "PCP"
00710
00711 # else
00712 #  define ARCHITECTURE_ID ""
00713 # endif
00714
00715 #else
00716 #  define ARCHITECTURE_ID
00717 #endif
00718
00719 /* Convert integer to decimal digit literals.  */
00720 #define DEC(n)                   \
00721   ('0' + (((n) / 10000000)%10)), \
00722   ('0' + (((n) / 1000000)%10)),  \
00723   ('0' + (((n) / 100000)%10)),   \
```

```
00724   ('0' + (((n) / 10000)%10)),     \
00725   ('0' + (((n) / 1000)%10)),      \
00726   ('0' + (((n) / 100)%10)),       \
00727   ('0' + (((n) / 10)%10)),        \
00728   ('0' +  ((n) % 10))
00729
00730 /* Convert integer to hex digit literals.  */
00731 #define HEX(n)             \
00732   ('0' + ((n)>>28 & 0xF)), \
00733   ('0' + ((n)>>24 & 0xF)), \
00734   ('0' + ((n)>>20 & 0xF)), \
00735   ('0' + ((n)>>16 & 0xF)), \
00736   ('0' + ((n)>>12 & 0xF)), \
00737   ('0' + ((n)>>8  & 0xF)), \
00738   ('0' + ((n)>>4  & 0xF)), \
00739   ('0' + ((n)     & 0xF))
00740
00741 /* Construct a string literal encoding the version number. */
00742 #ifdef COMPILER_VERSION
00743 char const* info_version = "INFO" ":" "compiler_version[" COMPILER_VERSION "]";
00744
00745 /* Construct a string literal encoding the version number components. */
00746 #elif defined(COMPILER_VERSION_MAJOR)
00747 char const info_version[] = {
00748   'I', 'N', 'F', 'O', ':',
00749   'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','[',
00750   COMPILER_VERSION_MAJOR,
00751 # ifdef COMPILER_VERSION_MINOR
00752   '.', COMPILER_VERSION_MINOR,
00753 #  ifdef COMPILER_VERSION_PATCH
00754   '.', COMPILER_VERSION_PATCH,
00755 #   ifdef COMPILER_VERSION_TWEAK
00756     '.', COMPILER_VERSION_TWEAK,
00757 #   endif
00758 #  endif
00759 # endif
00760   ']','\0'};
00761 #endif
00762
00763 /* Construct a string literal encoding the internal version number. */
00764 #ifdef COMPILER_VERSION_INTERNAL
00765 char const info_version_internal[] = {
00766   'I', 'N', 'F', 'O', ':',
00767   'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','_',
00768   'i','n','t','e','r','n','a','l','[',
00769   COMPILER_VERSION_INTERNAL,']','\0'};
00770 #elif defined(COMPILER_VERSION_INTERNAL_STR)
00771 char const* info_version_internal = "INFO" ":" "compiler_version_internal["
      COMPILER_VERSION_INTERNAL_STR "]";
00772 #endif
00773
00774 /* Construct a string literal encoding the version number components. */
00775 #ifdef SIMULATE_VERSION_MAJOR
00776 char const info_simulate_version[] = {
00777   'I', 'N', 'F', 'O', ':',
00778   's','i','m','u','l','a','t','e','_','v','e','r','s','i','o','n','[',
00779   SIMULATE_VERSION_MAJOR,
00780 # ifdef SIMULATE_VERSION_MINOR
00781   '.', SIMULATE_VERSION_MINOR,
00782 #  ifdef SIMULATE_VERSION_PATCH
00783   '.', SIMULATE_VERSION_PATCH,
00784 #   ifdef SIMULATE_VERSION_TWEAK
00785     '.', SIMULATE_VERSION_TWEAK,
00786 #   endif
00787 #  endif
00788 # endif
00789   ']','\0'};
00790 #endif
00791
00792 /* Construct the string literal in pieces to prevent the source from
00793    getting matched.  Store it in a pointer rather than an array
00794    because some compilers will just produce instructions to fill the
00795    array rather than assigning a pointer to a static array.  */
00796 char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]";
00797 char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]";
00798
00799
00800
00801 #if !defined(__STDC__) && !defined(__clang__)
00802 # if defined(_MSC_VER) || defined(__ibmxl__) || defined(__IBMC__)
00803 #  define C_VERSION "90"
00804 # else
00805 #  define C_VERSION
00806 # endif
00807 #elif __STDC_VERSION__ > 201710L
00808 # define C_VERSION "23"
00809 #elif __STDC_VERSION__ >= 201710L
```

```
00810 # define C_VERSION "17"
00811 #elif __STDC_VERSION__ >= 201000L
00812 # define C_VERSION "11"
00813 #elif __STDC_VERSION__ >= 199901L
00814 # define C_VERSION "99"
00815 #else
00816 # define C_VERSION "90"
00817 #endif
00818 const char* info_language_standard_default =
00819   "INFO" ":" "standard_default[" C_VERSION "]";
00820
00821 const char* info_language_extensions_default = "INFO" ":" "extensions_default["
00822 #if (defined(__clang__) || defined(__GNUC__) || defined(__xlC__) ||           \
00823      defined(__TI_COMPILER_VERSION__)) &&                                     \
00824   !defined(__STRICT_ANSI__)
00825   "ON"
00826 #else
00827   "OFF"
00828 #endif
00829 "]";
00830
00831 /*--------------------------------------------------------------------------*/
00832
00833 #ifdef ID_VOID_MAIN
00834 void main() {}
00835 #else
00836 # if defined(__CLASSIC_C__)
00837 int main(argc, argv) int argc; char *argv[];
00838 # else
00839 int main(int argc, char* argv[])
00840 # endif
00841 {
00842   int require = 0;
00843   require += info_compiler[argc];
00844   require += info_platform[argc];
00845   require += info_arch[argc];
00846 #ifdef COMPILER_VERSION_MAJOR
00847   require += info_version[argc];
00848 #endif
00849 #ifdef COMPILER_VERSION_INTERNAL
00850   require += info_version_internal[argc];
00851 #endif
00852 #ifdef SIMULATE_ID
00853   require += info_simulate[argc];
00854 #endif
00855 #ifdef SIMULATE_VERSION_MAJOR
00856   require += info_simulate_version[argc];
00857 #endif
00858 #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
00859   require += info_cray[argc];
00860 #endif
00861   require += info_language_standard_default[argc];
00862   require += info_language_extensions_default[argc];
00863   (void)argv;
00864   return require;
00865 }
00866 #endif
```

5.3 cmake-build-release/CMakeFiles/3.26.4/CompilerIdC/CMakeCCompilerId.c File Reference

Macros

- #define __has_include(x) 0
- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X) #X
- #define STRINGIFY(X) STRINGIFY_HELPER(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define C_VERSION

Functions

- int main (int argc, char ∗argv[ ])

Variables

- char const ∗ info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const ∗ info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const ∗ info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char ∗ info_language_standard_default
- const char ∗ info_language_extensions_default

5.3.1 Macro Definition Documentation

5.3.1.1 __has_include

```
#define __has_include(
              x ) 0
```

Definition at line 17 of file CMakeCCompilerId.c.

5.3.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

Definition at line 716 of file CMakeCCompilerId.c.

5.3.1.3 C_VERSION

```
#define C_VERSION
```

Definition at line 805 of file CMakeCCompilerId.c.

5.3.1.4 COMPILER_ID

```
#define COMPILER_ID ""
```

Definition at line 427 of file CMakeCCompilerId.c.

5.3.1.5 DEC

```
#define DEC(
              n )
```

Value:

```
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

Definition at line 720 of file CMakeCCompilerId.c.

5.3.1.6 HEX

```
#define HEX(
              n )
```

Value:

```
  ('0' + ((n)»28 & 0xF)), \
  ('0' + ((n)»24 & 0xF)), \
  ('0' + ((n)»20 & 0xF)), \
  ('0' + ((n)»16 & 0xF)), \
  ('0' + ((n)»12 & 0xF)), \
  ('0' + ((n)»8  & 0xF)), \
  ('0' + ((n)»4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

Definition at line 731 of file CMakeCCompilerId.c.

5.3.1.7 PLATFORM_ID

```
#define PLATFORM_ID
```

Definition at line 558 of file CMakeCCompilerId.c.

5.3.1.8 STRINGIFY

```
#define STRINGIFY(
              X ) STRINGIFY_HELPER(X)
```

Definition at line 448 of file CMakeCCompilerId.c.

5.3.1.9 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
              X ) #X
```

Definition at line 447 of file CMakeCCompilerId.c.

5.3.2 Function Documentation

5.3.2.1 main()

```
int main (
             int argc,
             char * argv[] )
```

Definition at line 839 of file CMakeCCompilerId.c.

5.3.3 Variable Documentation

5.3.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

Definition at line 797 of file CMakeCCompilerId.c.

5.3.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

Definition at line 434 of file CMakeCCompilerId.c.

5.3.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Initial value:

```
= "INFO" ":" "extensions_default["



  "OFF"

"]"
```

Definition at line 821 of file CMakeCCompilerId.c.

5.3.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Initial value:

```
=
  "INFO" ":" "standard_default[" C_VERSION "]"
```

Definition at line 818 of file CMakeCCompilerId.c.

5.3.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

Definition at line 796 of file CMakeCCompilerId.c.

5.4 CMakeCCompilerId.c

Go to the documentation of this file.

```
00001 #ifdef __cplusplus
00002 # error "A C++ compiler has been selected for C."
00003 #endif
00004
00005 #if defined(__18CXX)
00006 # define ID_VOID_MAIN
00007 #endif
00008 #if defined(__CLASSIC_C__)
00009 /* cv-qualifiers did not exist in K&R C */
00010 # define const
00011 # define volatile
00012 #endif
00013
00014 #if !defined(__has_include)
00015 /* If the compiler does not have __has_include, pretend the answer is
00016    always no.  */
00017 #  define __has_include(x) 0
00018 #endif
```

```
00019
00020
00021 /* Version number components: V=Version, R=Revision, P=Patch
00022    Version date components:   YYYY=Year, MM=Month,   DD=Day  */
00023
00024 #if defined(__INTEL_COMPILER) || defined(__ICC)
00025 # define COMPILER_ID "Intel"
00026 # if defined(_MSC_VER)
00027 #  define SIMULATE_ID "MSVC"
00028 # endif
00029 # if defined(__GNUC__)
00030 #  define SIMULATE_ID "GNU"
00031 # endif
00032   /* __INTEL_COMPILER = VRP prior to 2021, and then VVVV for 2021 and later,
00033      except that a few beta releases use the old format with V=2021.  */
00034 # if __INTEL_COMPILER < 2021 || __INTEL_COMPILER == 202110 || __INTEL_COMPILER == 202111
00035 #  define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER/100)
00036 #  define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER/10 % 10)
00037 #  if defined(__INTEL_COMPILER_UPDATE)
00038 #   define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER_UPDATE)
00039 #  else
00040 #   define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER   % 10)
00041 #  endif
00042 # else
00043 #  define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER)
00044 #  define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER_UPDATE)
00045    /* The third version component from --version is an update index,
00046       but no macro is provided for it.  */
00047 #  define COMPILER_VERSION_PATCH DEC(0)
00048 # endif
00049 # if defined(__INTEL_COMPILER_BUILD_DATE)
00050    /* __INTEL_COMPILER_BUILD_DATE = YYYYMMDD */
00051 #  define COMPILER_VERSION_TWEAK DEC(__INTEL_COMPILER_BUILD_DATE)
00052 # endif
00053 # if defined(_MSC_VER)
00054    /* _MSC_VER = VVRR */
00055 #  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00056 #  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00057 # endif
00058 # if defined(__GNUC__)
00059 #  define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
00060 # elif defined(__GNUG__)
00061 #  define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
00062 # endif
00063 # if defined(__GNUC_MINOR__)
00064 #  define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
00065 # endif
00066 # if defined(__GNUC_PATCHLEVEL__)
00067 #  define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00068 # endif
00069
00070 #elif (defined(__clang__) && defined(__INTEL_CLANG_COMPILER)) || defined(__INTEL_LLVM_COMPILER)
00071 # define COMPILER_ID "IntelLLVM"
00072 #if defined(_MSC_VER)
00073 # define SIMULATE_ID "MSVC"
00074 #endif
00075 #if defined(__GNUC__)
00076 # define SIMULATE_ID "GNU"
00077 #endif
00078 /* __INTEL_LLVM_COMPILER = VVVVRP prior to 2021.2.0, VVVVRRPP for 2021.2.0 and
00079  * later.  Look for 6 digit vs. 8 digit version number to decide encoding.
00080  * VVVV is no smaller than the current year when a version is released.
00081  */
00082 #if __INTEL_LLVM_COMPILER < 1000000L
00083 # define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/100)
00084 # define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/10 % 10)
00085 # define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER    % 10)
00086 #else
00087 # define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/10000)
00088 # define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/100 % 100)
00089 # define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER     % 100)
00090 #endif
00091 #if defined(_MSC_VER)
00092   /* _MSC_VER = VVRR */
00093 # define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00094 # define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00095 #endif
00096 #if defined(__GNUC__)
00097 # define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
00098 #elif defined(__GNUG__)
00099 # define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
00100 #endif
00101 #if defined(__GNUC_MINOR__)
00102 # define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
00103 #endif
00104 #if defined(__GNUC_PATCHLEVEL__)
00105 # define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
```

```
00106 #endif
00107
00108 #elif defined(__PATHCC__)
00109 # define COMPILER_ID "PathScale"
00110 # define COMPILER_VERSION_MAJOR DEC(__PATHCC__)
00111 # define COMPILER_VERSION_MINOR DEC(__PATHCC_MINOR__)
00112 # if defined(__PATHCC_PATCHLEVEL__)
00113 #  define COMPILER_VERSION_PATCH DEC(__PATHCC_PATCHLEVEL__)
00114 # endif
00115
00116 #elif defined(__BORLANDC__) && defined(__CODEGEARC_VERSION__)
00117 # define COMPILER_ID "Embarcadero"
00118 # define COMPILER_VERSION_MAJOR HEX(__CODEGEARC_VERSION__»24 & 0x00FF)
00119 # define COMPILER_VERSION_MINOR HEX(__CODEGEARC_VERSION__»16 & 0x00FF)
00120 # define COMPILER_VERSION_PATCH DEC(__CODEGEARC_VERSION__     & 0xFFFF)
00121
00122 #elif defined(__BORLANDC__)
00123 # define COMPILER_ID "Borland"
00124   /* __BORLANDC__ = 0xVRR */
00125 # define COMPILER_VERSION_MAJOR HEX(__BORLANDC__»8)
00126 # define COMPILER_VERSION_MINOR HEX(__BORLANDC__ & 0xFF)
00127
00128 #elif defined(__WATCOMC__) && __WATCOMC__ < 1200
00129 # define COMPILER_ID "Watcom"
00130    /* __WATCOMC__ = VVRR */
00131 # define COMPILER_VERSION_MAJOR DEC(__WATCOMC__ / 100)
00132 # define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
00133 # if (__WATCOMC__ % 10) > 0
00134 #  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
00135 # endif
00136
00137 #elif defined(__WATCOMC__)
00138 # define COMPILER_ID "OpenWatcom"
00139    /* __WATCOMC__ = VVRP + 1100 */
00140 # define COMPILER_VERSION_MAJOR DEC((__WATCOMC__ - 1100) / 100)
00141 # define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
00142 # if (__WATCOMC__ % 10) > 0
00143 #  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
00144 # endif
00145
00146 #elif defined(__SUNPRO_C)
00147 # define COMPILER_ID "SunPro"
00148 # if __SUNPRO_C >= 0x5100
00149    /* __SUNPRO_C = 0xVRRP */
00150 #  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_C»12)
00151 #  define COMPILER_VERSION_MINOR HEX(__SUNPRO_C»4 & 0xFF)
00152 #  define COMPILER_VERSION_PATCH HEX(__SUNPRO_C    & 0xF)
00153 # else
00154    /* __SUNPRO_CC = 0xVRP */
00155 #  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_C»8)
00156 #  define COMPILER_VERSION_MINOR HEX(__SUNPRO_C»4 & 0xF)
00157 #  define COMPILER_VERSION_PATCH HEX(__SUNPRO_C    & 0xF)
00158 # endif
00159
00160 #elif defined(__HP_cc)
00161 # define COMPILER_ID "HP"
00162   /* __HP_cc = VVRRPP */
00163 # define COMPILER_VERSION_MAJOR DEC(__HP_cc/10000)
00164 # define COMPILER_VERSION_MINOR DEC(__HP_cc/100 % 100)
00165 # define COMPILER_VERSION_PATCH DEC(__HP_cc     % 100)
00166
00167 #elif defined(__DECC)
00168 # define COMPILER_ID "Compaq"
00169   /* __DECC_VER = VVRRTPPPP */
00170 # define COMPILER_VERSION_MAJOR DEC(__DECC_VER/10000000)
00171 # define COMPILER_VERSION_MINOR DEC(__DECC_VER/100000  % 100)
00172 # define COMPILER_VERSION_PATCH DEC(__DECC_VER         % 10000)
00173
00174 #elif defined(__IBMC__) && defined(__COMPILER_VER__)
00175 # define COMPILER_ID "zOS"
00176   /* __IBMC__ = VRP */
00177 # define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
00178 # define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
00179 # define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)
00180
00181 #elif defined(__open_xl__) && defined(__clang__)
00182 # define COMPILER_ID "IBMClang"
00183 # define COMPILER_VERSION_MAJOR DEC(__open_xl_version__)
00184 # define COMPILER_VERSION_MINOR DEC(__open_xl_release__)
00185 # define COMPILER_VERSION_PATCH DEC(__open_xl_modification__)
00186 # define COMPILER_VERSION_TWEAK DEC(__open_xl_ptf_fix_level__)
00187
00188
00189 #elif defined(__ibmxl__) && defined(__clang__)
00190 # define COMPILER_ID "XLClang"
00191 # define COMPILER_VERSION_MAJOR DEC(__ibmxl_version__)
00192 # define COMPILER_VERSION_MINOR DEC(__ibmxl_release__)
```

```
00193 # define COMPILER_VERSION_PATCH DEC(__ibmxl_modification__)
00194 # define COMPILER_VERSION_TWEAK DEC(__ibmxl_ptf_fix_level__)
00195
00196
00197 #elif defined(__IBMC__) && !defined(__COMPILER_VER__) && __IBMC__ >= 800
00198 # define COMPILER_ID "XL"
00199   /* __IBMC__ = VRP */
00200 # define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
00201 # define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
00202 # define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)
00203
00204 #elif defined(__IBMC__) && !defined(__COMPILER_VER__) && __IBMC__ < 800
00205 # define COMPILER_ID "VisualAge"
00206   /* __IBMC__ = VRP */
00207 # define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
00208 # define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
00209 # define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)
00210
00211 #elif defined(__NVCOMPILER)
00212 # define COMPILER_ID "NVHPC"
00213 # define COMPILER_VERSION_MAJOR DEC(__NVCOMPILER_MAJOR__)
00214 # define COMPILER_VERSION_MINOR DEC(__NVCOMPILER_MINOR__)
00215 # if defined(__NVCOMPILER_PATCHLEVEL__)
00216 #  define COMPILER_VERSION_PATCH DEC(__NVCOMPILER_PATCHLEVEL__)
00217 # endif
00218
00219 #elif defined(__PGI)
00220 # define COMPILER_ID "PGI"
00221 # define COMPILER_VERSION_MAJOR DEC(__PGIC__)
00222 # define COMPILER_VERSION_MINOR DEC(__PGIC_MINOR__)
00223 # if defined(__PGIC_PATCHLEVEL__)
00224 #  define COMPILER_VERSION_PATCH DEC(__PGIC_PATCHLEVEL__)
00225 # endif
00226
00227 #elif defined(_CRAYC)
00228 # define COMPILER_ID "Cray"
00229 # define COMPILER_VERSION_MAJOR DEC(_RELEASE_MAJOR)
00230 # define COMPILER_VERSION_MINOR DEC(_RELEASE_MINOR)
00231
00232 #elif defined(__TI_COMPILER_VERSION__)
00233 # define COMPILER_ID "TI"
00234   /* __TI_COMPILER_VERSION__ = VVVRRRPPP */
00235 # define COMPILER_VERSION_MAJOR DEC(__TI_COMPILER_VERSION__/1000000)
00236 # define COMPILER_VERSION_MINOR DEC(__TI_COMPILER_VERSION__/1000   % 1000)
00237 # define COMPILER_VERSION_PATCH DEC(__TI_COMPILER_VERSION__        % 1000)
00238
00239 #elif defined(__CLANG_FUJITSU)
00240 # define COMPILER_ID "FujitsuClang"
00241 # define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
00242 # define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
00243 # define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
00244 # define COMPILER_VERSION_INTERNAL_STR __clang_version__
00245
00246
00247 #elif defined(__FUJITSU)
00248 # define COMPILER_ID "Fujitsu"
00249 # if defined(__FCC_version__)
00250 #   define COMPILER_VERSION __FCC_version__
00251 # elif defined(__FCC_major__)
00252 #   define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
00253 #   define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
00254 #   define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
00255 # endif
00256 # if defined(__fcc_version)
00257 #   define COMPILER_VERSION_INTERNAL DEC(__fcc_version)
00258 # elif defined(__FCC_VERSION)
00259 #   define COMPILER_VERSION_INTERNAL DEC(__FCC_VERSION)
00260 # endif
00261
00262
00263 #elif defined(__ghs__)
00264 # define COMPILER_ID "GHS"
00265 /* __GHS_VERSION_NUMBER = VVVVRP */
00266 # ifdef __GHS_VERSION_NUMBER
00267 # define COMPILER_VERSION_MAJOR DEC(__GHS_VERSION_NUMBER / 100)
00268 # define COMPILER_VERSION_MINOR DEC(__GHS_VERSION_NUMBER / 10 % 10)
00269 # define COMPILER_VERSION_PATCH DEC(__GHS_VERSION_NUMBER      % 10)
00270 # endif
00271
00272 #elif defined(__TASKING__)
00273 # define COMPILER_ID "Tasking"
00274   # define COMPILER_VERSION_MAJOR DEC(__VERSION__/1000)
00275   # define COMPILER_VERSION_MINOR DEC(__VERSION__ % 100)
00276 # define COMPILER_VERSION_INTERNAL DEC(__VERSION__)
00277
00278 #elif defined(__TINYC__)
00279 # define COMPILER_ID "TinyCC"
```

```
00280
00281 #elif defined(__BCC__)
00282 # define COMPILER_ID "Bruce"
00283
00284 #elif defined(__SCO_VERSION__)
00285 # define COMPILER_ID "SCO"
00286
00287 #elif defined(__ARMCC_VERSION) && !defined(__clang__)
00288 # define COMPILER_ID "ARMCC"
00289 #if __ARMCC_VERSION >= 1000000
00290   /* __ARMCC_VERSION = VRRPPPP */
00291   # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/1000000)
00292   # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 100)
00293   # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION     % 10000)
00294 #else
00295   /* __ARMCC_VERSION = VRPPPP */
00296   # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/100000)
00297   # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 10)
00298   # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION    % 10000)
00299 #endif
00300
00301
00302 #elif defined(__clang__) && defined(__apple_build_version__)
00303 # define COMPILER_ID "AppleClang"
00304 # if defined(_MSC_VER)
00305 #  define SIMULATE_ID "MSVC"
00306 # endif
00307 # define COMPILER_VERSION_MAJOR DEC(__clang_major__)
00308 # define COMPILER_VERSION_MINOR DEC(__clang_minor__)
00309 # define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
00310 # if defined(_MSC_VER)
00311   /* _MSC_VER = VVRR */
00312 #  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00313 #  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00314 # endif
00315 # define COMPILER_VERSION_TWEAK DEC(__apple_build_version__)
00316
00317 #elif defined(__clang__) && defined(__ARMCOMPILER_VERSION)
00318 # define COMPILER_ID "ARMClang"
00319   # define COMPILER_VERSION_MAJOR DEC(__ARMCOMPILER_VERSION/1000000)
00320   # define COMPILER_VERSION_MINOR DEC(__ARMCOMPILER_VERSION/10000 % 100)
00321   # define COMPILER_VERSION_PATCH DEC(__ARMCOMPILER_VERSION     % 10000)
00322 # define COMPILER_VERSION_INTERNAL DEC(__ARMCOMPILER_VERSION)
00323
00324 #elif defined(__clang__)
00325 # define COMPILER_ID "Clang"
00326 # if defined(_MSC_VER)
00327 #  define SIMULATE_ID "MSVC"
00328 # endif
00329 # define COMPILER_VERSION_MAJOR DEC(__clang_major__)
00330 # define COMPILER_VERSION_MINOR DEC(__clang_minor__)
00331 # define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
00332 # if defined(_MSC_VER)
00333   /* _MSC_VER = VVRR */
00334 #  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00335 #  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00336 # endif
00337
00338 #elif defined(__LCC__) && (defined(__GNUC__) || defined(__GNUG__) || defined(__MCST__))
00339 # define COMPILER_ID "LCC"
00340 # define COMPILER_VERSION_MAJOR DEC(__LCC__ / 100)
00341 # define COMPILER_VERSION_MINOR DEC(__LCC__ % 100)
00342 # if defined(__LCC_MINOR__)
00343 #  define COMPILER_VERSION_PATCH DEC(__LCC_MINOR__)
00344 # endif
00345 # if defined(__GNUC__) && defined(__GNUC_MINOR__)
00346 #  define SIMULATE_ID "GNU"
00347 #  define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
00348 #  define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
00349 #  if defined(__GNUC_PATCHLEVEL__)
00350 #   define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00351 #  endif
00352 # endif
00353
00354 #elif defined(__GNUC__)
00355 # define COMPILER_ID "GNU"
00356 # define COMPILER_VERSION_MAJOR DEC(__GNUC__)
00357 # if defined(__GNUC_MINOR__)
00358 #  define COMPILER_VERSION_MINOR DEC(__GNUC_MINOR__)
00359 # endif
00360 # if defined(__GNUC_PATCHLEVEL__)
00361 #  define COMPILER_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00362 # endif
00363
00364 #elif defined(_MSC_VER)
00365 # define COMPILER_ID "MSVC"
00366   /* _MSC_VER = VVRR */
```

```
00367 # define COMPILER_VERSION_MAJOR DEC(_MSC_VER / 100)
00368 # define COMPILER_VERSION_MINOR DEC(_MSC_VER % 100)
00369 # if defined(_MSC_FULL_VER)
00370 #  if _MSC_VER >= 1400
00371     /* _MSC_FULL_VER = VVRRPPPPP */
00372 #   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 100000)
00373 #  else
00374     /* _MSC_FULL_VER = VVRRPPPP */
00375 #   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 10000)
00376 #  endif
00377 # endif
00378 # if defined(_MSC_BUILD)
00379 #  define COMPILER_VERSION_TWEAK DEC(_MSC_BUILD)
00380 # endif
00381
00382 #elif defined(_ADI_COMPILER)
00383 # define COMPILER_ID "ADSP"
00384 #if defined(__VERSIONNUM__)
00385   /* __VERSIONNUM__ = 0xVVRRPPTT */
00386 #  define COMPILER_VERSION_MAJOR DEC(__VERSIONNUM__ » 24 & 0xFF)
00387 #  define COMPILER_VERSION_MINOR DEC(__VERSIONNUM__ » 16 & 0xFF)
00388 #  define COMPILER_VERSION_PATCH DEC(__VERSIONNUM__ » 8 & 0xFF)
00389 #  define COMPILER_VERSION_TWEAK DEC(__VERSIONNUM__ & 0xFF)
00390 #endif
00391
00392 #elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
00393 # define COMPILER_ID "IAR"
00394 # if defined(__VER__) && defined(__ICCARM__)
00395 #  define COMPILER_VERSION_MAJOR DEC((__VER__) / 1000000)
00396 #  define COMPILER_VERSION_MINOR DEC(((__VER__) / 1000) % 1000)
00397 #  define COMPILER_VERSION_PATCH DEC((__VER__) % 1000)
00398 #  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
00399 # elif defined(__VER__) && (defined(__ICCAVR__) || defined(__ICCRX__) || defined(__ICCRH850__) ||
      defined(__ICCRL78__) || defined(__ICC430__) || defined(__ICCRISCV__) || defined(__ICCV850__) ||
      defined(__ICC8051__) || defined(__ICCSTM8__))
00400 #  define COMPILER_VERSION_MAJOR DEC((__VER__) / 100)
00401 #  define COMPILER_VERSION_MINOR DEC((__VER__) - (((__VER__) / 100)*100))
00402 #  define COMPILER_VERSION_PATCH DEC(__SUBVERSION__)
00403 #  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
00404 # endif
00405
00406 #elif defined(__SDCC_VERSION_MAJOR) || defined(SDCC)
00407 # define COMPILER_ID "SDCC"
00408 # if defined(__SDCC_VERSION_MAJOR)
00409 #  define COMPILER_VERSION_MAJOR DEC(__SDCC_VERSION_MAJOR)
00410 #  define COMPILER_VERSION_MINOR DEC(__SDCC_VERSION_MINOR)
00411 #  define COMPILER_VERSION_PATCH DEC(__SDCC_VERSION_PATCH)
00412 # else
00413   /* SDCC = VRP */
00414 #  define COMPILER_VERSION_MAJOR DEC(SDCC/100)
00415 #  define COMPILER_VERSION_MINOR DEC(SDCC/10 % 10)
00416 #  define COMPILER_VERSION_PATCH DEC(SDCC    % 10)
00417 # endif
00418
00419
00420 /* These compilers are either not known or too old to define an
00421   identification macro.  Try to identify the platform and guess that
00422   it is the native compiler.  */
00423 #elif defined(__hpux) || defined(__hpua)
00424 # define COMPILER_ID "HP"
00425
00426 #else /* unknown compiler */
00427 # define COMPILER_ID ""
00428 #endif
00429
00430 /* Construct the string literal in pieces to prevent the source from
00431    getting matched.  Store it in a pointer rather than an array
00432    because some compilers will just produce instructions to fill the
00433    array rather than assigning a pointer to a static array.  */
00434 char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]";
00435 #ifdef SIMULATE_ID
00436 char const* info_simulate = "INFO" ":" "simulate[" SIMULATE_ID "]";
00437 #endif
00438
00439 #ifdef __QNXNTO__
00440 char const* qnxnto = "INFO" ":" "qnxnto[]";
00441 #endif
00442
00443 #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
00444 char const *info_cray = "INFO" ":" "compiler_wrapper[CrayPrgEnv]";
00445 #endif
00446
00447 #define STRINGIFY_HELPER(X) #X
00448 #define STRINGIFY(X) STRINGIFY_HELPER(X)
00449
00450 /* Identify known platforms by name.  */
00451 #if defined(__linux) || defined(__linux__) || defined(linux)
```

```
00452 # define PLATFORM_ID "Linux"
00453
00454 #elif defined(__MSYS__)
00455 # define PLATFORM_ID "MSYS"
00456
00457 #elif defined(__CYGWIN__)
00458 # define PLATFORM_ID "Cygwin"
00459
00460 #elif defined(__MINGW32__)
00461 # define PLATFORM_ID "MinGW"
00462
00463 #elif defined(__APPLE__)
00464 # define PLATFORM_ID "Darwin"
00465
00466 #elif defined(_WIN32) || defined(__WIN32__) || defined(WIN32)
00467 # define PLATFORM_ID "Windows"
00468
00469 #elif defined(__FreeBSD__) || defined(__FreeBSD)
00470 # define PLATFORM_ID "FreeBSD"
00471
00472 #elif defined(__NetBSD__) || defined(__NetBSD)
00473 # define PLATFORM_ID "NetBSD"
00474
00475 #elif defined(__OpenBSD__) || defined(__OPENBSD)
00476 # define PLATFORM_ID "OpenBSD"
00477
00478 #elif defined(__sun) || defined(sun)
00479 # define PLATFORM_ID "SunOS"
00480
00481 #elif defined(_AIX) || defined(__AIX) || defined(__AIX__) || defined(__aix) || defined(__aix__)
00482 # define PLATFORM_ID "AIX"
00483
00484 #elif defined(__hpux) || defined(__hpux__)
00485 # define PLATFORM_ID "HP-UX"
00486
00487 #elif defined(__HAIKU__)
00488 # define PLATFORM_ID "Haiku"
00489
00490 #elif defined(__BeOS) || defined(__BEOS__) || defined(_BEOS)
00491 # define PLATFORM_ID "BeOS"
00492
00493 #elif defined(__QNX__) || defined(__QNXNTO__)
00494 # define PLATFORM_ID "QNX"
00495
00496 #elif defined(__tru64) || defined(_tru64) || defined(__TRU64__)
00497 # define PLATFORM_ID "Tru64"
00498
00499 #elif defined(__riscos) || defined(__riscos__)
00500 # define PLATFORM_ID "RISCos"
00501
00502 #elif defined(__sinix) || defined(__sinix__) || defined(__SINIX__)
00503 # define PLATFORM_ID "SINIX"
00504
00505 #elif defined(__UNIX_SV__)
00506 # define PLATFORM_ID "UNIX_SV"
00507
00508 #elif defined(__bsdos__)
00509 # define PLATFORM_ID "BSDOS"
00510
00511 #elif defined(_MPRAS) || defined(MPRAS)
00512 # define PLATFORM_ID "MP-RAS"
00513
00514 #elif defined(__osf) || defined(__osf__)
00515 # define PLATFORM_ID "OSF1"
00516
00517 #elif defined(_SCO_SV) || defined(SCO_SV) || defined(sco_sv)
00518 # define PLATFORM_ID "SCO_SV"
00519
00520 #elif defined(__ultrix) || defined(__ultrix__) || defined(_ULTRIX)
00521 # define PLATFORM_ID "ULTRIX"
00522
00523 #elif defined(__XENIX__) || defined(_XENIX) || defined(XENIX)
00524 # define PLATFORM_ID "Xenix"
00525
00526 #elif defined(__WATCOMC__)
00527 # if defined(__LINUX__)
00528 #  define PLATFORM_ID "Linux"
00529
00530 # elif defined(__DOS__)
00531 #  define PLATFORM_ID "DOS"
00532
00533 # elif defined(__OS2__)
00534 #  define PLATFORM_ID "OS2"
00535
00536 # elif defined(__WINDOWS__)
00537 #  define PLATFORM_ID "Windows3x"
00538
```

```
00539 # elif defined(__VXWORKS__)
00540 #  define PLATFORM_ID "VxWorks"
00541
00542 # else /* unknown platform */
00543 #  define PLATFORM_ID
00544 # endif
00545
00546 #elif defined(__INTEGRITY)
00547 # if defined(INT_178B)
00548 #  define PLATFORM_ID "Integrity178"
00549
00550 # else /* regular Integrity */
00551 #  define PLATFORM_ID "Integrity"
00552 # endif
00553
00554 # elif defined(_ADI_COMPILER)
00555 #  define PLATFORM_ID "ADSP"
00556
00557 #else /* unknown platform */
00558 # define PLATFORM_ID
00559
00560 #endif
00561
00562 /* For windows compilers MSVC and Intel we can determine
00563    the architecture of the compiler being used.  This is because
00564    the compilers do not have flags that can change the architecture,
00565    but rather depend on which compiler is being used
00566 */
00567 #if defined(_WIN32) && defined(_MSC_VER)
00568 # if defined(_M_IA64)
00569 #  define ARCHITECTURE_ID "IA64"
00570
00571 # elif defined(_M_ARM64EC)
00572 #  define ARCHITECTURE_ID "ARM64EC"
00573
00574 # elif defined(_M_X64) || defined(_M_AMD64)
00575 #  define ARCHITECTURE_ID "x64"
00576
00577 # elif defined(_M_IX86)
00578 #  define ARCHITECTURE_ID "X86"
00579
00580 # elif defined(_M_ARM64)
00581 #  define ARCHITECTURE_ID "ARM64"
00582
00583 # elif defined(_M_ARM)
00584 #  if _M_ARM == 4
00585 #   define ARCHITECTURE_ID "ARMV4I"
00586 #  elif _M_ARM == 5
00587 #   define ARCHITECTURE_ID "ARMV5I"
00588 #  else
00589 #   define ARCHITECTURE_ID "ARMV" STRINGIFY(_M_ARM)
00590 #  endif
00591
00592 # elif defined(_M_MIPS)
00593 #  define ARCHITECTURE_ID "MIPS"
00594
00595 # elif defined(_M_SH)
00596 #  define ARCHITECTURE_ID "SHx"
00597
00598 # else /* unknown architecture */
00599 #  define ARCHITECTURE_ID ""
00600 # endif
00601
00602 #elif defined(__WATCOMC__)
00603 # if defined(_M_I86)
00604 #  define ARCHITECTURE_ID "I86"
00605
00606 # elif defined(_M_IX86)
00607 #  define ARCHITECTURE_ID "X86"
00608
00609 # else /* unknown architecture */
00610 #  define ARCHITECTURE_ID ""
00611 # endif
00612
00613 #elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
00614 # if defined(__ICCARM__)
00615 #  define ARCHITECTURE_ID "ARM"
00616
00617 # elif defined(__ICCRX__)
00618 #  define ARCHITECTURE_ID "RX"
00619
00620 # elif defined(__ICCRH850__)
00621 #  define ARCHITECTURE_ID "RH850"
00622
00623 # elif defined(__ICCRL78__)
00624 #  define ARCHITECTURE_ID "RL78"
00625
```

```
00626 # elif defined(__ICCRISCV__)
00627 #  define ARCHITECTURE_ID "RISCV"
00628
00629 # elif defined(__ICCAVR__)
00630 #  define ARCHITECTURE_ID "AVR"
00631
00632 # elif defined(__ICC430__)
00633 #  define ARCHITECTURE_ID "MSP430"
00634
00635 # elif defined(__ICCV850__)
00636 #  define ARCHITECTURE_ID "V850"
00637
00638 # elif defined(__ICC8051__)
00639 #  define ARCHITECTURE_ID "8051"
00640
00641 # elif defined(__ICCSTM8__)
00642 #  define ARCHITECTURE_ID "STM8"
00643
00644 # else /* unknown architecture */
00645 #  define ARCHITECTURE_ID ""
00646 # endif
00647
00648 #elif defined(__ghs__)
00649 # if defined(__PPC64__)
00650 #  define ARCHITECTURE_ID "PPC64"
00651
00652 # elif defined(__ppc__)
00653 #  define ARCHITECTURE_ID "PPC"
00654
00655 # elif defined(__ARM__)
00656 #  define ARCHITECTURE_ID "ARM"
00657
00658 # elif defined(__x86_64__)
00659 #  define ARCHITECTURE_ID "x64"
00660
00661 # elif defined(__i386__)
00662 #  define ARCHITECTURE_ID "X86"
00663
00664 # else /* unknown architecture */
00665 #  define ARCHITECTURE_ID ""
00666 # endif
00667
00668 #elif defined(__TI_COMPILER_VERSION__)
00669 # if defined(__TI_ARM__)
00670 #  define ARCHITECTURE_ID "ARM"
00671
00672 # elif defined(__MSP430__)
00673 #  define ARCHITECTURE_ID "MSP430"
00674
00675 # elif defined(__TMS320C28XX__)
00676 #  define ARCHITECTURE_ID "TMS320C28x"
00677
00678 # elif defined(__TMS320C6X__) || defined(_TMS320C6X)
00679 #  define ARCHITECTURE_ID "TMS320C6x"
00680
00681 # else /* unknown architecture */
00682 #  define ARCHITECTURE_ID ""
00683 # endif
00684
00685 # elif defined(__ADSPSHARC__)
00686 #  define ARCHITECTURE_ID "SHARC"
00687
00688 # elif defined(__ADSPBLACKFIN__)
00689 #  define ARCHITECTURE_ID "Blackfin"
00690
00691 #elif defined(__TASKING__)
00692
00693 # if defined(__CTC__) || defined(__CPTC__)
00694 #  define ARCHITECTURE_ID "TriCore"
00695
00696 # elif defined(__CMCS__)
00697 #  define ARCHITECTURE_ID "MCS"
00698
00699 # elif defined(__CARM__)
00700 #  define ARCHITECTURE_ID "ARM"
00701
00702 # elif defined(__CARC__)
00703 #  define ARCHITECTURE_ID "ARC"
00704
00705 # elif defined(__C51__)
00706 #  define ARCHITECTURE_ID "8051"
00707
00708 # elif defined(__CPCP__)
00709 #  define ARCHITECTURE_ID "PCP"
00710
00711 # else
00712 #  define ARCHITECTURE_ID ""
```

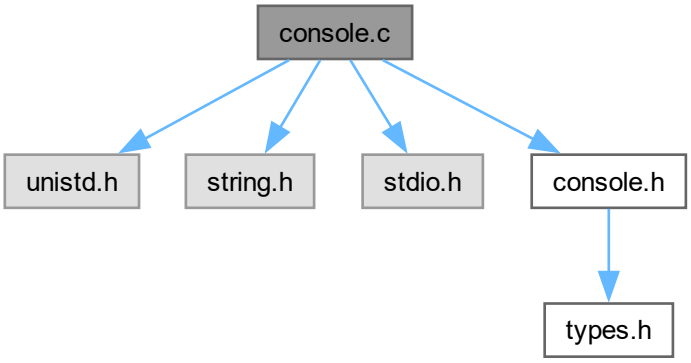
```
00713 # endif
00714
00715 #else
00716 #  define ARCHITECTURE_ID
00717 #endif
00718
00719 /* Convert integer to decimal digit literals.  */
00720 #define DEC(n)                   \
00721   ('0' + (((n) / 10000000)%10)), \
00722   ('0' + (((n) / 1000000)%10)),  \
00723   ('0' + (((n) / 100000)%10)),   \
00724   ('0' + (((n) / 10000)%10)),    \
00725   ('0' + (((n) / 1000)%10)),     \
00726   ('0' + (((n) / 100)%10)),      \
00727   ('0' + (((n) / 10)%10)),       \
00728   ('0' +  ((n) % 10))
00729
00730 /* Convert integer to hex digit literals.  */
00731 #define HEX(n)             \
00732   ('0' + ((n)>>28 & 0xF)), \
00733   ('0' + ((n)>>24 & 0xF)), \
00734   ('0' + ((n)>>20 & 0xF)), \
00735   ('0' + ((n)>>16 & 0xF)), \
00736   ('0' + ((n)>>12 & 0xF)), \
00737   ('0' + ((n)>>8  & 0xF)), \
00738   ('0' + ((n)>>4  & 0xF)), \
00739   ('0' + ((n)     & 0xF))
00740
00741 /* Construct a string literal encoding the version number. */
00742 #ifdef COMPILER_VERSION
00743 char const* info_version = "INFO" ":" "compiler_version[" COMPILER_VERSION "]";
00744
00745 /* Construct a string literal encoding the version number components. */
00746 #elif defined(COMPILER_VERSION_MAJOR)
00747 char const info_version[] = {
00748   'I', 'N', 'F', 'O', ':',
00749   'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','[',
00750   COMPILER_VERSION_MAJOR,
00751 # ifdef COMPILER_VERSION_MINOR
00752   '.', COMPILER_VERSION_MINOR,
00753 #  ifdef COMPILER_VERSION_PATCH
00754    '.', COMPILER_VERSION_PATCH,
00755 #   ifdef COMPILER_VERSION_TWEAK
00756    '.', COMPILER_VERSION_TWEAK,
00757 #   endif
00758 #  endif
00759 # endif
00760   ']','\0'};
00761 #endif
00762
00763 /* Construct a string literal encoding the internal version number. */
00764 #ifdef COMPILER_VERSION_INTERNAL
00765 char const info_version_internal[] = {
00766   'I', 'N', 'F', 'O', ':',
00767   'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','_',
00768   'i','n','t','e','r','n','a','l','[',
00769   COMPILER_VERSION_INTERNAL,']','\0'};
00770 #elif defined(COMPILER_VERSION_INTERNAL_STR)
00771 char const* info_version_internal = "INFO" ":" "compiler_version_internal["
      COMPILER_VERSION_INTERNAL_STR "]";
00772 #endif
00773
00774 /* Construct a string literal encoding the version number components. */
00775 #ifdef SIMULATE_VERSION_MAJOR
00776 char const info_simulate_version[] = {
00777   'I', 'N', 'F', 'O', ':',
00778   's','i','m','u','l','a','t','e','_','v','e','r','s','i','o','n','[',
00779   SIMULATE_VERSION_MAJOR,
00780 # ifdef SIMULATE_VERSION_MINOR
00781   '.', SIMULATE_VERSION_MINOR,
00782 #  ifdef SIMULATE_VERSION_PATCH
00783    '.', SIMULATE_VERSION_PATCH,
00784 #   ifdef SIMULATE_VERSION_TWEAK
00785    '.', SIMULATE_VERSION_TWEAK,
00786 #   endif
00787 #  endif
00788 # endif
00789   ']','\0'};
00790 #endif
00791
00792 /* Construct the string literal in pieces to prevent the source from
00793    getting matched.  Store it in a pointer rather than an array
00794    because some compilers will just produce instructions to fill the
00795    array rather than assigning a pointer to a static array.  */
00796 char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]";
00797 char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]";
00798
```

```
00799
00800
00801 #if !defined(__STDC__) && !defined(__clang__)
00802 # if defined(_MSC_VER) || defined(__ibmxl__) || defined(__IBMC__)
00803 #  define C_VERSION "90"
00804 # else
00805 #  define C_VERSION
00806 # endif
00807 #elif __STDC_VERSION__ > 201710L
00808 # define C_VERSION "23"
00809 #elif __STDC_VERSION__ >= 201710L
00810 # define C_VERSION "17"
00811 #elif __STDC_VERSION__ >= 201000L
00812 # define C_VERSION "11"
00813 #elif __STDC_VERSION__ >= 199901L
00814 # define C_VERSION "99"
00815 #else
00816 # define C_VERSION "90"
00817 #endif
00818 const char* info_language_standard_default =
00819   "INFO" ":" "standard_default[" C_VERSION "]";
00820
00821 const char* info_language_extensions_default = "INFO" ":" "extensions_default["
00822 #if (defined(__clang__) || defined(__GNUC__) || defined(__xlC__) ||           \
00823      defined(__TI_COMPILER_VERSION__)) &&                                     \
00824   !defined(__STRICT_ANSI__)
00825   "ON"
00826 #else
00827   "OFF"
00828 #endif
00829 "]";
00830
00831 /*---------------------------------------------------------------------------*/
00832
00833 #ifdef ID_VOID_MAIN
00834 void main() {}
00835 #else
00836 # if defined(__CLASSIC_C__)
00837 int main(argc, argv) int argc; char *argv[];
00838 # else
00839 int main(int argc, char* argv[])
00840 # endif
00841 {
00842   int require = 0;
00843   require += info_compiler[argc];
00844   require += info_platform[argc];
00845   require += info_arch[argc];
00846 #ifdef COMPILER_VERSION_MAJOR
00847   require += info_version[argc];
00848 #endif
00849 #ifdef COMPILER_VERSION_INTERNAL
00850   require += info_version_internal[argc];
00851 #endif
00852 #ifdef SIMULATE_ID
00853   require += info_simulate[argc];
00854 #endif
00855 #ifdef SIMULATE_VERSION_MAJOR
00856   require += info_simulate_version[argc];
00857 #endif
00858 #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
00859   require += info_cray[argc];
00860 #endif
00861   require += info_language_standard_default[argc];
00862   require += info_language_extensions_default[argc];
00863   (void)argv;
00864   return require;
00865 }
00866 #endif
```

5.5 console.c File Reference

```
#include <unistd.h>
#include <string.h>
#include <stdio.h>
#include "console.h"
```

Include dependency graph for console.c:



Functions

- int init_console (int argc, char **argv, char *theme_file, char *output_file, char *in_file)

  This function is used to initialize from the console.
- file_type_e ends_with (char *file)

  This function is used to get the file type.

5.5.1 Function Documentation

5.5.1.1 ends_with()

```
file_type_e ends_with (
            char * file )
```

This function is used to get the file type.

Parameters

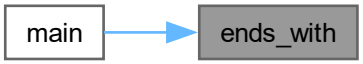
file	file path

Returns

file_type

Definition at line 39 of file console.c.

Here is the caller graph for this function:



5.5.1.2 init_console()

```
int init_console (
            int argc,
            char ** argv,
            char * theme_file,
            char * output_file,
            char * in_file )
```

This function is used to initialize from the console.

Parameters

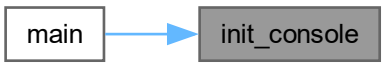
argc	argument_cont from main
argv	arguments list from main
theme_file	theme file path
output_file	output file path

Returns

0 if success, -1 if error

Definition at line 8 of file console.c.

Here is the caller graph for this function:



5.6 console.c

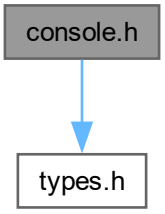
Go to the documentation of this file.

```
00001 //
00002 // Created by sziha on 16/10/2023.
00003 //
00004 #include <unistd.h>
00005 #include <string.h>
00006 #include <stdio.h>
00007 #include "console.h"
00008 int init_console(int argc, char** argv, char* theme_file, char* output_file, char* in_file) {
00009     int c;
00010     while ((c = getopt(argc,argv,":o:t:h")) != -1) {
00011         switch (c) {
00012             case 't':
00013                 strcpy(theme_file, optarg);
00014                 break;
00015             case 'h':
00016                 printf("Usage: console -t <theme_file> -o <output_file> -i <input_file>\n");
00017                 return  -1;
00018             case 'o':
00019                 strcpy(output_file, optarg);
00020                 break;
00021             case ':':
00022                 strcpy(in_file, optarg);
00023                 break;
00024             case '?':
00025                 if (optopt == 'o') {
00026                     fprintf(stderr, "Option -%c requires an argument.\n", optopt);
00027                 } else if (optopt == 't') {
00028                     fprintf(stderr, "Option -%c requires an argument.\n", optopt);
00029                 } else {
00030                     fprintf(stderr, "Unknown option `-%c'.\n", optopt);
00031                 }
00032             default:
00033                 break;
00034         }
00035     }
00036     return 0;
00037 }
00038
00039 file_type_e ends_with(char* file)
00040 {
00041     char* fileExt = strrchr(file, '.');
00042     if (strcmp(fileExt, file) != 0 || fileExt != NULL)
00043     {
00044         if (strcmp(fileExt, ".jpg") == 0)
00045             return file_type_jpg;
00046         else if (strcmp(fileExt, ".png") == 0)
00047             return file_type_png;
00048         else if (strcmp(fileExt, ".c") == 0)
00049             return file_type_c;
00050         else if (strcmp(fileExt, ".h") == 0)
00051             return file_type_h;
00052         /*else if (strcmp(fileExt, "md") == 0)
00053             return file_type_md;*/
00054     }
00055     return DEFAULT_FILE_TYPE;
00056 }
```

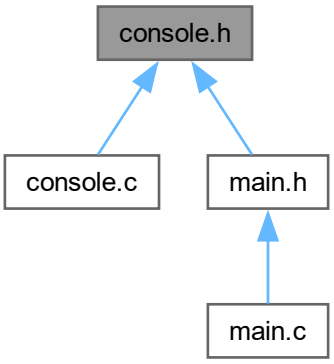
5.7 console.h File Reference

```
#include "types.h"
```

Include dependency graph for console.h:



This graph shows which files directly or indirectly include this file:



Functions

- int init_console (int argc, char **argv, char *theme_file, char *output_file, char *in_file)
  This function is used to initialize from the console.
- file_type_e ends_with (char *file)
  This function is used to get the file type.

5.7.1 Function Documentation

5.7.1.1 ends_with()

```
file_type_e ends_with (
            char * file )
```

This function is used to get the file type.

Parameters

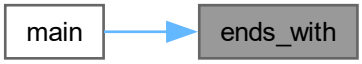
file	file path

Returns

file_type

Definition at line 39 of file console.c.

Here is the caller graph for this function:



5.7.1.2 init_console()

```
int init_console (
            int argc,
            char ** argv,
            char * theme_file,
            char * output_file,
            char * in_file )
```

This function is used to initialize from the console.

Parameters

argc	argument_cont from main
argv	arguments list from main
theme_file	theme file path
output_file	output file path

Returns

0 if success, -1 if error

Definition at line 8 of file console.c.

Here is the caller graph for this function:

5.8 console.h

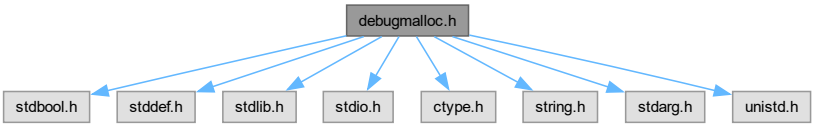
Go to the documentation of this file.

```
00001 //
00002 // Created by sziha on 16/10/2023.
00003 //
00004
00005 #ifndef NHF_CONSOLE_H
00006 #define NHF_CONSOLE_H
00007 #include "types.h"
00016 int init_console(int argc, char** argv, char* theme_file, char* output_file, char* in_file);
00022 file_type_e ends_with(char* file);
00023 #endif //NHF_CONSOLE_H
```

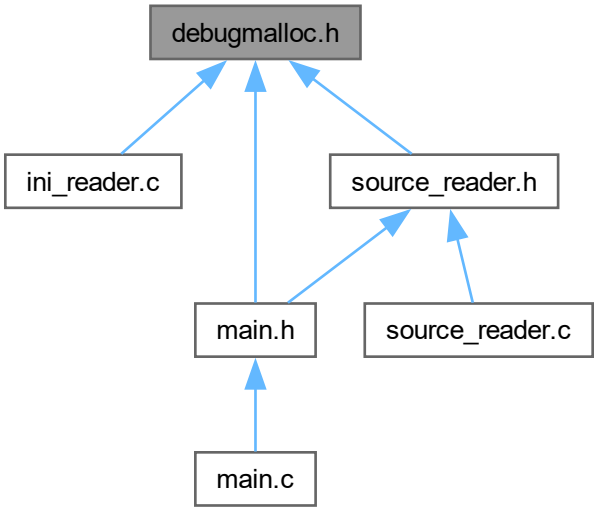
5.9 debugmalloc.h File Reference

```
#include <stdbool.h>
#include <stddef.h>
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdarg.h>
#include <unistd.h>
```

Include dependency graph for debugmalloc.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct DebugmallocEntry
- struct DebugmallocData

Macros

- #define malloc(S) debugmalloc_malloc_full((S), "malloc", #S, __FILE__, __LINE__, false)
- #define calloc(N, S) debugmalloc_malloc_full((N)∗(S), "calloc", #N ", " #S, __FILE__, __LINE__, true)
- #define realloc(P, S) debugmalloc_realloc_full((P), (S), "realloc", #S, __FILE__, __LINE__)
- #define free(P) debugmalloc_free_full((P), "free", __FILE__, __LINE__)

Typedefs

- typedef struct DebugmallocEntry DebugmallocEntry
- typedef struct DebugmallocData DebugmallocData

Enumerations

- enum { debugmalloc_canary_size = 64 , debugmalloc_canary_char = 'K' , debugmalloc_tablesize = 256 , debugmalloc_max_block_size_default = 1048576 }

5.9.1 Macro Definition Documentation

5.9.1.1 calloc

```
#define calloc(
              N,
              S ) debugmalloc_malloc_full((N)*(S), "calloc", #N ", " #S, __FILE__, __LINE__↩
, true)
```

Definition at line 499 of file debugmalloc.h.

5.9.1.2 free

```
#define free(
              P ) debugmalloc_free_full((P), "free", __FILE__, __LINE__)
```

Definition at line 501 of file debugmalloc.h.

5.9.1.3 malloc

```
#define malloc(
              S ) debugmalloc_malloc_full((S), "malloc", #S, __FILE__, __LINE__, false)
```

Definition at line 498 of file debugmalloc.h.

5.9.1.4 realloc

```
#define realloc(
              P,
              S ) debugmalloc_realloc_full((P), (S), "realloc", #S, __FILE__, __LINE__)
```

Definition at line 500 of file debugmalloc.h.

5.9.2 Typedef Documentation

5.9.2.1 DebugmallocData

```
typedef struct DebugmallocData DebugmallocData
```

5.9.2.2 DebugmallocEntry

```
typedef struct DebugmallocEntry DebugmallocEntry
```

5.9.3 Enumeration Type Documentation

5.9.3.1 anonymous enum

```
anonymous enum
```

Enumerator

debugmalloc_canary_size	
debugmalloc_canary_char	
debugmalloc_tablesize	
debugmalloc_max_block_size_default	

Definition at line 13 of file debugmalloc.h.

5.10 debugmalloc.h

Go to the documentation of this file.

```
00001 #ifndef DEBUGMALLOC_H
00002 #define DEBUGMALLOC_H
00003
00004 #include <stdbool.h>
00005 #include <stddef.h>
00006 #include <stdlib.h>
00007 #include <stdio.h>
00008 #include <ctype.h>
00009 #include <string.h>
00010 #include <stdarg.h>
00011
00012
00013 enum {
00014     /* size of canary in bytes. should be multiple of largest alignment
00015      * required by any data type (usually 8 or 16) */
00016     debugmalloc_canary_size = 64,
00017
00018     /* canary byte */
00019     debugmalloc_canary_char = 'K',
00020
00021     /* hash table size for allocated entries */
00022     debugmalloc_tablesize = 256,
00023
00024     /* max block size for allocation, can be modified with debugmalloc_max_block_size() */
00025     debugmalloc_max_block_size_default = 1048576
00026 };
00027
00028
00029 /* make getpid and putenv "crossplatform". deprecated on windows but they work just fine,
00030  * however not declared. */
00031 #ifdef _WIN32
00032     /* windows */
00033     #include <process.h>
00034     #ifdef _MSC_VER
00035         /* visual studio, getenv/getpid deprecated warning */
00036         #pragma warning(disable: 4996)
00037     #else
00038         /* other windows. the declaration is unfortunately hidden
00039          * in mingw header files by ifdefs. */
00040         int putenv(const char *);
00041     #endif
00042 #else
00043     /* posix */
00044     #include <unistd.h>
00045 #endif
00046
00047
00048 /* linked list entry for allocated blocks */
00049 typedef struct DebugmallocEntry {
00050     void *real_mem;     /* the address of the real allocation */
00051     void *user_mem;     /* address shown to the user */
00052     size_t size;        /* size of block requested by user */
00053
00054     char file[64];      /* malloc called in this file */
00055     unsigned line;      /* malloc called at this line in file */
00056     char func[32];      /* allocation function called (malloc, calloc, realloc) */
00057     char expr[128];     /* expression calculating the size of allocation */
00058
00059     struct DebugmallocEntry *prev, *next;  /* for doubly linked list */
00060 } DebugmallocEntry;
00061
00062
00063 /* debugmalloc singleton, storing all state */
```

```
00064 typedef struct DebugmallocData {
00065     char logfile[256];    /* log file name or empty string */
00066     long max_block_size;  /* max size of a single block allocated */
00067     long alloc_count;     /* currently allocated; decreased with free */
00068     long long alloc_bytes;
00069     long all_alloc_count; /* all allocations, never decreased */
00070     long long all_alloc_bytes;
00071     DebugmallocEntry head[debugmalloc_tablesize], tail[debugmalloc_tablesize];  /* head and tail
      elements of allocation lists */
00072 } DebugmallocData;
00073
00074
00075 /* this forward declaration is required by the singleton manager function */
00076 static DebugmallocData * debugmalloc_create(void);
00077
00078
00079 /* creates singleton instance. as this function is static included to different
00080  * translation units, multiple instances of the static variables are created.
00081  * to make sure it is really a singleton, these instances must know each other
00082  * somethow. an environment variable is used for that purpose, ie. the address
00083  * of the singleton allocated is stored by the operating system.
00084  * this implementation is not thread-safe. */
00085 static DebugmallocData * debugmalloc_singleton(void) {
00086     static char envstr[100];
00087     static void *instance = NULL;
00088
00089     /* if we do not know the address of the singleton:
00090      * - maybe we are the one to create it (env variable also does not exist)
00091      * - or it is already created, and stored in the env variable. */
00092     if (instance == NULL) {
00093         char envvarname[100] = "";
00094         sprintf(envvarname, "%s%d", "debugmallocsingleton", (int) getpid());
00095         char *envptr = getenv(envvarname);
00096         if (envptr == NULL) {
00097             /* no env variable: create singleton. */
00098             instance = debugmalloc_create();
00099             sprintf(envstr, "%s=%p", envvarname, instance);
00100             putenv(envstr);
00101         } else {
00102             /* another copy of this function already created it. */
00103             int ok = sscanf(envptr, "%p", &instance);
00104             if (ok != 1) {
00105                 fprintf(stderr, "debugmalloc: nem lehet ertelmezni: %s!\n", envptr);
00106                 abort();
00107             }
00108         }
00109     }
00110
00111     return (DebugmallocData *) instance;
00112 }
00113
00114
00115 /* better version of strncpy, always terminates string with \0. */
00116 static void debugmalloc_strlcpy(char *dest, char const *src, size_t destsize) {
00117     strncpy(dest, src, destsize);
00118     dest[destsize - 1] = '\0';
00119 }
00120
00121
00122 /* set the name of the log file for debugmalloc. empty filename
00123  * means logging to stderr. */
00124 static void debugmalloc_log_file(char const *logfilename) {
00125     if (logfilename == NULL)
00126         logfilename = "";
00127     DebugmallocData *instance = debugmalloc_singleton();
00128     debugmalloc_strlcpy(instance->logfile, logfilename, sizeof(instance->logfile));
00129 }
00130
00131
00132 /* set the maximum size of one block. useful for debugging purposes. */
00133 static void debugmalloc_max_block_size(long max_block_size) {
00134     DebugmallocData *instance = debugmalloc_singleton();
00135     instance->max_block_size = max_block_size;
00136 }
00137
00138
00139
00140 /* printf to the log file, or stderr. */
00141 static void debugmalloc_log(char const *format, ...) {
00142     DebugmallocData *instance = debugmalloc_singleton();
00143     FILE *f = stderr;
00144     if (instance->logfile[0] != '\0') {
00145         f = fopen(instance->logfile, "at");
00146         if (f == NULL) {
00147             f = stderr;
00148             fprintf(stderr, "debugmalloc: nem tudom megnyitni a %s fajlt irasra!\n",
      instance->logfile);
```

```
00149             debugmalloc_strlcpy(instance->logfile, "", sizeof(instance->logfile));
00150         }
00151     }
00152
00153     va_list ap;
00154     va_start(ap, format);
00155     vfprintf(f, format, ap);
00156     va_end(ap);
00157
00158     if (f != stderr)
00159         fclose(f);
00160 }
00161
00162
00163 /* initialize a memory block allocated for the user. the start and the end
00164  * of the block is initialized with the canary characters. if 'zero' is
00165  * true, the user memory area is zero-initialized, otherwise it is also
00166  * filled with the canary character to simulate garbage in memory. */
00167 static void debugmalloc_memory_init(DebugmallocEntry *elem, bool zero) {
00168     unsigned char *real_mem = (unsigned char *) elem->real_mem;
00169     unsigned char *user_mem = (unsigned char *) elem->user_mem;
00170     unsigned char *canary1 = real_mem;
00171     unsigned char *canary2 = real_mem + debugmalloc_canary_size + elem->size;
00172     memset(canary1, debugmalloc_canary_char, debugmalloc_canary_size);
00173     memset(canary2, debugmalloc_canary_char, debugmalloc_canary_size);
00174     memset(user_mem, zero ? 0 : debugmalloc_canary_char, elem->size);
00175 }
00176
00177 /* check canary, return true if ok, false if corrupted. */
00178 static bool debugmalloc_canary_ok(DebugmallocEntry const *elem) {
00179     unsigned char *real_mem = (unsigned char *) elem->real_mem;
00180     unsigned char *canary1 = real_mem;
00181     unsigned char *canary2 = real_mem + debugmalloc_canary_size + elem->size;
00182     for (size_t i = 0; i < debugmalloc_canary_size; ++i) {
00183         if (canary1[i] != debugmalloc_canary_char)
00184             return false;
00185         if (canary2[i] != debugmalloc_canary_char)
00186             return false;
00187     }
00188     return true;
00189 }
00190
00191
00192 /* dump memory contents to log file. */
00193 static void debugmalloc_dump_memory(char const *mem, size_t size) {
00194     for (unsigned y = 0; y < (size + 15) / 16; y++) {
00195         char line[80];
00196         int pos = 0;
00197         pos += sprintf(line + pos, "      %04x  ", y * 16);
00198         for (unsigned x = 0; x < 16; x++) {
00199             if (y * 16 + x < size)
00200                 pos += sprintf(line + pos, "%02x ", mem[y * 16 + x]);
00201             else
00202                 pos += sprintf(line + pos, "   ");
00203         }
00204         pos += sprintf(line + pos, "  ");
00205         for (unsigned x = 0; x < 16; x++) {
00206             if (y * 16 + x < size) {
00207                 unsigned char c = mem[y * 16 + x];
00208                 pos += sprintf(line + pos, "%c", isprint(c) ? c : '.');
00209             }
00210             else {
00211                 pos += sprintf(line + pos, " ");
00212             }
00213         }
00214         debugmalloc_log("%s\n", line);
00215     }
00216 }
00217
00218
00219 /* dump data of allocated memory block.
00220  * if the canary is corrupted, it is also written to the log. */
00221 static void debugmalloc_dump_elem(DebugmallocEntry const *elem) {
00222     bool canary_ok = debugmalloc_canary_ok(elem);
00223
00224     debugmalloc_log("  %p, %u bajt, kanari: %s\n"
00225                     "  %s:%u, %s(%s)\n",
00226                        elem->user_mem, (unsigned) elem->size, canary_ok ? "ok" : "**SERULT**",
00227                        elem->file, elem->line,
00228                        elem->func, elem->expr);
00229
00230     if (!canary_ok) {
00231         debugmalloc_log("    ELOTTE kanari: \n");
00232         debugmalloc_dump_memory((char const *) elem->real_mem, debugmalloc_canary_size);
00233     }
00234
00235     debugmalloc_dump_memory((char const *) elem->user_mem, elem->size > 64 ? 64 : elem->size);
```

```
00236
00237     if (!canary_ok) {
00238         debugmalloc_log("    UTANA kanari: \n");
00239         debugmalloc_dump_memory((char const *) elem->real_mem + debugmalloc_canary_size + elem->size,
    debugmalloc_canary_size);
00240     }
00241 }
00242
00243
00244 /* dump data of all memory blocks allocated. */
00245 static void debugmalloc_dump(void) {
00246     DebugmallocData *instance = debugmalloc_singleton();
00247     debugmalloc_log("** DEBUGMALLOC DUMP ***********************************\n");
00248     int cnt = 0;
00249     for (size_t i = 0; i < debugmalloc_tablesize; i++) {
00250         DebugmallocEntry *head = &instance->head[i];
00251         for (DebugmallocEntry *iter = head->next; iter->next != NULL; iter = iter->next) {
00252             ++cnt;
00253             debugmalloc_log("** %d/%d. rekord:\n", cnt, instance->alloc_count);
00254             debugmalloc_dump_elem(iter);
00255         }
00256     }
00257     debugmalloc_log("** DEBUGMALLOC DUMP VEGE ******************************\n");
00258 }
00259
00260
00261 /* called at program exit to dump data if there is a leak,
00262  * ie. allocated block remained. */
00263 static void debugmalloc_atexit_dump(void) {
00264     DebugmallocData *instance = debugmalloc_singleton();
00265
00266     if (instance->alloc_count > 0) {
00267         debugmalloc_log("\n"
00268                         "*********************************************************\n"
00269                         "* MEMORIASZIVARGAS VAN A PROGRAMBAN!!!\n"
00270                         "*********************************************************\n"
00271                         "\n");
00272         debugmalloc_dump();
00273     } else {
00274         debugmalloc_log("*************************************************\n"
00275                         "* Debugmalloc: nincs memoriaszivargas a programban.\n"
00276                         "* Osszes foglalas: %d blokk, %d bajt.\n"
00277                         "*************************************************\n",
00278                         instance->all_alloc_count, instance->all_alloc_bytes);
00279     }
00280 }
00281
00282
00283 /* hash function for bucket hash. */
00284 static size_t debugmalloc_hash(void *address) {
00285     /* the last few bits are ignored, as they are usually zero for
00286      * alignment purposes. all tested architectures used 16 byte allocation. */
00287     size_t cut = (size_t)address >> 4;
00288     return cut % debugmalloc_tablesize;
00289 }
00290
00291
00292 /* insert element to hash table. */
00293 static void debugmalloc_insert(DebugmallocEntry *entry) {
00294     DebugmallocData *instance = debugmalloc_singleton();
00295     size_t idx = debugmalloc_hash(entry->user_mem);
00296     DebugmallocEntry *head = &instance->head[idx];
00297     entry->prev = head;
00298     entry->next = head->next;
00299     head->next->prev = entry;
00300     head->next = entry;
00301     instance->alloc_count += 1;
00302     instance->alloc_bytes += entry->size;
00303     instance->all_alloc_count += 1;
00304     instance->all_alloc_bytes += entry->size;
00305 }
00306
00307
00308 /* remove element from hash table */
00309 static void debugmalloc_remove(DebugmallocEntry *entry) {
00310     DebugmallocData *instance = debugmalloc_singleton();
00311     entry->next->prev = entry->prev;
00312     entry->prev->next = entry->next;
00313     instance->alloc_count -= 1;
00314     instance->alloc_bytes -= entry->size;
00315 }
00316
00317
00318 /* find element in hash table, given with the memory address that the user sees.
00319  * @return the linked list entry, or null if not found. */
00320 static DebugmallocEntry *debugmalloc_find(void *mem) {
00321     DebugmallocData *instance = debugmalloc_singleton();
```

```
00322     size_t idx = debugmalloc_hash(mem);
00323     DebugmallocEntry *head = &instance->head[idx];
00324     for (DebugmallocEntry *iter = head->next; iter->next != NULL; iter = iter->next)
00325         if (iter->user_mem == mem)
00326             return iter;
00327     return NULL;
00328 }
00329
00330
00331 /* allocate memory. this function is called via the macro. */
00332 static void *debugmalloc_malloc_full(size_t size, char const *func, char const *expr, char const
    *file, unsigned line, bool zero) {
00333     /* imitate standard malloc: return null if size is zero */
00334     if (size == 0)
00335         return NULL;
00336
00337     /* check max size */
00338     DebugmallocData *instance = debugmalloc_singleton();
00339     if (size > (long long unsigned int) instance->max_block_size) {
00340         debugmalloc_log("debugmalloc: %s @ %s:%u: a blokk merete tul nagy, %u bajt;
    debugmalloc_max_block_size() fuggvennyel novelheto.\n", func, file, line, (unsigned) size);
00341         abort();
00342     }
00343
00344     /* allocate more memory, make room for canary */
00345     void *real_mem = malloc(size + 2 * debugmalloc_canary_size);
00346     if (real_mem == NULL) {
00347         debugmalloc_log("debugmalloc: %s @ %s:%u: nem sikerult %u meretu memoriat foglalni!\n", func,
    file, line, (unsigned) size);
00348         return NULL;
00349     }
00350
00351     /* allocate memory for linked list element */
00352     DebugmallocEntry *newentry = (DebugmallocEntry *) malloc(sizeof(DebugmallocEntry));
00353     if (newentry == NULL) {
00354         free(real_mem);
00355         debugmalloc_log("debugmalloc: %s @ %s:%u: le tudtam foglalni %u memoriat, de utana a sajatnak
    nem, sry\n", func, file, line, (unsigned) size);
00356         abort();
00357     }
00358
00359     /* metadata of allocation: caller function, code line etc. */
00360     debugmalloc_strlcpy(newentry->func, func, sizeof(newentry->func));
00361     debugmalloc_strlcpy(newentry->expr, expr, sizeof(newentry->expr));
00362     debugmalloc_strlcpy(newentry->file, file, sizeof(newentry->file));
00363     newentry->line = line;
00364
00365     /* address of allocated memory chunk */
00366     newentry->real_mem = real_mem;
00367     newentry->user_mem = (unsigned char *) real_mem + debugmalloc_canary_size;
00368     newentry->size = size;
00369     debugmalloc_memory_init(newentry, zero);
00370
00371     /* store in list and return pointer to user area */
00372     debugmalloc_insert(newentry);
00373     return newentry->user_mem;
00374 }
00375
00376
00377 /* free memory and remove list item. before deleting, the chuck is filled with
00378  * the canary byte to make sure that the user will see garbage if the memory
00379  * is accessed after freeing. */
00380 static void debugmalloc_free_inner(DebugmallocEntry *deleted) {
00381     debugmalloc_remove(deleted);
00382
00383     /* fill with garbage, then remove from linked list */
00384     memset(deleted->real_mem, debugmalloc_canary_char, deleted->size + 2 * debugmalloc_canary_size);
00385     free(deleted->real_mem);
00386     free(deleted);
00387 }
00388
00389
00390 /* free memory - called via the macro.
00391  * as all allocations are tracked in the list, this function can terminate the program
00392  * if a block is freed twice or the free function is called with an invalid address. */
00393 static void debugmalloc_free_full(void *mem, char const *func, char const *file, unsigned line) {
00394     /* imitate standard free function: if ptr is null, no operation is performed */
00395     if (mem == NULL)
00396         return;
00397
00398     /* find allocation, abort if not found */
00399     DebugmallocEntry *deleted = debugmalloc_find(mem);
00400     if (deleted == NULL) {
00401         debugmalloc_log("debugmalloc: %s @ %s:%u: olyan teruletet probalsz felszabaditani, ami nincs
    lefoglalva!\n", func, file, line);
00402         abort();
00403     }
```

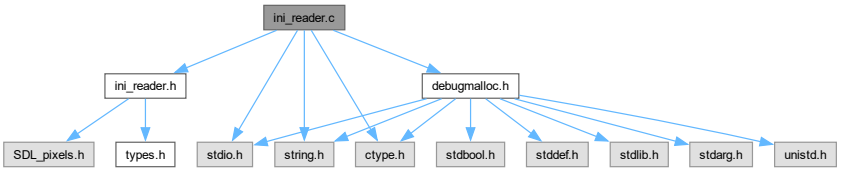
```
00404
00405     /* check canary and then free memory */
00406     if (!debugmalloc_canary_ok(deleted)) {
00407         debugmalloc_log("debugmalloc: %s @ %s:%u: a %p memoriateruletet tulindexelted!\n", func, file,
      line, mem);
00408         debugmalloc_dump_elem(deleted);
00409     }
00410     debugmalloc_free_inner(deleted);
00411 }
00412
00413
00414 /* realloc-like function. */
00415 static void *debugmalloc_realloc_full(void *oldmem, size_t newsize, char const *func, char const
      *expr, char const *file, unsigned line) {
00416     /* imitate standard realloc: equivalent to free if size is null. */
00417     if (newsize == 0) {
00418         debugmalloc_free_full(oldmem, func, file, line);
00419         return NULL;
00420     }
00421     /* imitate standard realloc: equivalent to malloc if first param is NULL */
00422     if (oldmem == NULL)
00423         return debugmalloc_malloc_full(newsize, func, expr, file, line, 0);
00424
00425     /* find old allocation. abort if not found. */
00426     DebugmallocEntry *oldentry = debugmalloc_find(oldmem);
00427     if (oldentry == NULL) {
00428         debugmalloc_log("debugmalloc: %s @ %s:%u: olyan teruletet probalsz atmeretezni, ami nincs
      lefoglalva!\n", func, file, line);
00429         abort();
00430     }
00431
00432     /* create new allocation, copy & free old data */
00433     void *newmem = debugmalloc_malloc_full(newsize, func, expr, file, line, false);
00434     if (newmem == NULL) {
00435         debugmalloc_log("debugmalloc: %s @ %s:%u: nem sikerult uj memoriat foglalni az
      atmeretezeshez!\n", func, file, line);
00436         /* imitate standard realloc: original block is untouched, but return NULL */
00437         return NULL;
00438     }
00439     size_t smaller = oldentry->size < newsize ? oldentry->size : newsize;
00440     memcpy(newmem, oldmem, smaller);
00441     debugmalloc_free_inner(oldentry);
00442
00443     return newmem;
00444 }
00445
00446
00447 /* initialize debugmalloc singleton. returns the newly allocated instance */
00448 static DebugmallocData * debugmalloc_create(void) {
00449     /* config check */
00450     if (debugmalloc_canary_size % 16 != 0) {
00451         debugmalloc_log("debugmalloc: a kanari merete legyen 16-tal oszthato\n");
00452         abort();
00453     }
00454     if (debugmalloc_canary_char == 0) {
00455         debugmalloc_log("debugmalloc: a kanari legyen 0-tol kulonbozo\n");
00456         abort();
00457     }
00458     /* avoid compiler warning if these functions are not used */
00459     (void) debugmalloc_realloc_full;
00460     (void) debugmalloc_log_file;
00461     (void) debugmalloc_max_block_size;
00462
00463     /* create and initialize instance */
00464     DebugmallocData *instance = (DebugmallocData *) malloc(sizeof(DebugmallocData));
00465     if (instance == NULL) {
00466         debugmalloc_log("debugmalloc: nem sikerult elinditani a memoriakezelest\n");
00467         abort();
00468     }
00469     debugmalloc_strlcpy(instance->logfile, "", sizeof(instance->logfile));
00470     instance->max_block_size = debugmalloc_max_block_size_default;
00471     instance->alloc_count = 0;
00472     instance->alloc_bytes = 0;
00473     instance->all_alloc_count = 0;
00474     instance->all_alloc_bytes = 0;
00475     for (size_t i = 0; i < debugmalloc_tablesize; i++) {
00476         instance->head[i].prev = NULL;
00477         instance->head[i].next = &instance->tail[i];
00478         instance->tail[i].next = NULL;
00479         instance->tail[i].prev = &instance->head[i];
00480     }
00481
00482     atexit(debugmalloc_atexit_dump);
00483     return instance;
00484 }
00485
00486
```

```
00487 /* These macro-like functions forward all allocation/free
00488  * calls to debugmalloc. Usage is the same, malloc(size)
00489  * gives the address of a new memory block, free(ptr)
00490  * deallocates etc.
00491  *
00492  * If you use this file, make sure that you include this
00493  * in *ALL* translation units (*.c) of your source. The
00494  * builtin free() function cannot deallocate a memory block
00495  * that was allocated via debugmalloc, yet the name of
00496  * the function is the same! */
00497
00498 #define malloc(S) debugmalloc_malloc_full((S), "malloc", #S, __FILE__, __LINE__, false)
00499 #define calloc(N,S) debugmalloc_malloc_full((N)*(S), "calloc", #N ", " #S, __FILE__, __LINE__, true)
00500 #define realloc(P,S) debugmalloc_realloc_full((P), (S), "realloc", #S, __FILE__, __LINE__)
00501 #define free(P) debugmalloc_free_full((P), "free", __FILE__, __LINE__)
00502
00503 #endif
```

5.11 ini_reader.c File Reference

```
#include "ini_reader.h"
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include "debugmalloc.h"
```

Include dependency graph for ini_reader.c:



Functions

- int read_ini (const char *filename, theme_t *theme)

  reads the theme ini file for custom themes
- void stoLower (char *str)

  turns a string to lowercase
- void set_rgba (char *hex, SDL_Colour *colour)

  Sets the rgba of an SDL_Colour.

5.11.1 Function Documentation

5.11.1.1 read_ini()

```
int read_ini (
            const char * filename,
            theme_t * theme )
```

reads the theme ini file for custom themes

Parameters

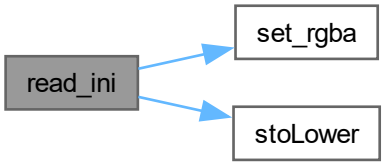
filename	name of the ini file (including the .ini)
theme	pointer to the theme variable

Returns

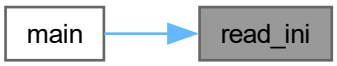
0 if ok, 1 if error

Definition at line 11 of file ini_reader.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.1.2 set_rgba()

```
void set_rgba (
            char * hex,
            SDL_Colour * colour )
```

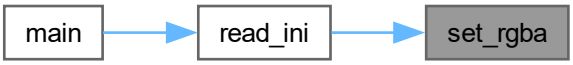
Sets the rgba of an SDL_Colour.

Parameters

hex	hexadecimal string beginning with an #
colour	pointer to the SDL_Colour

Definition at line 88 of file ini_reader.c.

Here is the caller graph for this function:



5.11.1.3 stoLower()

```
void stoLower (
            char * str )
```

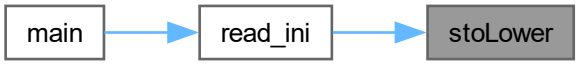
turns a string to lowercase

Parameters

str	string

Definition at line 81 of file ini_reader.c.

Here is the caller graph for this function:



5.12 ini_reader.c

Go to the documentation of this file.

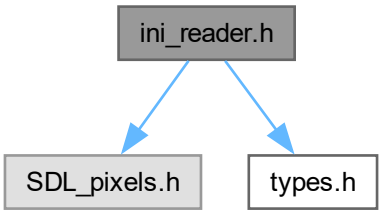
```
00001 //
00002 // Created by sziha on 16/10/2023.
00003 //
00004
00005 #include "ini_reader.h"
00006 #include <stdio.h>
00007 #include <string.h>
00008 #include <ctype.h>
00009 #include "debugmalloc.h"
00010
00011 int read_ini(const char *filename, theme_t *theme) {
00012     FILE *fp = fopen(filename, "r");
00013     if (fp == NULL) {
```

```
00014         fprintf(stderr,"Couldn't open file");
00015         return -1;
00016     }
00017     mapping_t mappings_sc[2];
00018     mapping_t mappings_c[] = {
00019             {"functions", &(theme->functions)},
00020             {"structs", &(theme->structs)},
00021             {"vars", &(theme->variables)},
00022             {"conds", &(theme->conditionals)},
00023             {"loops", &(theme->loops)},
00024             {"main", &(theme->main_)},
00025     };
00026     //printf("file open\n") ;
00027     char line[256]; //if context == -1 go to next line else check subcontext and add to theme
00028     colour_t *context = NULL;
00029     SDL_Colour *subContext = NULL;
00030     while (fgets(line, 256, fp) != NULL) {
00031         //printf("%s", line);
00032         if (line[0] == '[') {
00033             char *name = line + 1;
00034             name = strtok(name, "]");
00035             stoLower(name);
00036             //printf("%s\n", name);
00037             for (int i = 0; i < 6; i++) {
00038                 if (strcmp(name, mappings_c[i].key) == 0) {
00039                     context = (colour_t *) mappings_c[i].value;
00040                     break;
00041                 }
00042             }
00043             if (context == NULL) {
00044                 fprintf(stderr, "Unknown context: %s\n", name);
00045             }
00046             mappings_sc[0].key = "background";
00047             mappings_sc[0].value = &(context->background);
00048             mappings_sc[1].key = "text";
00049             mappings_sc[1].value = &(context->text);
00050             continue;
00051         }
00052         else if (line[0] != ';' && context != NULL)
00053         {
00054             char *value = strtok(line, "=");
00055             unsigned long long int val_len = strlen(value);
00056             //printf("%s", value);
00057             char *valend = value+val_len-1;
00058             while (isspace(*valend)){
00059                 *valend = '\0';
00060                 valend--;
00061             }
00062             for (int i = 0; i < 2; i++) {
00063                 if (strcmp(value, mappings_sc[i].key) == 0) {
00064                     subContext = (SDL_Color *) mappings_sc[i].value;
00065                     break;
00066                 }
00067             }
00068             if (subContext == NULL) {
00069                 fprintf(stderr, "Unknown sub context: %s\n", value);
00070             }
00071             value = strtok(NULL, "=");
00072             while (isspace(*value))
00073                 value++;
00074             set_rgba(value, subContext);
00075         }
00076     }
00077     fclose(fp);
00078     return 0;
00079 }
00080
00081 void stoLower(char *str) {
00082     while (*str != '\0') {
00083         *str = (char) tolower(*str);
00084         str++;
00085     }
00086 }
00087
00088 void set_rgba(char *hex, SDL_Colour *colour) {
00089     char rgba[3] = {hex[1], hex[2],'\0'};
00090     colour->r = strtoul(rgba, NULL, 16);
00091     rgba[0] = hex[3]; rgba[1] = hex[4];
00092     colour->g = strtoul(rgba, NULL, 16);
00093     rgba[0] = hex[5]; rgba[1] = hex[6];
00094     colour->b = strtoul(rgba, NULL, 16);
00095     rgba[0] = hex[7]; rgba[1] = hex[8];
00096     colour->a = strtoul(rgba, NULL, 16);
00097 }
```

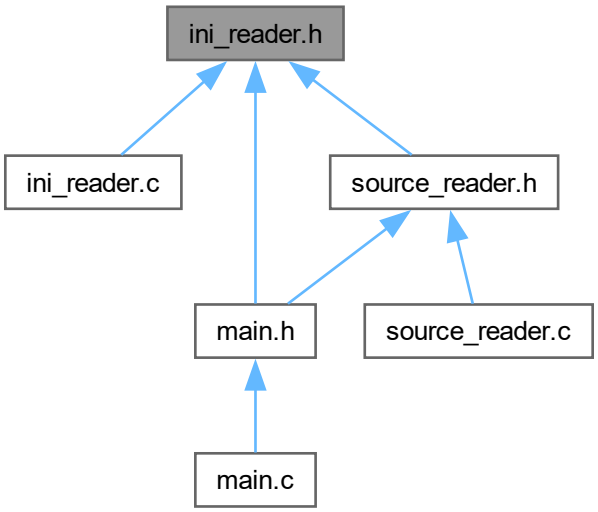
5.13 ini_reader.h File Reference

```
#include <SDL_pixels.h>
#include "types.h"
```

Include dependency graph for ini_reader.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct colour_t

  colouring struct for theme
- struct theme_t

  struct for theme

Enumerations

- enum context_e {
  function , structs , variable , conditional ,
  loop , main_ }

  enum for ini file context
- enum sub_context_e { background , text }

  enum for ini file sub_context (in ini documentation is named value, but i use it like another context so it doesnt matter)

Functions

- int read_ini (const char *filename, theme_t *theme)

  reads the theme ini file for custom themes
- void stoLower (char *str)

  turns a string to lowercase
- void set_rgba (char *hex, SDL_Colour *colour)

  Sets the rgba of an SDL_Colour.

5.13.1 Enumeration Type Documentation

5.13.1.1 context_e

```
enum context_e
```

enum for ini file context

Enumerator

function	
structs	
variable	
conditional	
loop	
main_	

Definition at line 14 of file ini_reader.h.

5.13.1.2 sub_context_e

```
enum sub_context_e
```

enum for ini file sub_context (in ini documentation is named value, but i use it like another context so it doesnt matter)

Enumerator

background	
text	

Definition at line 26 of file ini_reader.h.

5.13.2 Function Documentation

5.13.2.1 read_ini()

```
int read_ini (
            const char * filename,
            theme_t * theme )
```

reads the theme ini file for custom themes

Parameters

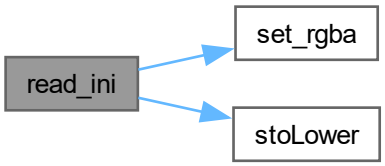
filename	name of the ini file (including the .ini)
theme	pointer to the theme variable

Returns

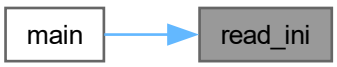
0 if ok, 1 if error

Definition at line 11 of file ini_reader.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.13.2.2 set_rgba()

```
void set_rgba (
            char * hex,
            SDL_Colour * colour )
```

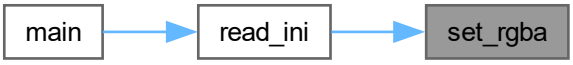
Sets the rgba of an SDL_Colour.

Parameters

hex	hexadecimal string beginning with an #
colour	pointer to the SDL_Colour

Definition at line 88 of file ini_reader.c.

Here is the caller graph for this function:



5.13.2.3 stoLower()

```
void stoLower (
            char * str )
```

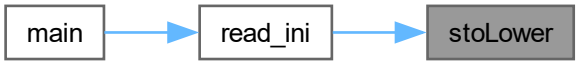
turns a string to lowercase

Parameters

str	string

Definition at line 81 of file ini_reader.c.

Here is the caller graph for this function:



5.14 ini_reader.h

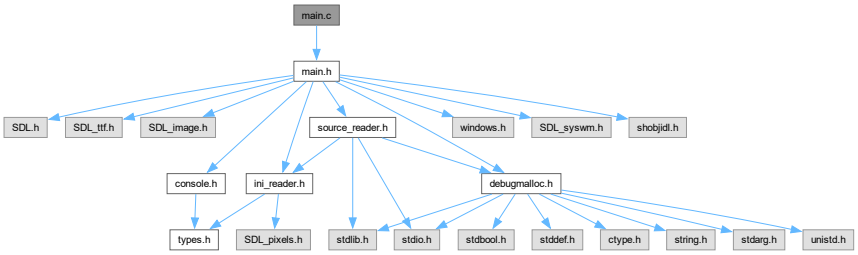
Go to the documentation of this file.

```
00001 //
00002 // Created by sziha on 16/10/2023.
00003 //
00004
00005 #ifndef NHF_INI_READER_H
00006 #define NHF_INI_READER_H
00007
00008 #include <SDL_pixels.h>
00009 #include "types.h"
00010
00014 typedef enum {
00015     function,
00016     structs,
00017     variable,
00018     conditional,
00019     loop,
00020     main_
00021 } context_e;
00026 typedef enum {
00027     background,
00028     text
00029 } sub_context_e;
00030
00034 typedef struct {
00035     SDL_Colour background;
00036     SDL_Colour text;
00037 } colour_t;
00038
00042 typedef struct {
00043     colour_t functions;
00044     colour_t structs;
00045     colour_t variables;
00046     colour_t conditionals;
00047     colour_t loops;
00048     colour_t main_;
00049 } theme_t;
00050
00057 int read_ini(const char *filename, theme_t *theme);
00062 void stoLower(char *str);
00068 void set_rgba(char *hex, SDL_Colour *colour);
00069 #endif //NHF_INI_READER_H
```

5.15 main.c File Reference

`#include "main.h"`

Include dependency graph for main.c:



Functions

- int main (int argc, char **argv)

  Obvious.
- HWND GetHwnd (SDL_Window *window)

Gets win32 window handle.

- void ActivateMenu (HWND windowRef)

  Creates a menu for the given window handle.

- char * file_open_dialog (HWND windowRef)

  Creates a file open dialog for opening source files.

- char * file_save_dialog (HWND windowRef)

  Creates a file save dialog for saving image files.

5.15.1 Function Documentation

5.15.1.1 ActivateMenu()

```
void ActivateMenu (
            HWND windowRef )
```

Creates a menu for the given window handle.

Parameters

windowRef	win32 window handle

Definition at line 110 of file main.c.

Here is the caller graph for this function:

5.15.1.2 file_open_dialog()

```
char * file_open_dialog (
            HWND windowRef )
```

Creates a file open dialog for opening source files.

Parameters

windowRef	

Returns

file to open

Definition at line 132 of file main.c.

Here is the caller graph for this function:

5.15.1.3 file_save_dialog()

```
char * file_save_dialog (
            HWND windowRef )
```

Creates a file save dialog for saving image files.

Parameters

windowRef	

Returns

file to save

Definition at line 179 of file main.c.

Here is the caller graph for this function:

5.15.1.4 GetHwnd()

```
HWND GetHwnd (
            SDL_Window * window )
```

Gets win32 window handle.

Parameters

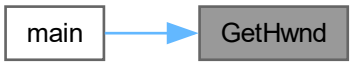
window	sdl window

Returns

win32 window handle

Definition at line 104 of file main.c.

Here is the caller graph for this function:



5.15.1.5 main()

```
int main (
            int argc,
            char ** argv )
```

Obvious.

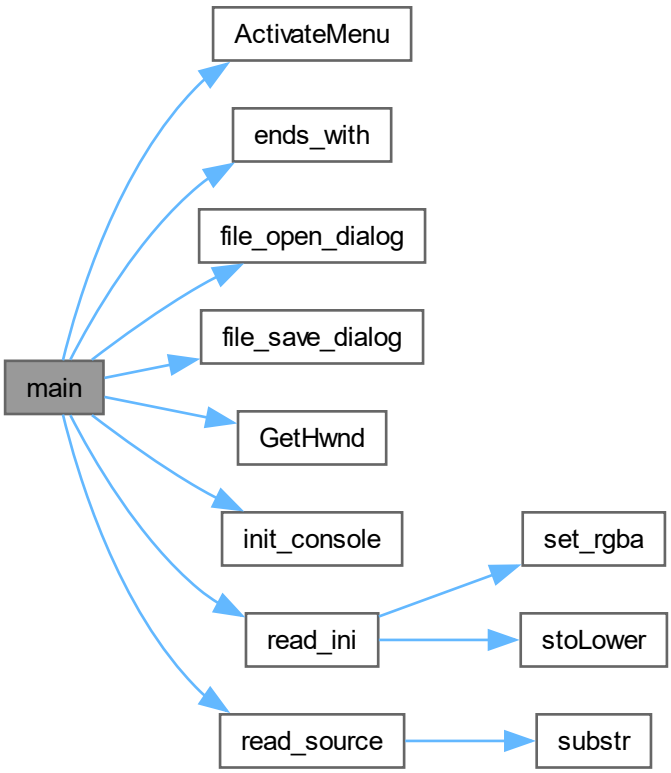
Parameters

argc	
argv	

Returns

Definition at line 3 of file main.c.

Here is the call graph for this function:



5.16 main.c

Go to the documentation of this file.

```
00001 #include "main.h"
00002
00003 int main(int argc, char** argv) {
00004     char *theme_file = "theme.ini";
00005     char *input_file = "test.c";
00006     char *output_file = "test";
00007     theme_t *theme = &default_theme;
00008     printf("%u %u %u", theme->main_.background.r, theme->main_.background.g,
      theme->main_.background.b);
00009     file_type_e output_type;
00010     file_type_e input_type;
00011     node_t *root = NULL;
00012     int quit = 0;
00013     HWND windowRef;
00014     SDL_Event event;
00015
00016     if (argc != 1){
00017         if(init_console(argc, argv, theme_file, output_file, input_file) == -1) return -1;
```

```
00018         output_type = ends_with(output_file);
00019         input_type = ends_with(input_file);
00020     }
00021     if (read_ini(theme_file, theme) == -1) theme = &default_theme;
00022     SDL_Window *window = SDL_CreateWindow("Source to Flow", SDL_WINDOWPOS_UNDEFINED,
      SDL_WINDOWPOS_UNDEFINED, 640, 480, SDL_WINDOW_SHOWN | SDL_WINDOW_RESIZABLE);
00023     windowRef = GetHwnd(window);
00024     ActivateMenu(windowRef);
00025     SDL_Renderer *renderer = SDL_CreateRenderer(window, -1, 0);
00026     SDL_Surface *surface = SDL_CreateRGBSurface(1,640,480,32,0,0,0,0);
00027     SDL_Texture *texture = SDL_CreateTextureFromSurface(renderer, surface);
00028     SDL_FreeSurface(surface);
00029     SDL_EventState(SDL_SYSWMEVENT, SDL_ENABLE);
00030     if (input_type == file_type_c){
00031         root = read_source(input_file);
00032     }
00033     if (output_type == file_type_c){
00034         printf("HOW");
00035     }
00036     while (!quit) {
00037         SDL_PollEvent(&event);
00038         switch (event.type) {
00039             case SDL_WINDOWEVENT_CLOSE:
00040                 event.type = SDL_QUIT;
00041                 SDL_PushEvent(&event);
00042                 break;
00043             case SDL_QUIT:
00044                 quit = 1;
00045                 break;
00046             case SDL_SYSWMEVENT:
00047                 if (event.syswm.msg->msg.win.msg == WM_COMMAND){
00048                     char* temp;
00049                     switch (event.syswm.msg->msg.win.wParam){
00050                         case ID_EXIT:
00051                             quit = 1;
00052                             break;
00053                         case ID_OPEN_FILE:
00054                             ;
00055                             temp = file_open_dialog(windowRef);
00056                             if (temp == NULL) break;
00057                             input_file = temp;
00058                             input_type = ends_with(input_file);
00059                             root = read_source(input_file);
00060                             //printf("%s", input_file);
00061                             break;
00062                         case ID_SAVE_FLOW:
00063                             ;
00064                             temp = file_save_dialog(windowRef);
00065                             if (temp == NULL) break;
00066                             output_file = temp;
00067                             output_type = ends_with(output_file);
00068                             //printf("%s", output_file);
00069                             break;
00070                         case ID_LOAD_THEME:
00071                             break;
00072                         case ID_RESET_THEME:
00073                             theme = &default_theme;
00074                             //redarw call
00075                             break;
00076                         case ID_ZOOM_IN:
00077                             break;
00078                         case ID_ZOOM_OUT:
00079                             break;
00080                         case ID_ZOOM_RESET:
00081                             break;
00082                         default:
00083                             break;
00084                     }
00085                 }
00086                 break;
00087         }
00088 
00089         //SDL_SetRenderDrawColor(renderer, theme->main_.background.r, theme->main_.background.g,
      theme->main_.background.b, theme->main_.background.a);
00090         //SDL_RenderClear(renderer);
00091         SDL_RenderPresent(renderer);
00092     }
00093     SDL_DestroyTexture(texture);
00094     SDL_DestroyRenderer(renderer);
00095     SDL_DestroyWindow(window);
00096     SDL_Quit();
00097     //free(theme);
00098     //free(input_file);
00099     //free(output_file);
00100     free(root);
00101     return 0;
00102 }
```

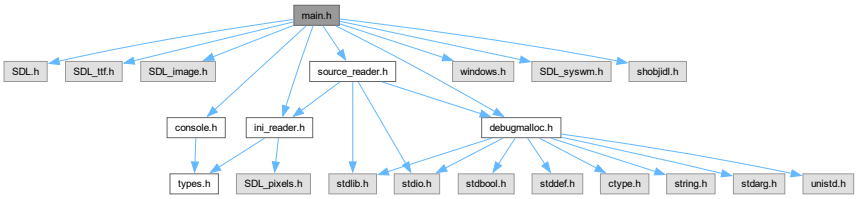
```
00103
00104 HWND GetHwnd(SDL_Window *window){
00105     SDL_SysWMinfo windowInfo;
00106     if(!SDL_GetWindowWMInfo(window,&windowInfo)) return NULL;
00107     return windowInfo.info.win.window;
00108 }
00109
00110 void ActivateMenu(HWND windowRef)
00111 {
00112     HMENU hMenuBar = CreateMenu();
00113     HMENU hFile = CreateMenu();
00114     HMENU hView = CreateMenu();
00115
00116     AppendMenu(hMenuBar, MF_POPUP, (UINT_PTR)hFile, "File");
00117     AppendMenu(hMenuBar, MF_POPUP, (UINT_PTR)hView, "View");
00118     AppendMenu(hMenuBar, MF_STRING, ID_EXIT, "Exit");
00119
00120     AppendMenu(hFile, MF_STRING, ID_OPEN_FILE, "Open File");
00121     AppendMenu(hFile, MF_STRING, ID_SAVE_FLOW, "Save flowchart");
00122
00123     AppendMenu(hView, MF_STRING, ID_LOAD_THEME, "Load Theme");
00124     AppendMenu(hView, MF_STRING, ID_RESET_THEME, "Reset Theme");
00125     AppendMenu(hView, MF_STRING, ID_ZOOM_IN, "Zoom In");
00126     AppendMenu(hView, MF_STRING, ID_ZOOM_OUT, "Zoom Out");
00127     AppendMenu(hView, MF_STRING, ID_ZOOM_RESET, "Zoom Reset");
00128
00129     SetMenu(windowRef, hMenuBar);
00130 }
00131
00132 char* file_open_dialog(HWND windowRef)
00133 {
00134     char *file_path = NULL;
00135     HRESULT hr = CoInitializeEx(NULL, COINIT_APARTMENTTHREADED |
00136                                       COINIT_DISABLE_OLE1DDE);
00137     COMDLG_FILTERSPEC filterspec = {L"Source Files", L"*.c;*.h"};
00138     if (SUCCEEDED(hr))
00139     {
00140         IFileOpenDialog *pFileOpen;
00141
00142         // Create the FileOpenDialog object.
00143         hr = CoCreateInstance(&CLSID_FileOpenDialog, NULL, CLSCTX_ALL,
00144                               &IID_IFileOpenDialog, (void**)(&pFileOpen));
00145         if (SUCCEEDED(hr))
00146         {
00147             pFileOpen->lpVtbl->SetFileTypes(pFileOpen, 1, &filterspec);
00148             // Show the Open dialog box.
00149             hr = pFileOpen->lpVtbl->Show(pFileOpen, windowRef);
00150
00151             // Get the file name from the dialog box.
00152             if (SUCCEEDED(hr))
00153             {
00154                 IShellItem *pItem;
00155                 hr = pFileOpen->lpVtbl->GetResult(pFileOpen, &pItem);
00156                 if (SUCCEEDED(hr))
00157                 {
00158                     PWSTR pszFilePath;
00159                     hr = pItem->lpVtbl->GetDisplayName(pItem,SIGDN_FILESYSPATH,&pszFilePath);
00160
00161                     // Display the file name to the user.
00162                     if (SUCCEEDED(hr))
00163                     {
00164                         //MessageBoxW(NULL, pszFilePath, L"File Path", MB_OK);
00165                         file_path = (char *)malloc(lstrlenW(pszFilePath) + 1);
00166                         wcstombs(file_path, pszFilePath, lstrlenW(pszFilePath) + 1);
00167                         CoTaskMemFree(pszFilePath);
00168                     }
00169                     pItem->lpVtbl->Release((IShellItem *) &pItem);
00170                 }
00171             }
00172             pFileOpen->lpVtbl->Release((IFileOpenDialog *) &pFileOpen);
00173         }
00174         CoUninitialize();
00175     }
00176     return file_path;
00177 }
00178
00179 char* file_save_dialog(HWND windowRef)
00180 {
00181     char *file_path = NULL;
00182     HRESULT hr = CoInitializeEx(NULL, COINIT_APARTMENTTHREADED |
00183                                       COINIT_DISABLE_OLE1DDE);
00184     COMDLG_FILTERSPEC filterspec[2] = {{L"png", L"*.png"}, {L"jpg", L"*.jpg"}}; //TODO: add md later
00185     if (SUCCEEDED(hr))
00186     {
00187         IFileSaveDialog *pFileSave;
00188
00189         // Create the FileOpenDialog object.
```

```
00190         hr = CoCreateInstance(&CLSID_FileSaveDialog, NULL, CLSCTX_ALL,
00191                               &IID_IFileSaveDialog, (void**)(&pFileSave));
00192         if (SUCCEEDED(hr))
00193         {
00194             pFileSave->lpVtbl->SetFileTypes(pFileSave, 2, filterspec);
00195             pFileSave->lpVtbl->SetFileName(pFileSave, L"Flowchart.png");
00196             // Show the Open dialog box.
00197             hr = pFileSave->lpVtbl->Show(pFileSave, windowRef);
00198             // Get the file name from the dialog box.
00199             if (SUCCEEDED(hr))
00200             {
00201                 IShellItem *pItem;
00202                 unsigned int i;
00203                 pFileSave->lpVtbl->GetFileTypeIndex(pFileSave, &i);
00204                 hr = pFileSave->lpVtbl->GetResult(pFileSave, &pItem);
00205                 if (SUCCEEDED(hr))
00206                 {
00207                     PWSTR pszFilePath;
00208                     hr = pItem->lpVtbl->GetDisplayName(pItem,SIGDN_FILESYSPATH,&pszFilePath);
00209 
00210                     // Display the file name to the user.
00211                     if (SUCCEEDED(hr))
00212                     {
00213                         //MessageBoxW(NULL, pszFilePath, L"File Path", MB_OK);
00214                         file_path = (char *)malloc(lstrlenW(pszFilePath) + 5);
00215                         wcstombs(file_path, pszFilePath, lstrlenW(pszFilePath) + 1);
00216                         strcat(file_path, i == 1 ? ".png" : ".jpg");
00217                         CoTaskMemFree(pszFilePath);
00218                     }
00219                     pItem->lpVtbl->Release((IShellItem *) &pItem);
00220                 }
00221             }
00222             pFileSave->lpVtbl->Release((IFileSaveDialog *) &pFileSave);
00223         }
00224         CoUninitialize();
00225     }
00226     return file_path;
00227 }
```

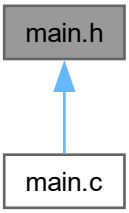
5.17 main.h File Reference

```
#include <SDL.h>
#include <SDL_ttf.h>
#include <SDL_image.h>
#include "console.h"
#include "ini_reader.h"
#include <windows.h>
#include <SDL_syswm.h>
#include <shobjidl.h>
#include "source_reader.h"
#include "debugmalloc.h"
```

Include dependency graph for main.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define ID_OPEN_FILE 1
- #define ID_SAVE_FLOW 2
- #define ID_LOAD_THEME 3
- #define ID_RESET_THEME 4
- #define ID_ZOOM_IN 5
- #define ID_ZOOM_OUT 6
- #define ID_ZOOM_RESET 7
- #define ID_EXIT 8

Functions

- int main (int argc, char **argv)

  Obvious.
- HWND GetHwnd (SDL_Window *window)

  Gets win32 window handle.
- void ActivateMenu (HWND windowRef)

  Creates a menu for the given window handle.
- char * file_open_dialog (HWND windowRef)

  Creates a file open dialog for opening source files.
- char * file_save_dialog (HWND windowRef)

  Creates a file save dialog for saving image files.

5.17.1 Macro Definition Documentation

5.17.1.1 ID_EXIT

```
#define ID_EXIT 8
```

Definition at line 34 of file main.h.

5.17.1.2 ID_LOAD_THEME

```
#define ID_LOAD_THEME 3
```

Definition at line 29 of file main.h.

5.17.1.3 ID_OPEN_FILE

```
#define ID_OPEN_FILE 1
```

Definition at line 27 of file main.h.

5.17.1.4 ID_RESET_THEME

```
#define ID_RESET_THEME 4
```

Definition at line 30 of file main.h.

5.17.1.5 ID_SAVE_FLOW

```
#define ID_SAVE_FLOW 2
```

Definition at line 28 of file main.h.

5.17.1.6 ID_ZOOM_IN

```
#define ID_ZOOM_IN 5
```

Definition at line 31 of file main.h.

5.17.1.7 ID_ZOOM_OUT

```
#define ID_ZOOM_OUT 6
```

Definition at line 32 of file main.h.

5.17.1.8 ID_ZOOM_RESET

```
#define ID_ZOOM_RESET 7
```

Definition at line 33 of file main.h.

5.17.2 Function Documentation

5.17.2.1 ActivateMenu()

```
void ActivateMenu (
            HWND windowRef )
```

Creates a menu for the given window handle.

Parameters

windowRef	win32 window handle

Definition at line 110 of file main.c.

Here is the caller graph for this function:

5.17.2.2 file_open_dialog()

```
char * file_open_dialog (
            HWND windowRef )
```

Creates a file open dialog for opening source files.

Parameters

windowRef	

Returns

file to open

Definition at line 132 of file main.c.

Here is the caller graph for this function:

5.17.2.3 file_save_dialog()

```
char * file_save_dialog (
            HWND windowRef )
```

Creates a file save dialog for saving image files.

Parameters

windowRef	

Returns

file to save

Definition at line 179 of file main.c.

Here is the caller graph for this function:

5.17.2.4 GetHwnd()

```
HWND GetHwnd (
            SDL_Window * window )
```

Gets win32 window handle.

Parameters

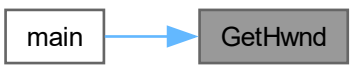
window	sdl window

Returns

win32 window handle

Definition at line 104 of file main.c.

Here is the caller graph for this function:



5.17.2.5 main()

```
int main (
            int argc,
            char ** argv )
```

Obvious.

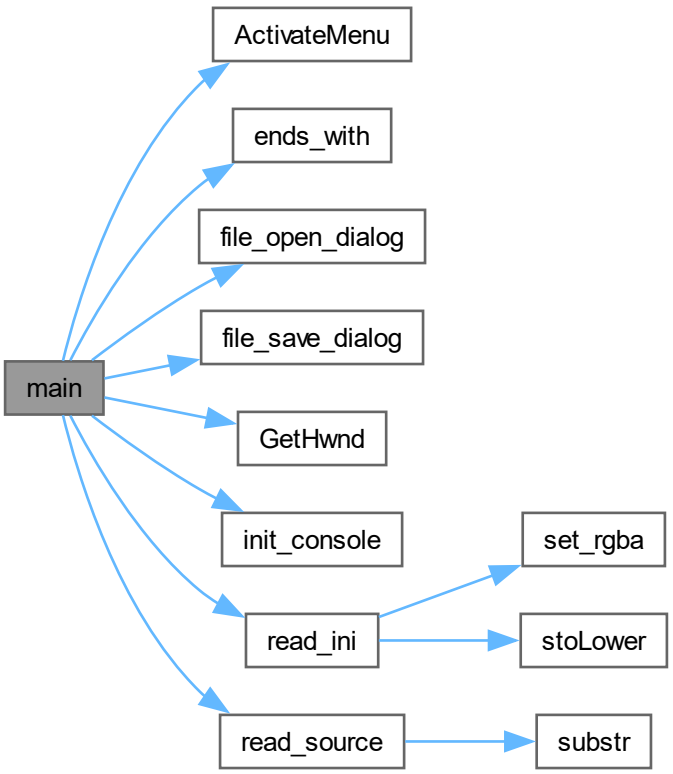
Parameters

argc	
argv	

Returns

Definition at line 3 of file main.c.

Here is the call graph for this function:



5.18 main.h

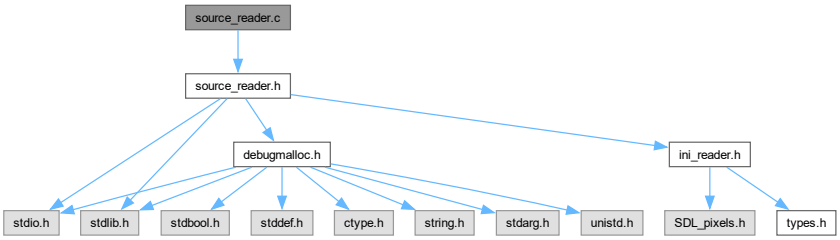
Go to the documentation of this file.

```
00001 //
00002 // Created by sziha on 18/10/2023.
00003 //
00004
00005 #ifndef NHF_MAIN_H
00006 #define NHF_MAIN_H
00007 #endif//NHF_MAIN_H
00008
00009 #include <SDL.h>
00010 #include <SDL_ttf.h>
00011 #include <SDL_image.h>
00012 #include "console.h"
00013 #include "ini_reader.h"
00014 #include <windows.h>
00015 #include <SDL_syswm.h>
00016 #include <shobjidl.h>
00017 #include "source_reader.h"
00018 #include "debugmalloc.h"
00025 int main(int argc, char** argv);
00026
00027 #define ID_OPEN_FILE 1
00028 #define ID_SAVE_FLOW 2
00029 #define ID_LOAD_THEME 3
00030 #define ID_RESET_THEME 4
00031 #define ID_ZOOM_IN 5
00032 #define ID_ZOOM_OUT 6
00033 #define ID_ZOOM_RESET 7
00034 #define ID_EXIT 8
00035
00041 HWND GetHwnd(SDL_Window *window);
00046 void ActivateMenu(HWND windowRef);
00047 //code from
      https://stackoverflow.com/questions/51250046/sdl2-win32-api-menubar-click-event-not-working
00053 char* file_open_dialog(HWND windowRef);
00059 char* file_save_dialog(HWND windowRef);
00060
00061 /*
00062  * Default theme for the viewport
00063  */
00064 static theme_t default_theme = {
00065         .main_ = {
00066             .background = {.r = 255,.g = 255,.b = 255,.a = 255},
00067             .text = {.r = 0,.g = 0,.b = 0,.a = 255}},
00068         .functions = {
00069             .background = {.r = 255,.g = 255,.b = 255,.a = 255},
00070             .text = {.r = 0,.g = 0,.b = 0,.a = 255}},
00071         .structs = {
00072             .background = {.r = 255,.g = 255,.b = 255,.a = 255},
00073             .text = {.r = 0,.g = 0,.b = 0,.a = 255}},
00074         .variables = {
00075             .background = {.r = 255,.g = 255,.b = 255,.a = 255},
00076             .text = {.r = 0,.g = 0,.b = 0,.a = 255}},
00077         .conditionals = {
00078             .background = {.r = 255,.g = 255,.b = 255,.a = 255},
00079             .text = {.r = 0,.g = 0,.b = 0,.a = 255}},
00080         .loops = {
00081             .background = {.r = 255,.g = 255,.b = 255,.a = 255},
00082             .text = {.r = 0,.g = 0,.b = 0,.a = 255}}
00083 };
```

5.19 source_reader.c File Reference

```
#include "source_reader.h"
```

Include dependency graph for source_reader.c:



Functions

- node_t * read_source (char *filename)

  NOT FULLY IMPLEMENTED YET.
- char * substr (char const *str, char start, char end)

  Returns a substirng from the first appearance of start until the first appearance of end.

5.19.1 Function Documentation

5.19.1.1 read_source()

```
node_t * read_source (
            char * filename )
```

NOT FULLY IMPLEMENTED YET.

Parameters

source_file	to read

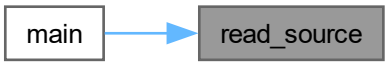
Returns

linked_list of all the lines

Definition at line 7 of file source_reader.c.

Here is the call graph for this function:

Here is the caller graph for this function:



5.19.1.2 substr()

```
char * substr (
            char const * str,
            char start,
            char end )
```

Returns a substirng from the first appearance of start until the first appearance of end.

Parameters

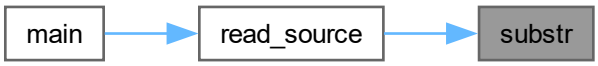
str	
start	
end	

Returns

Substring from start char (inclusive) to end char (non inclusive)

Definition at line 72 of file source_reader.c.

Here is the caller graph for this function:



5.20 source_reader.c

Go to the documentation of this file.

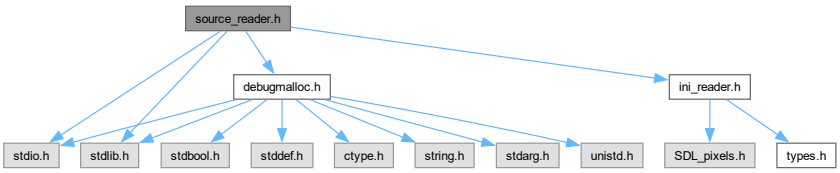
```
00001 //
00002 // Created by sziha on 21/10/2023.
00003 //
00004
00005 #include "source_reader.h"
00006
00007 node_t *read_source(char *filename){
00008     FILE *fp = fopen(filename, "r");
00009     if (fp == NULL) {
00010         fprintf(stderr, "Unable to open file %s\n", filename);
00011         return NULL;
00012     }
00013     node_t *first_node = (node_t *) malloc(sizeof(node_t));
00014     node_t *current_node = first_node;
00015     current_node->type = -1;
00016     char buffer[1000];
00017     int skip = 0;
00018     int c = 0;
00019     int c_bracket[2] = {0,0};
00020     for (int i = 0; c != EOF; i = (i == 999) ? 0 : i + 1) {
00021         c = getc(fp);
00022         if (skip == 1 && c != '\n') {
00023             continue;
00024         }
00025         if (c == '{')
00026         {
00027             c_bracket[1] = c_bracket[0];
00028             c_bracket[0]++;
00029         }
00030         else if (c == '}')
00031         {
00032             c_bracket[1] = c_bracket[0];
00033             c_bracket[0]--;
00034         }
00035         skip = 0;
00036         buffer[i] = (char) c;
00037         buffer[i + 1] = '\0';
00038         if (buffer[i] == '#' || buffer[i] == '\n' || (i > 1 && (buffer[i-1] == '/' && buffer[i] ==
    '/'))) {
00039             i--;
00040             skip = 1;
00041             if (i > 1 && (buffer[i-1] == '/' && buffer[i] == '/')) i--;
00042             continue;
00043         }
00044         fprintf(stderr,"%s", buffer);
00045         switch (current_node->type) {
00046             case structs:
00047                 if (current_node->name[0] == '\0') {
00048                     if (c_bracket[0] == 1)
00049                         i = 0;
00050                     if (c_bracket[0] == 0 && c_bracket[1] == 1) {
00051                         if (c != ';') continue;
00052                         char *name = substr(buffer, '}', ';');
00053                         if (name == NULL) continue;
00054                     }
00055                 }
00056                 break;
00057             default:
00058                 if (strstr(buffer, "struct") != NULL) {
00059                     printf("test");
00060                     current_node->type = structs;
```

```
00061                     if (sscanf(buffer,"struct %s", current_node->name) != 1) { //struct name
00062                         if (sscanf(buffer, "struct %s{", current_node->name) != 1)
    current_node->name[0] = '\0'; //struct name{
00063                     }
00064                 }
00065                 break;
00066         }
00067     }
00068     fclose(fp);
00069     return first_node;
00070 }
00071
00072 char* substr(char const *str, char start, char end) {
00073     char* s = strchr(str, start);
00074     char* e = strchr(str, end);
00075     if (s == NULL || e == NULL)
00076         return NULL;
00077     char* result = malloc((e - s) + 1);
00078     if (result == NULL)
00079         return NULL;
00080     strncpy(result, s, e-s);
00081     result[e-s+1] = '\0';
00082     return result;
00083 }
```

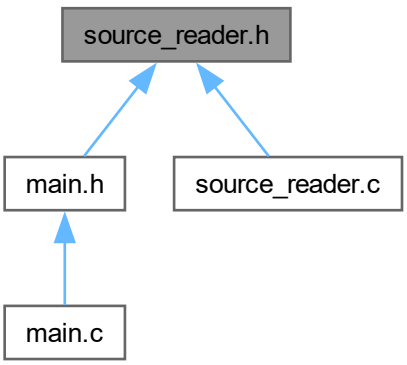
5.21 source_reader.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "ini_reader.h"
#include "debugmalloc.h"
```

Include dependency graph for source_reader.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct func_type_t
- struct struct_type_t
- struct variable_type_t
- struct conditional_type_t
- struct loop_type_t
- struct node

  linked list structure

Typedefs

- typedef struct node node_t

  linked list structure

Functions

- node_t * read_source (char *filename)

  NOT FULLY IMPLEMENTED YET.
- char * substr (char const *str, char start, char end)

  Returns a substirng from the first appearance of start until the first appearance of end.

5.21.1 Typedef Documentation

5.21.1.1 node_t

```
typedef struct node node_t
```

linked list structure

5.21.2 Function Documentation

5.21.2.1 read_source()

```
node_t * read_source (
            char * filename )
```

NOT FULLY IMPLEMENTED YET.

Parameters

source_file	to read

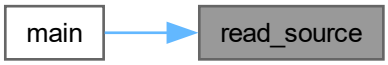
Returns

linked_list of all the lines

Definition at line 7 of file source_reader.c.

Here is the call graph for this function:

Here is the caller graph for this function:



5.21.2.2 substr()

```
char * substr (
            char const * str,
            char start,
            char end )
```

Returns a substirng from the first appearance of start until the first appearance of end.

Parameters

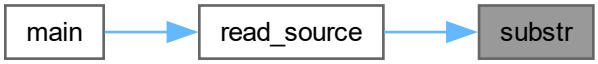
str	
start	
end	

Returns

Substring from start char (inclusive) to end char (non inclusive)

Definition at line 72 of file source_reader.c.

Here is the caller graph for this function:



5.22 source_reader.h

Go to the documentation of this file.

```
00001 //
00002 // Created by sziha on 21/10/2023.
00003 //
00004
00005 #ifndef NHF_SOURCE_READER_H
00006 #define NHF_SOURCE_READER_H
00007 #include <stdio.h>
00008 #include <stdlib.h>
00009 #include "ini_reader.h"
00010 #include "debugmalloc.h"
00011 typedef struct {
00012     char *return_type;
00013     char **args;
00014 } func_type_t;
00015
00016 typedef struct {
00017     char *args;
00018 } struct_type_t;
00019
00020 typedef struct {
00021     char *value;
00022 } variable_type_t;
00023
00024 typedef struct {
00025     char *condition;
00026 } conditional_type_t;
00027
00028 typedef struct {
00029     char *condition;
00030 } loop_type_t;
00034 typedef struct node{
00035     context_e type;
00036     char name[100];
00037     union {
00038         func_type_t func;
00039         struct_type_t struct_;
00040         variable_type_t variable;
00041         conditional_type_t conditional;
00042         loop_type_t loop;
00043     };
00044     int list_size;
00045     struct node **nextList;
00046 } node_t;
00052 node_t *read_source(char *filename);
00060 char* substr(char const *str, char start, char end);
00061 #endif //NHF_SOURCE_READER_H
```

5.23 Specification.md File Reference

5.24 test.c File Reference

Data Structures

- struct test
- struct test_t

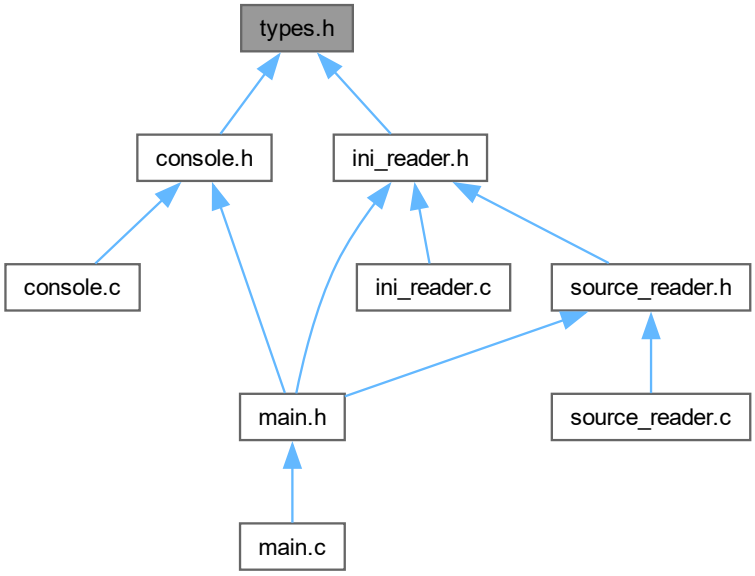
5.25 test.c

Go to the documentation of this file.

```
00001 //
00002 // Created by sziha on 25/10/2023.
00003 //
00004
00005 struct test{
00006     int a;
00007 };
00008 typedef struct{
00009     int b;
00010 } test_t;
```

5.26 types.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct mapping_t

  hash-map like struct for mapping strings to anything (not very safe)

Macros

- #define DEFAULT_FILE_TYPE file_type_png

Enumerations

- enum file_type_e { file_type_h , file_type_c , file_type_jpg , file_type_png }

  Input file enum.

5.26.1 Macro Definition Documentation

5.26.1.1 DEFAULT_FILE_TYPE

```
#define DEFAULT_FILE_TYPE file_type_png
```

Definition at line 26 of file types.h.

5.26.2 Enumeration Type Documentation

5.26.2.1 file_type_e

```
enum file_type_e
```

Input file enum.

Enumerator

file_type_h	
file_type_c	<header file
file_type_jpg	<c file
file_type_png	<jpg file

Definition at line 10 of file types.h.

5.27 types.h

Go to the documentation of this file.

```
00001 //
00002 // Created by sziha on 16/10/2023.
00003 //
00004
00005 #ifndef NHF_TYPES_H
00006 #define NHF_TYPES_H
00010 typedef enum {
00011     file_type_h,
00012     file_type_c,
00013     file_type_jpg,
00014     file_type_png,
00015     //file_type_md, /// <markdown file
00016 } file_type_e;
00017
00021 typedef struct {
00022     const char *key;
00023     const void *value;
00024 } mapping_t;
00025
00026 #define DEFAULT_FILE_TYPE file_type_png
00027
00028 #endif//NHF_TYPES_H
```

Index