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.

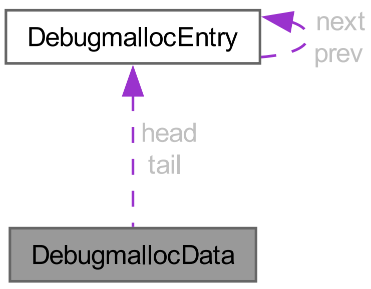
The documentation for this struct was generated from the following file:

- 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:

DebugmallocEntry
next
prev

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:

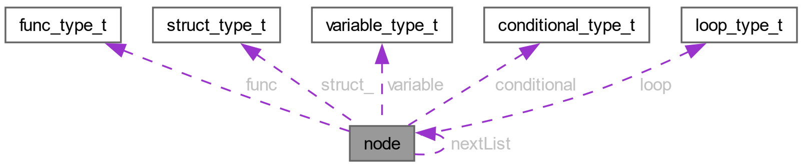
- types.h

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:

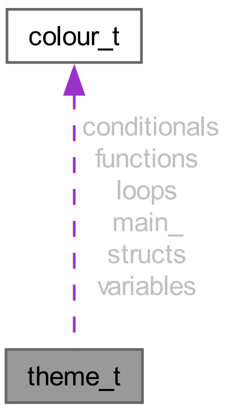
- test.c

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/CMake↩ CCompilerId.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.

```
#ifdef __cplusplus
# error "A C++ compiler has been selected for C."
#endif

#if defined(__18CXX)
# define ID_VOID_MAIN
#endif
#if defined(__CLASSIC_C__)
/* cv-qualifiers did not exist in K&R C */
# define const
# define volatile
#endif

#if !defined(__has_include)
/* If the compiler does not have __has_include, pretend the answer is
   always no.  */
#  define __has_include(x) 0
#endif


/* Version number components: V=Version, R=Revision, P=Patch
   Version date components:   YYYY=Year, MM=Month,   DD=Day  */

#if defined(__INTEL_COMPILER) || defined(__ICC)
# define COMPILER_ID "Intel"
# if defined(_MSC_VER)
#  define SIMULATE_ID "MSVC"
# endif
# if defined(__GNUC__)
#  define SIMULATE_ID "GNU"
# endif
  /* __INTEL_COMPILER = VRP prior to 2021, and then VVVV for 2021 and later,
     except that a few beta releases use the old format with V=2021.  */
# if __INTEL_COMPILER < 2021 || __INTEL_COMPILER == 202110 || __INTEL_COMPILER == 202111
#  define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER/100)
#  define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER/10 % 10)
#  if defined(__INTEL_COMPILER_UPDATE)
#   define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER_UPDATE)
#  else
#   define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER   % 10)
#  endif
# else
#  define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER)
#  define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER_UPDATE)
   /* The third version component from --version is an update index,
      but no macro is provided for it.  */
#  define COMPILER_VERSION_PATCH DEC(0)
# endif
# if defined(__INTEL_COMPILER_BUILD_DATE)
   /* __INTEL_COMPILER_BUILD_DATE = YYYYMMDD */
#  define COMPILER_VERSION_TWEAK DEC(__INTEL_COMPILER_BUILD_DATE)
# endif
# if defined(_MSC_VER)
   /* _MSC_VER = VVRR */
#  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif
# if defined(__GNUC__)
#  define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
# elif defined(__GNUG__)
#  define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
# endif
# if defined(__GNUC_MINOR__)
#  define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
# endif
# if defined(__GNUC_PATCHLEVEL__)
#  define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
# endif

#elif (defined(__clang__) && defined(__INTEL_CLANG_COMPILER)) || defined(__INTEL_LLVM_COMPILER)
# define COMPILER_ID "IntelLLVM"
#if defined(_MSC_VER)
# define SIMULATE_ID "MSVC"
#endif
#if defined(__GNUC__)
# define SIMULATE_ID "GNU"
#endif
/* __INTEL_LLVM_COMPILER = VVVVRP prior to 2021.2.0, VVVVRRPP for 2021.2.0 and
 * later.  Look for 6 digit vs. 8 digit version number to decide encoding.
 * VVVV is no smaller than the current year when a version is released.
 */
#if __INTEL_LLVM_COMPILER < 1000000L
# define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/100)
# define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER    % 10)
#else
# define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/10000)
# define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/100 % 100)
# define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER     % 100)
#endif
#if defined(_MSC_VER)
  /* _MSC_VER = VVRR */
# define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
# define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
#endif
#if defined(__GNUC__)
# define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
#elif defined(__GNUG__)
# define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
#endif
#if defined(__GNUC_MINOR__)
# define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
#endif
#if defined(__GNUC_PATCHLEVEL__)
# define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
#endif

#elif defined(__PATHCC__)
# define COMPILER_ID "PathScale"
# define COMPILER_VERSION_MAJOR DEC(__PATHCC__)
# define COMPILER_VERSION_MINOR DEC(__PATHCC_MINOR__)
# if defined(__PATHCC_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__PATHCC_PATCHLEVEL__)
# endif

#elif defined(__BORLANDC__) && defined(__CODEGEARC_VERSION__)
```

```
# define COMPILER_ID "Embarcadero"
# define COMPILER_VERSION_MAJOR HEX(__CODEGEARC_VERSION__»24 & 0x00FF)
# define COMPILER_VERSION_MINOR HEX(__CODEGEARC_VERSION__»16 & 0x00FF)
# define COMPILER_VERSION_PATCH DEC(__CODEGEARC_VERSION__     & 0xFFFF)

#elif defined(__BORLANDC__)
# define COMPILER_ID "Borland"
  /* __BORLANDC__ = 0xVRR */
# define COMPILER_VERSION_MAJOR HEX(__BORLANDC__»8)
# define COMPILER_VERSION_MINOR HEX(__BORLANDC__ & 0xFF)

#elif defined(__WATCOMC__) && __WATCOMC__ < 1200
# define COMPILER_ID "Watcom"
   /* __WATCOMC__ = VVRR */
# define COMPILER_VERSION_MAJOR DEC(__WATCOMC__ / 100)
# define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
# if (__WATCOMC__ % 10) > 0
#  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
# endif

#elif defined(__WATCOMC__)
# define COMPILER_ID "OpenWatcom"
   /* __WATCOMC__ = VVRP + 1100 */
# define COMPILER_VERSION_MAJOR DEC((__WATCOMC__ - 1100) / 100)
# define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
# if (__WATCOMC__ % 10) > 0
#  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
# endif

#elif defined(__SUNPRO_C)
# define COMPILER_ID "SunPro"
# if __SUNPRO_C >= 0x5100
   /* __SUNPRO_C = 0xVRRP */
#  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_C»12)
#  define COMPILER_VERSION_MINOR HEX(__SUNPRO_C»4 & 0xFF)
#  define COMPILER_VERSION_PATCH HEX(__SUNPRO_C    & 0xF)
# else
   /* __SUNPRO_CC = 0xVRP */
#  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_C»8)
#  define COMPILER_VERSION_MINOR HEX(__SUNPRO_C»4 & 0xF)
#  define COMPILER_VERSION_PATCH HEX(__SUNPRO_C    & 0xF)
# endif

#elif defined(__HP_cc)
# define COMPILER_ID "HP"
  /* __HP_cc = VVRRPP */
# define COMPILER_VERSION_MAJOR DEC(__HP_cc/10000)
# define COMPILER_VERSION_MINOR DEC(__HP_cc/100 % 100)
# define COMPILER_VERSION_PATCH DEC(__HP_cc     % 100)

#elif defined(__DECC)
# define COMPILER_ID "Compaq"
  /* __DECC_VER = VVRRTPPPP */
# define COMPILER_VERSION_MAJOR DEC(__DECC_VER/10000000)
# define COMPILER_VERSION_MINOR DEC(__DECC_VER/100000  % 100)
# define COMPILER_VERSION_PATCH DEC(__DECC_VER         % 10000)

#elif defined(__IBMC__) && defined(__COMPILER_VER__)
# define COMPILER_ID "zOS"
  /* __IBMC__ = VRP */
# define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
# define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)

#elif defined(__open_xl__) && defined(__clang__)
# define COMPILER_ID "IBMClang"
# define COMPILER_VERSION_MAJOR DEC(__open_xl_version__)
# define COMPILER_VERSION_MINOR DEC(__open_xl_release__)
# define COMPILER_VERSION_PATCH DEC(__open_xl_modification__)
# define COMPILER_VERSION_TWEAK DEC(__open_xl_ptf_fix_level__)


#elif defined(__ibmxl__) && defined(__clang__)
# define COMPILER_ID "XLClang"
# define COMPILER_VERSION_MAJOR DEC(__ibmxl_version__)
# define COMPILER_VERSION_MINOR DEC(__ibmxl_release__)
# define COMPILER_VERSION_PATCH DEC(__ibmxl_modification__)
# define COMPILER_VERSION_TWEAK DEC(__ibmxl_ptf_fix_level__)


#elif defined(__IBMC__) && !defined(__COMPILER_VER__) && __IBMC__ >= 800
# define COMPILER_ID "XL"
  /* __IBMC__ = VRP */
# define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
# define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)

```

```
#elif defined(__IBMC__) && !defined(__COMPILER_VER__) && __IBMC__ < 800
# define COMPILER_ID "VisualAge"
  /* __IBMC__ = VRP */
# define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
# define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)

#elif defined(__NVCOMPILER)
# define COMPILER_ID "NVHPC"
# define COMPILER_VERSION_MAJOR DEC(__NVCOMPILER_MAJOR__)
# define COMPILER_VERSION_MINOR DEC(__NVCOMPILER_MINOR__)
# if defined(__NVCOMPILER_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__NVCOMPILER_PATCHLEVEL__)
# endif

#elif defined(__PGI)
# define COMPILER_ID "PGI"
# define COMPILER_VERSION_MAJOR DEC(__PGIC__)
# define COMPILER_VERSION_MINOR DEC(__PGIC_MINOR__)
# if defined(__PGIC_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__PGIC_PATCHLEVEL__)
# endif

#elif defined(_CRAYC)
# define COMPILER_ID "Cray"
# define COMPILER_VERSION_MAJOR DEC(_RELEASE_MAJOR)
# define COMPILER_VERSION_MINOR DEC(_RELEASE_MINOR)

#elif defined(__TI_COMPILER_VERSION__)
# define COMPILER_ID "TI"
  /* __TI_COMPILER_VERSION__ = VVVRRRPPP */
# define COMPILER_VERSION_MAJOR DEC(__TI_COMPILER_VERSION__/1000000)
# define COMPILER_VERSION_MINOR DEC(__TI_COMPILER_VERSION__/1000   % 1000)
# define COMPILER_VERSION_PATCH DEC(__TI_COMPILER_VERSION__        % 1000)

#elif defined(__CLANG_FUJITSU)
# define COMPILER_ID "FujitsuClang"
# define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
# define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
# define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
# define COMPILER_VERSION_INTERNAL_STR __clang_version__


#elif defined(__FUJITSU)
# define COMPILER_ID "Fujitsu"
# if defined(__FCC_version__)
#   define COMPILER_VERSION __FCC_version__
# elif defined(__FCC_major__)
#   define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
#   define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
#   define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
# endif
# if defined(__fcc_version)
#   define COMPILER_VERSION_INTERNAL DEC(__fcc_version)
# elif defined(__FCC_VERSION)
#   define COMPILER_VERSION_INTERNAL DEC(__FCC_VERSION)
# endif


#elif defined(__ghs__)
# define COMPILER_ID "GHS"
/* __GHS_VERSION_NUMBER = VVVVRP */
# ifdef __GHS_VERSION_NUMBER
# define COMPILER_VERSION_MAJOR DEC(__GHS_VERSION_NUMBER / 100)
# define COMPILER_VERSION_MINOR DEC(__GHS_VERSION_NUMBER / 10 % 10)
# define COMPILER_VERSION_PATCH DEC(__GHS_VERSION_NUMBER      % 10)
# endif

#elif defined(__TASKING__)
# define COMPILER_ID "Tasking"
  # define COMPILER_VERSION_MAJOR DEC(__VERSION__/1000)
  # define COMPILER_VERSION_MINOR DEC(__VERSION__ % 100)
# define COMPILER_VERSION_INTERNAL DEC(__VERSION__)

#elif defined(__TINYC__)
# define COMPILER_ID "TinyCC"

#elif defined(__BCC__)
# define COMPILER_ID "Bruce"

#elif defined(__SCO_VERSION__)
# define COMPILER_ID "SCO"

#elif defined(__ARMCC_VERSION) && !defined(__clang__)
# define COMPILER_ID "ARMCC"
#if __ARMCC_VERSION >= 1000000
  /* __ARMCC_VERSION = VRRPPPP */
  # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/1000000)
  # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 100)
  # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION     % 10000)
#else
  /* __ARMCC_VERSION = VRPPPP */
  # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/100000)
  # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 10)
  # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION    % 10000)
#endif


#elif defined(__clang__) && defined(__apple_build_version__)
# define COMPILER_ID "AppleClang"
# if defined(_MSC_VER)
#  define SIMULATE_ID "MSVC"
# endif
# define COMPILER_VERSION_MAJOR DEC(__clang_major__)
# define COMPILER_VERSION_MINOR DEC(__clang_minor__)
# define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
# if defined(_MSC_VER)
   /* _MSC_VER = VVRR */
#  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif
# define COMPILER_VERSION_TWEAK DEC(__apple_build_version__)

#elif defined(__clang__) && defined(__ARMCOMPILER_VERSION)
# define COMPILER_ID "ARMClang"
  # define COMPILER_VERSION_MAJOR DEC(__ARMCOMPILER_VERSION/1000000)
  # define COMPILER_VERSION_MINOR DEC(__ARMCOMPILER_VERSION/10000 % 100)
  # define COMPILER_VERSION_PATCH DEC(__ARMCOMPILER_VERSION     % 10000)
# define COMPILER_VERSION_INTERNAL DEC(__ARMCOMPILER_VERSION)

#elif defined(__clang__)
# define COMPILER_ID "Clang"
# if defined(_MSC_VER)
#  define SIMULATE_ID "MSVC"
# endif
# define COMPILER_VERSION_MAJOR DEC(__clang_major__)
# define COMPILER_VERSION_MINOR DEC(__clang_minor__)
# define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
# if defined(_MSC_VER)
   /* _MSC_VER = VVRR */
#  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif

#elif defined(__LCC__) && (defined(__GNUC__) || defined(__GNUG__) || defined(__MCST__))
# define COMPILER_ID "LCC"
# define COMPILER_VERSION_MAJOR DEC(__LCC__ / 100)
# define COMPILER_VERSION_MINOR DEC(__LCC__ % 100)
# if defined(__LCC_MINOR__)
#  define COMPILER_VERSION_PATCH DEC(__LCC_MINOR__)
# endif
# if defined(__GNUC__) && defined(__GNUC_MINOR__)
#  define SIMULATE_ID "GNU"
#  define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
#  define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
#  if defined(__GNUC_PATCHLEVEL__)
#   define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
#  endif
# endif

#elif defined(__GNUC__)
# define COMPILER_ID "GNU"
# define COMPILER_VERSION_MAJOR DEC(__GNUC__)
# if defined(__GNUC_MINOR__)
#  define COMPILER_VERSION_MINOR DEC(__GNUC_MINOR__)
# endif
# if defined(__GNUC_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
# endif

#elif defined(_MSC_VER)
# define COMPILER_ID "MSVC"
  /* _MSC_VER = VVRR */
# define COMPILER_VERSION_MAJOR DEC(_MSC_VER / 100)
# define COMPILER_VERSION_MINOR DEC(_MSC_VER % 100)
# if defined(_MSC_FULL_VER)
#  if _MSC_VER >= 1400
    /* _MSC_FULL_VER = VVRRPPPPP */
#   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 100000)
#  else
    /* _MSC_FULL_VER = VVRRPPPP */
#   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 10000)
#  endif
# endif
# if defined(_MSC_BUILD)
#  define COMPILER_VERSION_TWEAK DEC(_MSC_BUILD)
# endif

#elif defined(_ADI_COMPILER)
# define COMPILER_ID "ADSP"
#if defined(__VERSIONNUM__)
  /* __VERSIONNUM__ = 0xVVRRPPTT */
#  define COMPILER_VERSION_MAJOR DEC(__VERSIONNUM__ » 24 & 0xFF)
#  define COMPILER_VERSION_MINOR DEC(__VERSIONNUM__ » 16 & 0xFF)
#  define COMPILER_VERSION_PATCH DEC(__VERSIONNUM__ » 8 & 0xFF)
#  define COMPILER_VERSION_TWEAK DEC(__VERSIONNUM__ & 0xFF)
#endif

#elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
# define COMPILER_ID "IAR"
# if defined(__VER__) && defined(__ICCARM__)
#  define COMPILER_VERSION_MAJOR DEC((__VER__) / 1000000)
#  define COMPILER_VERSION_MINOR DEC(((__VER__) / 1000) % 1000)
#  define COMPILER_VERSION_PATCH DEC((__VER__) % 1000)
#  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
# elif defined(__VER__) && (defined(__ICCAVR__) || defined(__ICCRX__) || defined(__ICCRH850__) ||
defined(__ICCRL78__) || defined(__ICC430__) || defined(__ICCRISCV__) || defined(__ICCV850__) ||
defined(__ICC8051__) || defined(__ICCSTM8__))
#  define COMPILER_VERSION_MAJOR DEC((__VER__) / 100)
#  define COMPILER_VERSION_MINOR DEC((__VER__) - (((__VER__) / 100)*100))
#  define COMPILER_VERSION_PATCH DEC(__SUBVERSION__)
#  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
# endif

#elif defined(__SDCC_VERSION_MAJOR) || defined(SDCC)
# define COMPILER_ID "SDCC"
# if defined(__SDCC_VERSION_MAJOR)
#  define COMPILER_VERSION_MAJOR DEC(__SDCC_VERSION_MAJOR)
#  define COMPILER_VERSION_MINOR DEC(__SDCC_VERSION_MINOR)
#  define COMPILER_VERSION_PATCH DEC(__SDCC_VERSION_PATCH)
# else
  /* SDCC = VRP */
#  define COMPILER_VERSION_MAJOR DEC(SDCC/100)
#  define COMPILER_VERSION_MINOR DEC(SDCC/10 % 10)
#  define COMPILER_VERSION_PATCH DEC(SDCC    % 10)
# endif


/* These compilers are either not known or too old to define an
  identification macro.  Try to identify the platform and guess that
  it is the native compiler.  */
#elif defined(__hpux) || defined(__hpua)
# define COMPILER_ID "HP"

#else /* unknown compiler */
# define COMPILER_ID ""
#endif

/* Construct the string literal in pieces to prevent the source from
   getting matched.  Store it in a pointer rather than an array
   because some compilers will just produce instructions to fill the
   array rather than assigning a pointer to a static array.  */
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]";
#ifdef SIMULATE_ID
char const* info_simulate = "INFO" ":" "simulate[" SIMULATE_ID "]";
#endif

#ifdef __QNXNTO__
char const* qnxnto = "INFO" ":" "qnxnto[]";
#endif

#if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
char const *info_cray = "INFO" ":" "compiler_wrapper[CrayPrgEnv]";
#endif

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

/* Identify known platforms by name.  */
#if defined(__linux) || defined(__linux__) || defined(linux)
# define PLATFORM_ID "Linux"

#elif defined(__MSYS__)
# define PLATFORM_ID "MSYS"

#elif defined(__CYGWIN__)
# define PLATFORM_ID "Cygwin"

#elif defined(__MINGW32__)
# define PLATFORM_ID "MinGW"

```

```
#elif defined(__APPLE__)
# define PLATFORM_ID "Darwin"

#elif defined(_WIN32) || defined(__WIN32__) || defined(WIN32)
# define PLATFORM_ID "Windows"

#elif defined(__FreeBSD__) || defined(__FreeBSD)
# define PLATFORM_ID "FreeBSD"

#elif defined(__NetBSD__) || defined(__NetBSD)
# define PLATFORM_ID "NetBSD"

#elif defined(__OpenBSD__) || defined(__OPENBSD)
# define PLATFORM_ID "OpenBSD"

#elif defined(__sun) || defined(sun)
# define PLATFORM_ID "SunOS"

#elif defined(_AIX) || defined(__AIX) || defined(__AIX__) || defined(__aix) || defined(__aix__)
# define PLATFORM_ID "AIX"

#elif defined(__hpux) || defined(__hpux__)
# define PLATFORM_ID "HP-UX"

#elif defined(__HAIKU__)
# define PLATFORM_ID "Haiku"

#elif defined(__BeOS) || defined(__BEOS__) || defined(_BEOS)
# define PLATFORM_ID "BeOS"

#elif defined(__QNX__) || defined(__QNXNTO__)
# define PLATFORM_ID "QNX"

#elif defined(__tru64) || defined(_tru64) || defined(__TRU64__)
# define PLATFORM_ID "Tru64"

#elif defined(__riscos) || defined(__riscos__)
# define PLATFORM_ID "RISCos"

#elif defined(__sinix) || defined(__sinix__) || defined(__SINIX__)
# define PLATFORM_ID "SINIX"

#elif defined(__UNIX_SV__)
# define PLATFORM_ID "UNIX_SV"

#elif defined(__bsdos__)
# define PLATFORM_ID "BSDOS"

#elif defined(_MPRAS) || defined(MPRAS)
# define PLATFORM_ID "MP-RAS"

#elif defined(__osf) || defined(__osf__)
# define PLATFORM_ID "OSF1"

#elif defined(_SCO_SV) || defined(SCO_SV) || defined(sco_sv)
# define PLATFORM_ID "SCO_SV"

#elif defined(__ultrix) || defined(__ultrix__) || defined(_ULTRIX)
# define PLATFORM_ID "ULTRIX"

#elif defined(__XENIX__) || defined(_XENIX) || defined(XENIX)
# define PLATFORM_ID "Xenix"

#elif defined(__WATCOMC__)
# if defined(__LINUX__)
#  define PLATFORM_ID "Linux"

# elif defined(__DOS__)
#  define PLATFORM_ID "DOS"

# elif defined(__OS2__)
#  define PLATFORM_ID "OS2"

# elif defined(__WINDOWS__)
#  define PLATFORM_ID "Windows3x"

# elif defined(__VXWORKS__)
#  define PLATFORM_ID "VxWorks"

# else /* unknown platform */
#  define PLATFORM_ID
# endif

#elif defined(__INTEGRITY)
# if defined(INT_178B)
#  define PLATFORM_ID "Integrity178"

# else /* regular Integrity */
#  define PLATFORM_ID "Integrity"
# endif

# elif defined(_ADI_COMPILER)
#  define PLATFORM_ID "ADSP"

#else /* unknown platform */
# define PLATFORM_ID

#endif

/* For windows compilers MSVC and Intel we can determine
   the architecture of the compiler being used.  This is because
   the compilers do not have flags that can change the architecture,
   but rather depend on which compiler is being used
*/
#if defined(_WIN32) && defined(_MSC_VER)
# if defined(_M_IA64)
#  define ARCHITECTURE_ID "IA64"

# elif defined(_M_ARM64EC)
#  define ARCHITECTURE_ID "ARM64EC"

# elif defined(_M_X64) || defined(_M_AMD64)
#  define ARCHITECTURE_ID "x64"

# elif defined(_M_IX86)
#  define ARCHITECTURE_ID "X86"

# elif defined(_M_ARM64)
#  define ARCHITECTURE_ID "ARM64"

# elif defined(_M_ARM)
#  if _M_ARM == 4
#   define ARCHITECTURE_ID "ARMV4I"
#  elif _M_ARM == 5
#   define ARCHITECTURE_ID "ARMV5I"
#  else
#   define ARCHITECTURE_ID "ARMV" STRINGIFY(_M_ARM)
#  endif

# elif defined(_M_MIPS)
#  define ARCHITECTURE_ID "MIPS"

# elif defined(_M_SH)
#  define ARCHITECTURE_ID "SHx"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__WATCOMC__)
# if defined(_M_I86)
#  define ARCHITECTURE_ID "I86"

# elif defined(_M_IX86)
#  define ARCHITECTURE_ID "X86"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
# if defined(__ICCARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__ICCRX__)
#  define ARCHITECTURE_ID "RX"

# elif defined(__ICCRH850__)
#  define ARCHITECTURE_ID "RH850"

# elif defined(__ICCRL78__)
#  define ARCHITECTURE_ID "RL78"

# elif defined(__ICCRISCV__)
#  define ARCHITECTURE_ID "RISCV"

# elif defined(__ICCAVR__)
#  define ARCHITECTURE_ID "AVR"

# elif defined(__ICC430__)
#  define ARCHITECTURE_ID "MSP430"

# elif defined(__ICCV850__)
#  define ARCHITECTURE_ID "V850"

# elif defined(__ICC8051__)
#  define ARCHITECTURE_ID "8051"

# elif defined(__ICCSTM8__)
#  define ARCHITECTURE_ID "STM8"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__ghs__)
# if defined(__PPC64__)
#  define ARCHITECTURE_ID "PPC64"

# elif defined(__ppc__)
#  define ARCHITECTURE_ID "PPC"

# elif defined(__ARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__x86_64__)
#  define ARCHITECTURE_ID "x64"

# elif defined(__i386__)
#  define ARCHITECTURE_ID "X86"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__TI_COMPILER_VERSION__)
# if defined(__TI_ARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__MSP430__)
#  define ARCHITECTURE_ID "MSP430"

# elif defined(__TMS320C28XX__)
#  define ARCHITECTURE_ID "TMS320C28x"

# elif defined(__TMS320C6X__) || defined(_TMS320C6X)
#  define ARCHITECTURE_ID "TMS320C6x"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

# elif defined(__ADSPSHARC__)
#  define ARCHITECTURE_ID "SHARC"

# elif defined(__ADSPBLACKFIN__)
#  define ARCHITECTURE_ID "Blackfin"

#elif defined(__TASKING__)

# if defined(__CTC__) || defined(__CPTC__)
#  define ARCHITECTURE_ID "TriCore"

# elif defined(__CMCS__)
#  define ARCHITECTURE_ID "MCS"

# elif defined(__CARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__CARC__)
#  define ARCHITECTURE_ID "ARC"

# elif defined(__C51__)
#  define ARCHITECTURE_ID "8051"

# elif defined(__CPCP__)
#  define ARCHITECTURE_ID "PCP"

# else
#  define ARCHITECTURE_ID ""
# endif

#else
#  define ARCHITECTURE_ID
#endif

/* Convert integer to decimal digit literals.  */
#define DEC(n)                   \
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
```

```c
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))

/* Convert integer to hex digit literals.  */
#define HEX(n)             \
  ('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))

/* Construct a string literal encoding the version number. */
#ifdef COMPILER_VERSION
char const* info_version = "INFO" ":" "compiler_version[" COMPILER_VERSION "]";

/* Construct a string literal encoding the version number components. */
#elif defined(COMPILER_VERSION_MAJOR)
char const info_version[] = {
  'I', 'N', 'F', 'O', ':',
  'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','[',
  COMPILER_VERSION_MAJOR,
# ifdef COMPILER_VERSION_MINOR
  '.', COMPILER_VERSION_MINOR,
#  ifdef COMPILER_VERSION_PATCH
   '.', COMPILER_VERSION_PATCH,
#   ifdef COMPILER_VERSION_TWEAK
    '.', COMPILER_VERSION_TWEAK,
#   endif
#  endif
# endif
  ']','\0'};
#endif

/* Construct a string literal encoding the internal version number. */
#ifdef COMPILER_VERSION_INTERNAL
char const info_version_internal[] = {
  'I', 'N', 'F', 'O', ':',
  'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','_',
  'i','n','t','e','r','n','a','l','[',
  COMPILER_VERSION_INTERNAL,']','\0'};
#elif defined(COMPILER_VERSION_INTERNAL_STR)
char const* info_version_internal = "INFO" ":" "compiler_version_internal["
COMPILER_VERSION_INTERNAL_STR "]";
#endif

/* Construct a string literal encoding the version number components. */
#ifdef SIMULATE_VERSION_MAJOR
char const info_simulate_version[] = {
  'I', 'N', 'F', 'O', ':',
  's','i','m','u','l','a','t','e','_','v','e','r','s','i','o','n','[',
  SIMULATE_VERSION_MAJOR,
# ifdef SIMULATE_VERSION_MINOR
  '.', SIMULATE_VERSION_MINOR,
#  ifdef SIMULATE_VERSION_PATCH
   '.', SIMULATE_VERSION_PATCH,
#   ifdef SIMULATE_VERSION_TWEAK
    '.', SIMULATE_VERSION_TWEAK,
#   endif
#  endif
# endif
  ']','\0'};
#endif

/* Construct the string literal in pieces to prevent the source from
   getting matched.  Store it in a pointer rather than an array
   because some compilers will just produce instructions to fill the
   array rather than assigning a pointer to a static array.  */
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]";
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]";



#if !defined(__STDC__) && !defined(__clang__)
# if defined(_MSC_VER) || defined(__ibmxl__) || defined(__IBMC__)
#  define C_VERSION "90"
# else
#  define C_VERSION
# endif
#elif __STDC_VERSION__ > 201710L
# define C_VERSION "23"
#elif __STDC_VERSION__ >= 201710L
```

```
# define C_VERSION "17"
#elif __STDC_VERSION__ >= 201000L
# define C_VERSION "11"
#elif __STDC_VERSION__ >= 199901L
# define C_VERSION "99"
#else
# define C_VERSION "90"
#endif
const char* info_language_standard_default =
  "INFO" ":" "standard_default[" C_VERSION "]";

const char* info_language_extensions_default = "INFO" ":" "extensions_default["
#if (defined(__clang__) || defined(__GNUC__) || defined(__xlC__) ||           \
     defined(__TI_COMPILER_VERSION__)) &&                                     \
  !defined(__STRICT_ANSI__)
  "ON"
#else
  "OFF"
#endif
"]";

/*--------------------------------------------------------------------------*/

#ifdef ID_VOID_MAIN
void main() {}
#else
# if defined(__CLASSIC_C__)
int main(argc, argv) int argc; char *argv[];
# else
int main(int argc, char* argv[])
# endif
{
  int require = 0;
  require += info_compiler[argc];
  require += info_platform[argc];
  require += info_arch[argc];
#ifdef COMPILER_VERSION_MAJOR
  require += info_version[argc];
#endif
#ifdef COMPILER_VERSION_INTERNAL
  require += info_version_internal[argc];
#endif
#ifdef SIMULATE_ID
  require += info_simulate[argc];
#endif
#ifdef SIMULATE_VERSION_MAJOR
  require += info_simulate_version[argc];
#endif
#if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
  require += info_cray[argc];
#endif
  require += info_language_standard_default[argc];
  require += info_language_extensions_default[argc];
  (void)argv;
  return require;
}
#endif
```

5.3 cmake-build-release/CMakeFiles/3.26.4/CompilerIdC/CMake↩ CCompilerId.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.

```
#ifdef __cplusplus
# error "A C++ compiler has been selected for C."
#endif

#if defined(__18CXX)
# define ID_VOID_MAIN
#endif
#if defined(__CLASSIC_C__)
/* cv-qualifiers did not exist in K&R C */
# define const
# define volatile
#endif

#if !defined(__has_include)
/* If the compiler does not have __has_include, pretend the answer is
   always no.  */
#  define __has_include(x) 0
#endif
```

```


/* Version number components: V=Version, R=Revision, P=Patch
   Version date components:   YYYY=Year, MM=Month,   DD=Day  */

#if defined(__INTEL_COMPILER) || defined(__ICC)
# define COMPILER_ID "Intel"
# if defined(_MSC_VER)
#  define SIMULATE_ID "MSVC"
# endif
# if defined(__GNUC__)
#  define SIMULATE_ID "GNU"
# endif
  /* __INTEL_COMPILER = VRP prior to 2021, and then VVVV for 2021 and later,
     except that a few beta releases use the old format with V=2021.  */
# if __INTEL_COMPILER < 2021 || __INTEL_COMPILER == 202110 || __INTEL_COMPILER == 202111
#  define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER/100)
#  define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER/10 % 10)
#  if defined(__INTEL_COMPILER_UPDATE)
#   define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER_UPDATE)
#  else
#   define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER   % 10)
#  endif
# else
#  define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER)
#  define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER_UPDATE)
   /* The third version component from --version is an update index,
      but no macro is provided for it.  */
#  define COMPILER_VERSION_PATCH DEC(0)
# endif
# if defined(__INTEL_COMPILER_BUILD_DATE)
   /* __INTEL_COMPILER_BUILD_DATE = YYYYMMDD */
#  define COMPILER_VERSION_TWEAK DEC(__INTEL_COMPILER_BUILD_DATE)
# endif
# if defined(_MSC_VER)
   /* _MSC_VER = VVRR */
#  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif
# if defined(__GNUC__)
#  define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
# elif defined(__GNUG__)
#  define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
# endif
# if defined(__GNUC_MINOR__)
#  define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
# endif
# if defined(__GNUC_PATCHLEVEL__)
#  define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
# endif

#elif (defined(__clang__) && defined(__INTEL_CLANG_COMPILER)) || defined(__INTEL_LLVM_COMPILER)
# define COMPILER_ID "IntelLLVM"
#if defined(_MSC_VER)
# define SIMULATE_ID "MSVC"
#endif
#if defined(__GNUC__)
# define SIMULATE_ID "GNU"
#endif
/* __INTEL_LLVM_COMPILER = VVVVRP prior to 2021.2.0, VVVVRRPP for 2021.2.0 and
 * later.  Look for 6 digit vs. 8 digit version number to decide encoding.
 * VVVV is no smaller than the current year when a version is released.
 */
#if __INTEL_LLVM_COMPILER < 1000000L
# define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/100)
# define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER    % 10)
#else
# define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/10000)
# define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/100 % 100)
# define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER     % 100)
#endif
#if defined(_MSC_VER)
  /* _MSC_VER = VVRR */
# define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
# define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
#endif
#if defined(__GNUC__)
# define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
#elif defined(__GNUG__)
# define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
#endif
#if defined(__GNUC_MINOR__)
# define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
#endif
#if defined(__GNUC_PATCHLEVEL__)
# define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
```

```
#endif

#elif defined(__PATHCC__)
# define COMPILER_ID "PathScale"
# define COMPILER_VERSION_MAJOR DEC(__PATHCC__)
# define COMPILER_VERSION_MINOR DEC(__PATHCC_MINOR__)
# if defined(__PATHCC_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__PATHCC_PATCHLEVEL__)
# endif

#elif defined(__BORLANDC__) && defined(__CODEGEARC_VERSION__)
# define COMPILER_ID "Embarcadero"
# define COMPILER_VERSION_MAJOR HEX(__CODEGEARC_VERSION__»24 & 0x00FF)
# define COMPILER_VERSION_MINOR HEX(__CODEGEARC_VERSION__»16 & 0x00FF)
# define COMPILER_VERSION_PATCH DEC(__CODEGEARC_VERSION__     & 0xFFFF)

#elif defined(__BORLANDC__)
# define COMPILER_ID "Borland"
  /* __BORLANDC__ = 0xVRR */
# define COMPILER_VERSION_MAJOR HEX(__BORLANDC__»8)
# define COMPILER_VERSION_MINOR HEX(__BORLANDC__ & 0xFF)

#elif defined(__WATCOMC__) && __WATCOMC__ < 1200
# define COMPILER_ID "Watcom"
   /* __WATCOMC__ = VVRR */
# define COMPILER_VERSION_MAJOR DEC(__WATCOMC__ / 100)
# define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
# if (__WATCOMC__ % 10) > 0
#  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
# endif

#elif defined(__WATCOMC__)
# define COMPILER_ID "OpenWatcom"
   /* __WATCOMC__ = VVRP + 1100 */
# define COMPILER_VERSION_MAJOR DEC((__WATCOMC__ - 1100) / 100)
# define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
# if (__WATCOMC__ % 10) > 0
#  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
# endif

#elif defined(__SUNPRO_C)
# define COMPILER_ID "SunPro"
# if __SUNPRO_C >= 0x5100
   /* __SUNPRO_C = 0xVRRP */
#  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_C»12)
#  define COMPILER_VERSION_MINOR HEX(__SUNPRO_C»4 & 0xFF)
#  define COMPILER_VERSION_PATCH HEX(__SUNPRO_C    & 0xF)
# else
   /* __SUNPRO_CC = 0xVRP */
#  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_C»8)
#  define COMPILER_VERSION_MINOR HEX(__SUNPRO_C»4 & 0xF)
#  define COMPILER_VERSION_PATCH HEX(__SUNPRO_C    & 0xF)
# endif

#elif defined(__HP_cc)
# define COMPILER_ID "HP"
  /* __HP_cc = VVRRPP */
# define COMPILER_VERSION_MAJOR DEC(__HP_cc/10000)
# define COMPILER_VERSION_MINOR DEC(__HP_cc/100 % 100)
# define COMPILER_VERSION_PATCH DEC(__HP_cc     % 100)

#elif defined(__DECC)
# define COMPILER_ID "Compaq"
  /* __DECC_VER = VVRRTPPPP */
# define COMPILER_VERSION_MAJOR DEC(__DECC_VER/10000000)
# define COMPILER_VERSION_MINOR DEC(__DECC_VER/100000  % 100)
# define COMPILER_VERSION_PATCH DEC(__DECC_VER         % 10000)

#elif defined(__IBMC__) && defined(__COMPILER_VER__)
# define COMPILER_ID "zOS"
  /* __IBMC__ = VRP */
# define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
# define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)

#elif defined(__open_xl__) && defined(__clang__)
# define COMPILER_ID "IBMClang"
# define COMPILER_VERSION_MAJOR DEC(__open_xl_version__)
# define COMPILER_VERSION_MINOR DEC(__open_xl_release__)
# define COMPILER_VERSION_PATCH DEC(__open_xl_modification__)
# define COMPILER_VERSION_TWEAK DEC(__open_xl_ptf_fix_level__)


#elif defined(__ibmxl__) && defined(__clang__)
# define COMPILER_ID "XLClang"
# define COMPILER_VERSION_MAJOR DEC(__ibmxl_version__)
# define COMPILER_VERSION_MINOR DEC(__ibmxl_release__)
# define COMPILER_VERSION_PATCH DEC(__ibmxl_modification__)
# define COMPILER_VERSION_TWEAK DEC(__ibmxl_ptf_fix_level__)


#elif defined(__IBMC__) && !defined(__COMPILER_VER__) && __IBMC__ >= 800
# define COMPILER_ID "XL"
  /* __IBMC__ = VRP */
# define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
# define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)

#elif defined(__IBMC__) && !defined(__COMPILER_VER__) && __IBMC__ < 800
# define COMPILER_ID "VisualAge"
  /* __IBMC__ = VRP */
# define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
# define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)

#elif defined(__NVCOMPILER)
# define COMPILER_ID "NVHPC"
# define COMPILER_VERSION_MAJOR DEC(__NVCOMPILER_MAJOR__)
# define COMPILER_VERSION_MINOR DEC(__NVCOMPILER_MINOR__)
# if defined(__NVCOMPILER_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__NVCOMPILER_PATCHLEVEL__)
# endif

#elif defined(__PGI)
# define COMPILER_ID "PGI"
# define COMPILER_VERSION_MAJOR DEC(__PGIC__)
# define COMPILER_VERSION_MINOR DEC(__PGIC_MINOR__)
# if defined(__PGIC_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__PGIC_PATCHLEVEL__)
# endif

#elif defined(_CRAYC)
# define COMPILER_ID "Cray"
# define COMPILER_VERSION_MAJOR DEC(_RELEASE_MAJOR)
# define COMPILER_VERSION_MINOR DEC(_RELEASE_MINOR)

#elif defined(__TI_COMPILER_VERSION__)
# define COMPILER_ID "TI"
  /* __TI_COMPILER_VERSION__ = VVVRRRPPP */
# define COMPILER_VERSION_MAJOR DEC(__TI_COMPILER_VERSION__/1000000)
# define COMPILER_VERSION_MINOR DEC(__TI_COMPILER_VERSION__/1000   % 1000)
# define COMPILER_VERSION_PATCH DEC(__TI_COMPILER_VERSION__        % 1000)

#elif defined(__CLANG_FUJITSU)
# define COMPILER_ID "FujitsuClang"
# define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
# define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
# define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
# define COMPILER_VERSION_INTERNAL_STR __clang_version__


#elif defined(__FUJITSU)
# define COMPILER_ID "Fujitsu"
# if defined(__FCC_version__)
#   define COMPILER_VERSION __FCC_version__
# elif defined(__FCC_major__)
#   define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
#   define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
#   define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
# endif
# if defined(__fcc_version)
#   define COMPILER_VERSION_INTERNAL DEC(__fcc_version)
# elif defined(__FCC_VERSION)
#   define COMPILER_VERSION_INTERNAL DEC(__FCC_VERSION)
# endif


#elif defined(__ghs__)
# define COMPILER_ID "GHS"
/* __GHS_VERSION_NUMBER = VVVVRP */
# ifdef __GHS_VERSION_NUMBER
# define COMPILER_VERSION_MAJOR DEC(__GHS_VERSION_NUMBER / 100)
# define COMPILER_VERSION_MINOR DEC(__GHS_VERSION_NUMBER / 10 % 10)
# define COMPILER_VERSION_PATCH DEC(__GHS_VERSION_NUMBER      % 10)
# endif

#elif defined(__TASKING__)
# define COMPILER_ID "Tasking"
  # define COMPILER_VERSION_MAJOR DEC(__VERSION__/1000)
  # define COMPILER_VERSION_MINOR DEC(__VERSION__ % 100)
# define COMPILER_VERSION_INTERNAL DEC(__VERSION__)

#elif defined(__TINYC__)
# define COMPILER_ID "TinyCC"
```

```

#elif defined(__BCC__)
# define COMPILER_ID "Bruce"

#elif defined(__SCO_VERSION__)
# define COMPILER_ID "SCO"

#elif defined(__ARMCC_VERSION) && !defined(__clang__)
# define COMPILER_ID "ARMCC"
#if __ARMCC_VERSION >= 1000000
  /* __ARMCC_VERSION = VRRPPPP */
  # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/1000000)
  # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 100)
  # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION     % 10000)
#else
  /* __ARMCC_VERSION = VRPPPP */
  # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/100000)
  # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 10)
  # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION    % 10000)
#endif


#elif defined(__clang__) && defined(__apple_build_version__)
# define COMPILER_ID "AppleClang"
# if defined(_MSC_VER)
#  define SIMULATE_ID "MSVC"
# endif
# define COMPILER_VERSION_MAJOR DEC(__clang_major__)
# define COMPILER_VERSION_MINOR DEC(__clang_minor__)
# define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
# if defined(_MSC_VER)
   /* _MSC_VER = VVRR */
#  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif
# define COMPILER_VERSION_TWEAK DEC(__apple_build_version__)

#elif defined(__clang__) && defined(__ARMCOMPILER_VERSION)
# define COMPILER_ID "ARMClang"
  # define COMPILER_VERSION_MAJOR DEC(__ARMCOMPILER_VERSION/1000000)
  # define COMPILER_VERSION_MINOR DEC(__ARMCOMPILER_VERSION/10000 % 100)
  # define COMPILER_VERSION_PATCH DEC(__ARMCOMPILER_VERSION     % 10000)
# define COMPILER_VERSION_INTERNAL DEC(__ARMCOMPILER_VERSION)

#elif defined(__clang__)
# define COMPILER_ID "Clang"
# if defined(_MSC_VER)
#  define SIMULATE_ID "MSVC"
# endif
# define COMPILER_VERSION_MAJOR DEC(__clang_major__)
# define COMPILER_VERSION_MINOR DEC(__clang_minor__)
# define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
# if defined(_MSC_VER)
   /* _MSC_VER = VVRR */
#  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif

#elif defined(__LCC__) && (defined(__GNUC__) || defined(__GNUG__) || defined(__MCST__))
# define COMPILER_ID "LCC"
# define COMPILER_VERSION_MAJOR DEC(__LCC__ / 100)
# define COMPILER_VERSION_MINOR DEC(__LCC__ % 100)
# if defined(__LCC_MINOR__)
#  define COMPILER_VERSION_PATCH DEC(__LCC_MINOR__)
# endif
# if defined(__GNUC__) && defined(__GNUC_MINOR__)
#  define SIMULATE_ID "GNU"
#  define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
#  define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
#  if defined(__GNUC_PATCHLEVEL__)
#   define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
#  endif
# endif

#elif defined(__GNUC__)
# define COMPILER_ID "GNU"
# define COMPILER_VERSION_MAJOR DEC(__GNUC__)
# if defined(__GNUC_MINOR__)
#  define COMPILER_VERSION_MINOR DEC(__GNUC_MINOR__)
# endif
# if defined(__GNUC_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
# endif

#elif defined(_MSC_VER)
# define COMPILER_ID "MSVC"
  /* _MSC_VER = VVRR */
```

```
# define COMPILER_VERSION_MAJOR DEC(_MSC_VER / 100)
# define COMPILER_VERSION_MINOR DEC(_MSC_VER % 100)
# if defined(_MSC_FULL_VER)
#  if _MSC_VER >= 1400
    /* _MSC_FULL_VER = VVRRPPPPP */
#   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 100000)
#  else
    /* _MSC_FULL_VER = VVRRPPPP */
#   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 10000)
#  endif
# endif
# if defined(_MSC_BUILD)
#  define COMPILER_VERSION_TWEAK DEC(_MSC_BUILD)
# endif

#elif defined(_ADI_COMPILER)
# define COMPILER_ID "ADSP"
#if defined(__VERSIONNUM__)
  /* __VERSIONNUM__ = 0xVVRRPPTT */
#  define COMPILER_VERSION_MAJOR DEC(__VERSIONNUM__ » 24 & 0xFF)
#  define COMPILER_VERSION_MINOR DEC(__VERSIONNUM__ » 16 & 0xFF)
#  define COMPILER_VERSION_PATCH DEC(__VERSIONNUM__ » 8 & 0xFF)
#  define COMPILER_VERSION_TWEAK DEC(__VERSIONNUM__ & 0xFF)
#endif

#elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
# define COMPILER_ID "IAR"
# if defined(__VER__) && defined(__ICCARM__)
#  define COMPILER_VERSION_MAJOR DEC((__VER__) / 1000000)
#  define COMPILER_VERSION_MINOR DEC(((__VER__) / 1000) % 1000)
#  define COMPILER_VERSION_PATCH DEC((__VER__) % 1000)
#  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
# elif defined(__VER__) && (defined(__ICCAVR__) || defined(__ICCRX__) || defined(__ICCRH850__) ||
      defined(__ICCRL78__) || defined(__ICC430__) || defined(__ICCRISCV__) || defined(__ICCV850__) ||
      defined(__ICC8051__) || defined(__ICCSTM8__))
#  define COMPILER_VERSION_MAJOR DEC((__VER__) / 100)
#  define COMPILER_VERSION_MINOR DEC((__VER__) - (((__VER__) / 100)*100))
#  define COMPILER_VERSION_PATCH DEC(__SUBVERSION__)
#  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
# endif

#elif defined(__SDCC_VERSION_MAJOR) || defined(SDCC)
# define COMPILER_ID "SDCC"
# if defined(__SDCC_VERSION_MAJOR)
#  define COMPILER_VERSION_MAJOR DEC(__SDCC_VERSION_MAJOR)
#  define COMPILER_VERSION_MINOR DEC(__SDCC_VERSION_MINOR)
#  define COMPILER_VERSION_PATCH DEC(__SDCC_VERSION_PATCH)
# else
  /* SDCC = VRP */
#  define COMPILER_VERSION_MAJOR DEC(SDCC/100)
#  define COMPILER_VERSION_MINOR DEC(SDCC/10 % 10)
#  define COMPILER_VERSION_PATCH DEC(SDCC    % 10)
# endif


/* These compilers are either not known or too old to define an
  identification macro.  Try to identify the platform and guess that
  it is the native compiler.  */
#elif defined(__hpux) || defined(__hpua)
# define COMPILER_ID "HP"

#else /* unknown compiler */
# define COMPILER_ID ""
#endif

/* Construct the string literal in pieces to prevent the source from
   getting matched.  Store it in a pointer rather than an array
   because some compilers will just produce instructions to fill the
   array rather than assigning a pointer to a static array.  */
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]";
#ifdef SIMULATE_ID
char const* info_simulate = "INFO" ":" "simulate[" SIMULATE_ID "]";
#endif

#ifdef __QNXNTO__
char const* qnxnto = "INFO" ":" "qnxnto[]";
#endif

#if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
char const *info_cray = "INFO" ":" "compiler_wrapper[CrayPrgEnv]";
#endif

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

/* Identify known platforms by name.  */
#if defined(__linux) || defined(__linux__) || defined(linux)
```

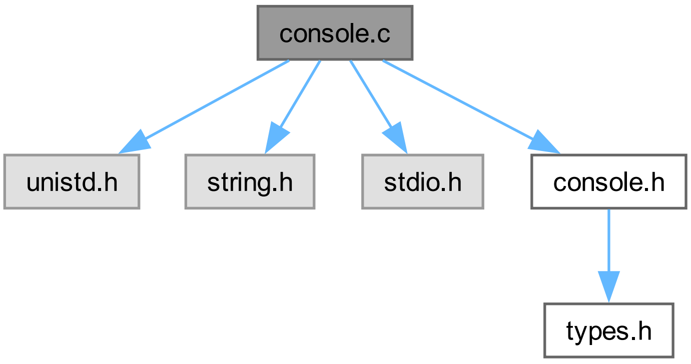
```
# define PLATFORM_ID "Linux"

#elif defined(__MSYS__)
# define PLATFORM_ID "MSYS"

#elif defined(__CYGWIN__)
# define PLATFORM_ID "Cygwin"

#elif defined(__MINGW32__)
# define PLATFORM_ID "MinGW"

#elif defined(__APPLE__)
# define PLATFORM_ID "Darwin"

#elif defined(_WIN32) || defined(__WIN32__) || defined(WIN32)
# define PLATFORM_ID "Windows"

#elif defined(__FreeBSD__) || defined(__FreeBSD)
# define PLATFORM_ID "FreeBSD"

#elif defined(__NetBSD__) || defined(__NetBSD)
# define PLATFORM_ID "NetBSD"

#elif defined(__OpenBSD__) || defined(__OPENBSD)
# define PLATFORM_ID "OpenBSD"

#elif defined(__sun) || defined(sun)
# define PLATFORM_ID "SunOS"

#elif defined(_AIX) || defined(__AIX) || defined(__AIX__) || defined(__aix) || defined(__aix__)
# define PLATFORM_ID "AIX"

#elif defined(__hpux) || defined(__hpux__)
# define PLATFORM_ID "HP-UX"

#elif defined(__HAIKU__)
# define PLATFORM_ID "Haiku"

#elif defined(__BeOS) || defined(__BEOS__) || defined(_BEOS)
# define PLATFORM_ID "BeOS"

#elif defined(__QNX__) || defined(__QNXNTO__)
# define PLATFORM_ID "QNX"

#elif defined(__tru64) || defined(_tru64) || defined(__TRU64__)
# define PLATFORM_ID "Tru64"

#elif defined(__riscos) || defined(__riscos__)
# define PLATFORM_ID "RISCos"

#elif defined(__sinix) || defined(__sinix__) || defined(__SINIX__)
# define PLATFORM_ID "SINIX"

#elif defined(__UNIX_SV__)
# define PLATFORM_ID "UNIX_SV"

#elif defined(__bsdos__)
# define PLATFORM_ID "BSDOS"

#elif defined(_MPRAS) || defined(MPRAS)
# define PLATFORM_ID "MP-RAS"

#elif defined(__osf) || defined(__osf__)
# define PLATFORM_ID "OSF1"

#elif defined(_SCO_SV) || defined(SCO_SV) || defined(sco_sv)
# define PLATFORM_ID "SCO_SV"

#elif defined(__ultrix) || defined(__ultrix__) || defined(_ULTRIX)
# define PLATFORM_ID "ULTRIX"

#elif defined(__XENIX__) || defined(_XENIX) || defined(XENIX)
# define PLATFORM_ID "Xenix"

#elif defined(__WATCOMC__)
# if defined(__LINUX__)
#  define PLATFORM_ID "Linux"

# elif defined(__DOS__)
#  define PLATFORM_ID "DOS"

# elif defined(__OS2__)
#  define PLATFORM_ID "OS2"

# elif defined(__WINDOWS__)
#  define PLATFORM_ID "Windows3x"

# elif defined(__VXWORKS__)
#  define PLATFORM_ID "VxWorks"

# else /* unknown platform */
#  define PLATFORM_ID
# endif

#elif defined(__INTEGRITY)
# if defined(INT_178B)
#  define PLATFORM_ID "Integrity178"

# else /* regular Integrity */
#  define PLATFORM_ID "Integrity"
# endif

# elif defined(_ADI_COMPILER)
#  define PLATFORM_ID "ADSP"

#else /* unknown platform */
# define PLATFORM_ID

#endif

/* For windows compilers MSVC and Intel we can determine
   the architecture of the compiler being used.  This is because
   the compilers do not have flags that can change the architecture,
   but rather depend on which compiler is being used
*/
#if defined(_WIN32) && defined(_MSC_VER)
# if defined(_M_IA64)
#  define ARCHITECTURE_ID "IA64"

# elif defined(_M_ARM64EC)
#  define ARCHITECTURE_ID "ARM64EC"

# elif defined(_M_X64) || defined(_M_AMD64)
#  define ARCHITECTURE_ID "x64"

# elif defined(_M_IX86)
#  define ARCHITECTURE_ID "X86"

# elif defined(_M_ARM64)
#  define ARCHITECTURE_ID "ARM64"

# elif defined(_M_ARM)
#  if _M_ARM == 4
#   define ARCHITECTURE_ID "ARMV4I"
#  elif _M_ARM == 5
#   define ARCHITECTURE_ID "ARMV5I"
#  else
#   define ARCHITECTURE_ID "ARMV" STRINGIFY(_M_ARM)
#  endif

# elif defined(_M_MIPS)
#  define ARCHITECTURE_ID "MIPS"

# elif defined(_M_SH)
#  define ARCHITECTURE_ID "SHx"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__WATCOMC__)
# if defined(_M_I86)
#  define ARCHITECTURE_ID "I86"

# elif defined(_M_IX86)
#  define ARCHITECTURE_ID "X86"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
# if defined(__ICCARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__ICCRX__)
#  define ARCHITECTURE_ID "RX"

# elif defined(__ICCRH850__)
#  define ARCHITECTURE_ID "RH850"

# elif defined(__ICCRL78__)
#  define ARCHITECTURE_ID "RL78"

# elif defined(__ICCRISCV__)
#  define ARCHITECTURE_ID "RISCV"

# elif defined(__ICCAVR__)
#  define ARCHITECTURE_ID "AVR"

# elif defined(__ICC430__)
#  define ARCHITECTURE_ID "MSP430"

# elif defined(__ICCV850__)
#  define ARCHITECTURE_ID "V850"

# elif defined(__ICC8051__)
#  define ARCHITECTURE_ID "8051"

# elif defined(__ICCSTM8__)
#  define ARCHITECTURE_ID "STM8"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__ghs__)
# if defined(__PPC64__)
#  define ARCHITECTURE_ID "PPC64"

# elif defined(__ppc__)
#  define ARCHITECTURE_ID "PPC"

# elif defined(__ARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__x86_64__)
#  define ARCHITECTURE_ID "x64"

# elif defined(__i386__)
#  define ARCHITECTURE_ID "X86"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__TI_COMPILER_VERSION__)
# if defined(__TI_ARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__MSP430__)
#  define ARCHITECTURE_ID "MSP430"

# elif defined(__TMS320C28XX__)
#  define ARCHITECTURE_ID "TMS320C28x"

# elif defined(__TMS320C6X__) || defined(_TMS320C6X)
#  define ARCHITECTURE_ID "TMS320C6x"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

# elif defined(__ADSPSHARC__)
#  define ARCHITECTURE_ID "SHARC"

# elif defined(__ADSPBLACKFIN__)
#  define ARCHITECTURE_ID "Blackfin"

#elif defined(__TASKING__)

# if defined(__CTC__) || defined(__CPTC__)
#  define ARCHITECTURE_ID "TriCore"

# elif defined(__CMCS__)
#  define ARCHITECTURE_ID "MCS"

# elif defined(__CARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__CARC__)
#  define ARCHITECTURE_ID "ARC"

# elif defined(__C51__)
#  define ARCHITECTURE_ID "8051"

# elif defined(__CPCP__)
#  define ARCHITECTURE_ID "PCP"

# else
#  define ARCHITECTURE_ID ""
# endif

#else
#  define ARCHITECTURE_ID
#endif

/* Convert integer to decimal digit literals.  */
#define DEC(n)                   \
  ('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))

/* Convert integer to hex digit literals.  */
#define HEX(n)             \
  ('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))

/* Construct a string literal encoding the version number. */
#ifdef COMPILER_VERSION
char const* info_version = "INFO" ":" "compiler_version[" COMPILER_VERSION "]";

/* Construct a string literal encoding the version number components. */
#elif defined(COMPILER_VERSION_MAJOR)
char const info_version[] = {
  'I', 'N', 'F', 'O', ':',
  'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','[',
  COMPILER_VERSION_MAJOR,
# ifdef COMPILER_VERSION_MINOR
  '.', COMPILER_VERSION_MINOR,
#  ifdef COMPILER_VERSION_PATCH
   '.', COMPILER_VERSION_PATCH,
#   ifdef COMPILER_VERSION_TWEAK
    '.', COMPILER_VERSION_TWEAK,
#   endif
#  endif
# endif
  ']','\0'};
#endif

/* Construct a string literal encoding the internal version number. */
#ifdef COMPILER_VERSION_INTERNAL
char const info_version_internal[] = {
  'I', 'N', 'F', 'O', ':',
  'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','_',
  'i','n','t','e','r','n','a','l','[',
  COMPILER_VERSION_INTERNAL,']','\0'};
#elif defined(COMPILER_VERSION_INTERNAL_STR)
char const* info_version_internal = "INFO" ":" "compiler_version_internal["
      COMPILER_VERSION_INTERNAL_STR "]";
#endif

/* Construct a string literal encoding the version number components. */
#ifdef SIMULATE_VERSION_MAJOR
char const info_simulate_version[] = {
  'I', 'N', 'F', 'O', ':',
  's','i','m','u','l','a','t','e','_','v','e','r','s','i','o','n','[',
  SIMULATE_VERSION_MAJOR,
# ifdef SIMULATE_VERSION_MINOR
  '.', SIMULATE_VERSION_MINOR,
#  ifdef SIMULATE_VERSION_PATCH
   '.', SIMULATE_VERSION_PATCH,
#   ifdef SIMULATE_VERSION_TWEAK
    '.', SIMULATE_VERSION_TWEAK,
#   endif
#  endif
# endif
  ']','\0'};
#endif

/* Construct the string literal in pieces to prevent the source from
   getting matched.  Store it in a pointer rather than an array
   because some compilers will just produce instructions to fill the
   array rather than assigning a pointer to a static array.  */
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]";
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]";

```

```


#if !defined(__STDC__) && !defined(__clang__)
# if defined(_MSC_VER) || defined(__ibmxl__) || defined(__IBMC__)
#  define C_VERSION "90"
# else
#  define C_VERSION
# endif
#elif __STDC_VERSION__ > 201710L
# define C_VERSION "23"
#elif __STDC_VERSION__ >= 201710L
# define C_VERSION "17"
#elif __STDC_VERSION__ >= 201000L
# define C_VERSION "11"
#elif __STDC_VERSION__ >= 199901L
# define C_VERSION "99"
#else
# define C_VERSION "90"
#endif
const char* info_language_standard_default =
  "INFO" ":" "standard_default[" C_VERSION "]";

const char* info_language_extensions_default = "INFO" ":" "extensions_default["
#if (defined(__clang__) || defined(__GNUC__) || defined(__xlC__) ||           \
     defined(__TI_COMPILER_VERSION__)) &&                                     \
  !defined(__STRICT_ANSI__)
  "ON"
#else
  "OFF"
#endif
"]";

/*--------------------------------------------------------------------------*/

#ifdef ID_VOID_MAIN
void main() {}
#else
# if defined(__CLASSIC_C__)
int main(argc, argv) int argc; char *argv[];
# else
int main(int argc, char* argv[])
# endif
{
  int require = 0;
  require += info_compiler[argc];
  require += info_platform[argc];
  require += info_arch[argc];
#ifdef COMPILER_VERSION_MAJOR
  require += info_version[argc];
#endif
#ifdef COMPILER_VERSION_INTERNAL
  require += info_version_internal[argc];
#endif
#ifdef SIMULATE_ID
  require += info_simulate[argc];
#endif
#ifdef SIMULATE_VERSION_MAJOR
  require += info_simulate_version[argc];
#endif
#if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
  require += info_cray[argc];
#endif
  require += info_language_standard_default[argc];
  require += info_language_extensions_default[argc];
  (void)argv;
  return require;
}
#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

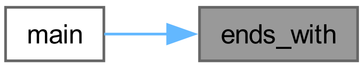
file	file path

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

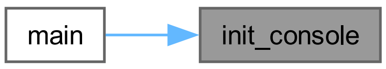
argc	argument_cont from main
argv	arguments list from main
theme_file	theme file path
output_file	output file path

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

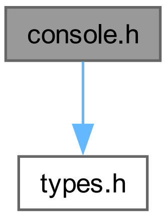
Go to the documentation of this file.

```c
//
// Created by sziha on 16/10/2023.
//
#include <unistd.h>
#include <string.h>
#include <stdio.h>
#include "console.h"
int init_console(int argc, char** argv, char* theme_file, char* output_file, char* in_file) {
    int c;
    while ((c = getopt(argc,argv,":o:t:h")) != -1) {
        switch (c) {
            case 't':
                strcpy(theme_file, optarg);
                break;
            case 'h':
                printf("Usage: console -t <theme_file> -o <output_file> -i <input_file>\n");
                return  -1;
            case 'o':
                strcpy(output_file, optarg);
                break;
            case ':':
                strcpy(in_file, optarg);
                break;
            case '?':
                if (optopt == 'o') {
                    fprintf(stderr, "Option -%c requires an argument.\n", optopt);
                } else if (optopt == 't') {
                    fprintf(stderr, "Option -%c requires an argument.\n", optopt);
                } else {
                    fprintf(stderr, "Unknown option `-%c'.\n", optopt);
                }
            default:
                break;
        }
    }
    return 0;
}

file_type_e ends_with(char* file)
{
    char* fileExt = strrchr(file, '.');
    if (strcmp(fileExt, file) != 0 || fileExt != NULL)
    {
        if (strcmp(fileExt, ".jpg") == 0)
            return file_type_jpg;
        else if (strcmp(fileExt, ".png") == 0)
            return file_type_png;
        else if (strcmp(fileExt, ".c") == 0)
            return file_type_c;
        else if (strcmp(fileExt, ".h") == 0)
            return file_type_h;
        /*else if (strcmp(fileExt, "md") == 0)
            return file_type_md;*/
    }
    return DEFAULT_FILE_TYPE;
}
```

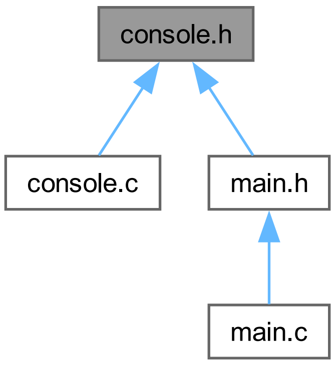
5.7 console.h File Reference

```
#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

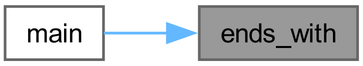
file	file path

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

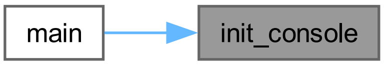
argc	argument_cont from main
argv	arguments list from main
theme_file	theme file path
output_file	output file path

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

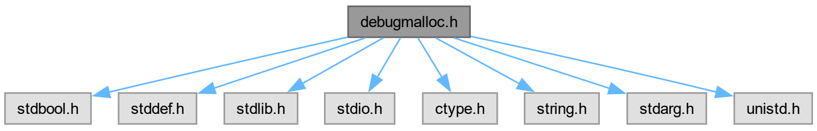
Go to the documentation of this file.

```
//
// Created by sziha on 16/10/2023.
//

#ifndef NHF_CONSOLE_H
#define NHF_CONSOLE_H
#include "types.h"
int init_console(int argc, char** argv, char* theme_file, char* output_file, char* in_file);
file_type_e ends_with(char* file);
#endif //NHF_CONSOLE_H
```

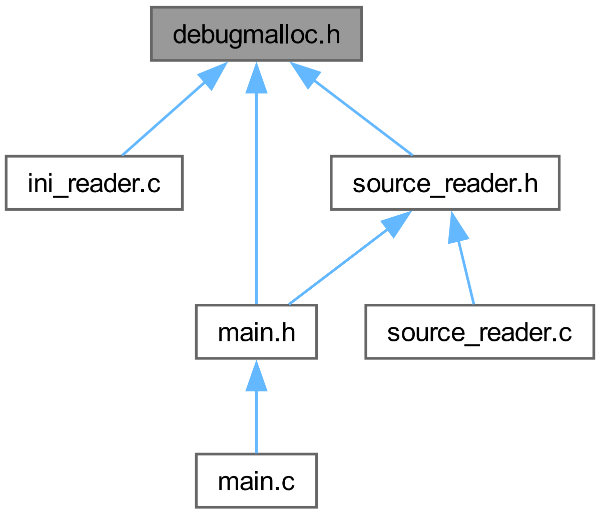
5.9 debugmalloc.h File Reference

```
#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 */
```

```
typedef struct DebugmallocData {
    char logfile[256];    /* log file name or empty string */
    long max_block_size;  /* max size of a single block allocated */
    long alloc_count;     /* currently allocated; decreased with free */
    long long alloc_bytes;
    long all_alloc_count; /* all allocations, never decreased */
    long long all_alloc_bytes;
    DebugmallocEntry head[debugmalloc_tablesize], tail[debugmalloc_tablesize];  /* head and tail
elements of allocation lists */
} DebugmallocData;


/* this forward declaration is required by the singleton manager function */
static DebugmallocData * debugmalloc_create(void);


/* creates singleton instance. as this function is static included to different
 * translation units, multiple instances of the static variables are created.
 * to make sure it is really a singleton, these instances must know each other
 * somethow. an environment variable is used for that purpose, ie. the address
 * of the singleton allocated is stored by the operating system.
 * this implementation is not thread-safe. */
static DebugmallocData * debugmalloc_singleton(void) {
    static char envstr[100];
    static void *instance = NULL;

    /* if we do not know the address of the singleton:
     * - maybe we are the one to create it (env variable also does not exist)
     * - or it is already created, and stored in the env variable. */
    if (instance == NULL) {
        char envvarname[100] = "";
        sprintf(envvarname, "%s%d", "debugmallocsingleton", (int) getpid());
        char *envptr = getenv(envvarname);
        if (envptr == NULL) {
            /* no env variable: create singleton. */
            instance = debugmalloc_create();
            sprintf(envstr, "%s=%p", envvarname, instance);
            putenv(envstr);
        } else {
            /* another copy of this function already created it. */
            int ok = sscanf(envptr, "%p", &instance);
            if (ok != 1) {
                fprintf(stderr, "debugmalloc: nem lehet ertelmezni: %s!\n", envptr);
                abort();
            }
        }
    }

    return (DebugmallocData *) instance;
}


/* better version of strncpy, always terminates string with \0. */
static void debugmalloc_strlcpy(char *dest, char const *src, size_t destsize) {
    strncpy(dest, src, destsize);
    dest[destsize - 1] = '\0';
}


/* set the name of the log file for debugmalloc. empty filename
 * means logging to stderr. */
static void debugmalloc_log_file(char const *logfilename) {
    if (logfilename == NULL)
        logfilename = "";
    DebugmallocData *instance = debugmalloc_singleton();
    debugmalloc_strlcpy(instance->logfile, logfilename, sizeof(instance->logfile));
}


/* set the maximum size of one block. useful for debugging purposes. */
static void debugmalloc_max_block_size(long max_block_size) {
    DebugmallocData *instance = debugmalloc_singleton();
    instance->max_block_size = max_block_size;
}



/* printf to the log file, or stderr. */
static void debugmalloc_log(char const *format, ...) {
    DebugmallocData *instance = debugmalloc_singleton();
    FILE *f = stderr;
    if (instance->logfile[0] != '\0') {
        f = fopen(instance->logfile, "at");
        if (f == NULL) {
            f = stderr;
            fprintf(stderr, "debugmalloc: nem tudom megnyitni a %s fajlt irasra!\n",
instance->logfile);
```

```
            debugmalloc_strlcpy(instance->logfile, "", sizeof(instance->logfile));
        }
    }

    va_list ap;
    va_start(ap, format);
    vfprintf(f, format, ap);
    va_end(ap);

    if (f != stderr)
        fclose(f);
}


/* initialize a memory block allocated for the user. the start and the end
 * of the block is initialized with the canary characters. if 'zero' is
 * true, the user memory area is zero-initialized, otherwise it is also
 * filled with the canary character to simulate garbage in memory. */
static void debugmalloc_memory_init(DebugmallocEntry *elem, bool zero) {
    unsigned char *real_mem = (unsigned char *) elem->real_mem;
    unsigned char *user_mem = (unsigned char *) elem->user_mem;
    unsigned char *canary1 = real_mem;
    unsigned char *canary2 = real_mem + debugmalloc_canary_size + elem->size;
    memset(canary1, debugmalloc_canary_char, debugmalloc_canary_size);
    memset(canary2, debugmalloc_canary_char, debugmalloc_canary_size);
    memset(user_mem, zero ? 0 : debugmalloc_canary_char, elem->size);
}

/* check canary, return true if ok, false if corrupted. */
static bool debugmalloc_canary_ok(DebugmallocEntry const *elem) {
    unsigned char *real_mem = (unsigned char *) elem->real_mem;
    unsigned char *canary1 = real_mem;
    unsigned char *canary2 = real_mem + debugmalloc_canary_size + elem->size;
    for (size_t i = 0; i < debugmalloc_canary_size; ++i) {
        if (canary1[i] != debugmalloc_canary_char)
            return false;
        if (canary2[i] != debugmalloc_canary_char)
            return false;
    }
    return true;
}


/* dump memory contents to log file. */
static void debugmalloc_dump_memory(char const *mem, size_t size) {
    for (unsigned y = 0; y < (size + 15) / 16; y++) {
        char line[80];
        int pos = 0;
        pos += sprintf(line + pos, "      %04x  ", y * 16);
        for (unsigned x = 0; x < 16; x++) {
            if (y * 16 + x < size)
                pos += sprintf(line + pos, "%02x ", mem[y * 16 + x]);
            else
                pos += sprintf(line + pos, "   ");
        }
        pos += sprintf(line + pos, "  ");
        for (unsigned x = 0; x < 16; x++) {
            if (y * 16 + x < size) {
                unsigned char c = mem[y * 16 + x];
                pos += sprintf(line + pos, "%c", isprint(c) ? c : '.');
            }
            else {
                pos += sprintf(line + pos, " ");
            }
        }
        debugmalloc_log("%s\n", line);
    }
}


/* dump data of allocated memory block.
 * if the canary is corrupted, it is also written to the log. */
static void debugmalloc_dump_elem(DebugmallocEntry const *elem) {
    bool canary_ok = debugmalloc_canary_ok(elem);

    debugmalloc_log("  %p, %u bajt, kanari: %s\n"
                    "  %s:%u, %s(%s)\n",
                       elem->user_mem, (unsigned) elem->size, canary_ok ? "ok" : "**SERULT**",
                       elem->file, elem->line,
                       elem->func, elem->expr);

    if (!canary_ok) {
        debugmalloc_log("    ELOTTE kanari: \n");
        debugmalloc_dump_memory((char const *) elem->real_mem, debugmalloc_canary_size);
    }

    debugmalloc_dump_memory((char const *) elem->user_mem, elem->size > 64 ? 64 : elem->size);

    if (!canary_ok) {
        debugmalloc_log("    UTANA kanari: \n");
        debugmalloc_dump_memory((char const *) elem->real_mem + debugmalloc_canary_size + elem->size,
debugmalloc_canary_size);
    }
}


/* dump data of all memory blocks allocated. */
static void debugmalloc_dump(void) {
    DebugmallocData *instance = debugmalloc_singleton();
    debugmalloc_log("** DEBUGMALLOC DUMP ************************************\n");
    int cnt = 0;
    for (size_t i = 0; i < debugmalloc_tablesize; i++) {
        DebugmallocEntry *head = &instance->head[i];
        for (DebugmallocEntry *iter = head->next; iter->next != NULL; iter = iter->next) {
            ++cnt;
            debugmalloc_log("** %d/%d. rekord:\n", cnt, instance->alloc_count);
            debugmalloc_dump_elem(iter);
        }
    }
    debugmalloc_log("** DEBUGMALLOC DUMP VEGE *******************************\n");
}


/* called at program exit to dump data if there is a leak,
 * ie. allocated block remained. */
static void debugmalloc_atexit_dump(void) {
    DebugmallocData *instance = debugmalloc_singleton();

    if (instance->alloc_count > 0) {
        debugmalloc_log("\n"
                        "********************************************************\n"
                        "* MEMORIASZIVARGAS VAN A PROGRAMBAN!!!\n"
                        "********************************************************\n"
                        "\n");
        debugmalloc_dump();
    } else {
        debugmalloc_log("****************************************************\n"
                        "* Debugmalloc: nincs memoriaszivargas a programban.\n"
                        "* Osszes foglalas: %d blokk, %d bajt.\n"
                        "****************************************************\n",
                        instance->all_alloc_count, instance->all_alloc_bytes);
    }
}


/* hash function for bucket hash. */
static size_t debugmalloc_hash(void *address) {
    /* the last few bits are ignored, as they are usually zero for
     * alignment purposes. all tested architectures used 16 byte allocation. */
    size_t cut = (size_t)address » 4;
    return cut % debugmalloc_tablesize;
}


/* insert element to hash table. */
static void debugmalloc_insert(DebugmallocEntry *entry) {
    DebugmallocData *instance = debugmalloc_singleton();
    size_t idx = debugmalloc_hash(entry->user_mem);
    DebugmallocEntry *head = &instance->head[idx];
    entry->prev = head;
    entry->next = head->next;
    head->next->prev = entry;
    head->next = entry;
    instance->alloc_count += 1;
    instance->alloc_bytes += entry->size;
    instance->all_alloc_count += 1;
    instance->all_alloc_bytes += entry->size;
}


/* remove element from hash table */
static void debugmalloc_remove(DebugmallocEntry *entry) {
    DebugmallocData *instance = debugmalloc_singleton();
    entry->next->prev = entry->prev;
    entry->prev->next = entry->next;
    instance->alloc_count -= 1;
    instance->alloc_bytes -= entry->size;
}


/* find element in hash table, given with the memory address that the user sees.
 * @return the linked list entry, or null if not found. */
static DebugmallocEntry *debugmalloc_find(void *mem) {
    DebugmallocData *instance = debugmalloc_singleton();
    size_t idx = debugmalloc_hash(mem);
    DebugmallocEntry *head = &instance->head[idx];
    for (DebugmallocEntry *iter = head->next; iter->next != NULL; iter = iter->next)
        if (iter->user_mem == mem)
            return iter;
    return NULL;
}


/* allocate memory. this function is called via the macro. */
static void *debugmalloc_malloc_full(size_t size, char const *func, char const *expr, char const
*file, unsigned line, bool zero) {
    /* imitate standard malloc: return null if size is zero */
    if (size == 0)
        return NULL;

    /* check max size */
    DebugmallocData *instance = debugmalloc_singleton();
    if (size > (long long unsigned int) instance->max_block_size) {
        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);
        abort();
    }

    /* allocate more memory, make room for canary */
    void *real_mem = malloc(size + 2 * debugmalloc_canary_size);
    if (real_mem == NULL) {
        debugmalloc_log("debugmalloc: %s @ %s:%u: nem sikerult %u meretu memoriat foglalni!\n", func,
file, line, (unsigned) size);
        return NULL;
    }

    /* allocate memory for linked list element */
    DebugmallocEntry *newentry = (DebugmallocEntry *) malloc(sizeof(DebugmallocEntry));
    if (newentry == NULL) {
        free(real_mem);
        debugmalloc_log("debugmalloc: %s @ %s:%u: le tudtam foglalni %u memoriat, de utana a sajatnak
nem, sry\n", func, file, line, (unsigned) size);
        abort();
    }

    /* metadata of allocation: caller function, code line etc. */
    debugmalloc_strlcpy(newentry->func, func, sizeof(newentry->func));
    debugmalloc_strlcpy(newentry->expr, expr, sizeof(newentry->expr));
    debugmalloc_strlcpy(newentry->file, file, sizeof(newentry->file));
    newentry->line = line;

    /* address of allocated memory chunk */
    newentry->real_mem = real_mem;
    newentry->user_mem = (unsigned char *) real_mem + debugmalloc_canary_size;
    newentry->size = size;
    debugmalloc_memory_init(newentry, zero);

    /* store in list and return pointer to user area */
    debugmalloc_insert(newentry);
    return newentry->user_mem;
}


/* free memory and remove list item. before deleting, the chuck is filled with
 * the canary byte to make sure that the user will see garbage if the memory
 * is accessed after freeing. */
static void debugmalloc_free_inner(DebugmallocEntry *deleted) {
    debugmalloc_remove(deleted);

    /* fill with garbage, then remove from linked list */
    memset(deleted->real_mem, debugmalloc_canary_char, deleted->size + 2 * debugmalloc_canary_size);
    free(deleted->real_mem);
    free(deleted);
}


/* free memory - called via the macro.
 * as all allocations are tracked in the list, this function can terminate the program
 * if a block is freed twice or the free function is called with an invalid address. */
static void debugmalloc_free_full(void *mem, char const *func, char const *file, unsigned line) {
    /* imitate standard free function: if ptr is null, no operation is performed */
    if (mem == NULL)
        return;

    /* find allocation, abort if not found */
    DebugmallocEntry *deleted = debugmalloc_find(mem);
    if (deleted == NULL) {
        debugmalloc_log("debugmalloc: %s @ %s:%u: olyan teruletet probalsz felszabaditani, ami nincs
lefoglalva!\n", func, file, line);
        abort();
    }
```

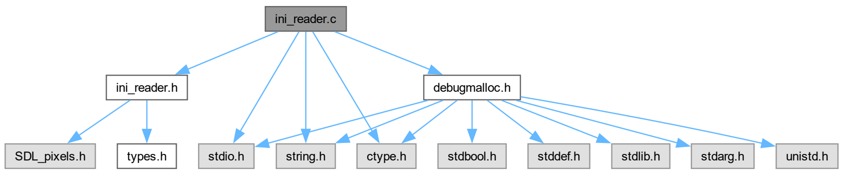
```c

    /* check canary and then free memory */
    if (!debugmalloc_canary_ok(deleted)) {
        debugmalloc_log("debugmalloc: %s @ %s:%u: a %p memoriateruletet tulindexelted!\n", func, file, line, mem);
        debugmalloc_dump_elem(deleted);
    }
    debugmalloc_free_inner(deleted);
}


/* realloc-like function. */
static void *debugmalloc_realloc_full(void *oldmem, size_t newsize, char const *func, char const *expr, char const *file, unsigned line) {
    /* imitate standard realloc: equivalent to free if size is null. */
    if (newsize == 0) {
        debugmalloc_free_full(oldmem, func, file, line);
        return NULL;
    }
    /* imitate standard realloc: equivalent to malloc if first param is NULL */
    if (oldmem == NULL)
        return debugmalloc_malloc_full(newsize, func, expr, file, line, 0);

    /* find old allocation. abort if not found. */
    DebugmallocEntry *oldentry = debugmalloc_find(oldmem);
    if (oldentry == NULL) {
        debugmalloc_log("debugmalloc: %s @ %s:%u: olyan teruletet probalsz atmeretezni, ami nincs lefoglalva!\n", func, file, line);
        abort();
    }

    /* create new allocation, copy & free old data */
    void *newmem = debugmalloc_malloc_full(newsize, func, expr, file, line, false);
    if (newmem == NULL) {
        debugmalloc_log("debugmalloc: %s @ %s:%u: nem sikerult uj memoriat foglalni az atmeretezeshez!\n", func, file, line);
        /* imitate standard realloc: original block is untouched, but return NULL */
        return NULL;
    }
    size_t smaller = oldentry->size < newsize ? oldentry->size : newsize;
    memcpy(newmem, oldmem, smaller);
    debugmalloc_free_inner(oldentry);

    return newmem;
}


/* initialize debugmalloc singleton. returns the newly allocated instance */
static DebugmallocData * debugmalloc_create(void) {
    /* config check */
    if (debugmalloc_canary_size % 16 != 0) {
        debugmalloc_log("debugmalloc: a kanari merete legyen 16-tal oszthato\n");
        abort();
    }
    if (debugmalloc_canary_char == 0) {
        debugmalloc_log("debugmalloc: a kanari legyen 0-tol kulonbozo\n");
        abort();
    }
    /* avoid compiler warning if these functions are not used */
    (void) debugmalloc_realloc_full;
    (void) debugmalloc_log_file;
    (void) debugmalloc_max_block_size;

    /* create and initialize instance */
    DebugmallocData *instance = (DebugmallocData *) malloc(sizeof(DebugmallocData));
    if (instance == NULL) {
        debugmalloc_log("debugmalloc: nem sikerult elinditani a memoriakezelest\n");
        abort();
    }
    debugmalloc_strlcpy(instance->logfile, "", sizeof(instance->logfile));
    instance->max_block_size = debugmalloc_max_block_size_default;
    instance->alloc_count = 0;
    instance->alloc_bytes = 0;
    instance->all_alloc_count = 0;
    instance->all_alloc_bytes = 0;
    for (size_t i = 0; i < debugmalloc_tablesize; i++) {
        instance->head[i].prev = NULL;
        instance->head[i].next = &instance->tail[i];
        instance->tail[i].next = NULL;
        instance->tail[i].prev = &instance->head[i];
    }

    atexit(debugmalloc_atexit_dump);
    return instance;
}


```

```
/* These macro-like functions forward all allocation/free
 * calls to debugmalloc. Usage is the same, malloc(size)
 * gives the address of a new memory block, free(ptr)
 * deallocates etc.
 *
 * If you use this file, make sure that you include this
 * in *ALL* translation units (*.c) of your source. The
 * builtin free() function cannot deallocate a memory block
 * that was allocated via debugmalloc, yet the name of
 * the function is the same! */

#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__)

#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

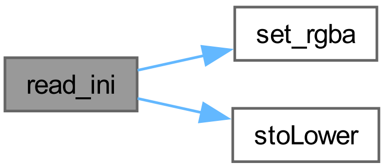
filename	name of the ini file (including the .ini)
theme	pointer to the theme variable

Returns

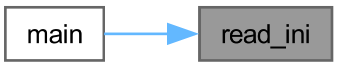
0 if ok, 1 if error

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 )
```

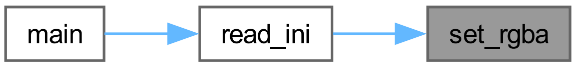
Sets the rgba of an SDL_Colour.

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 )
```

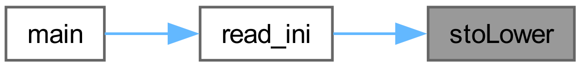
turns a string to lowercase

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.

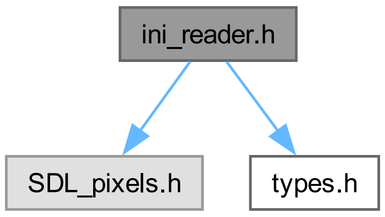
```
//
// Created by sziha on 16/10/2023.
//

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

int read_ini(const char *filename, theme_t *theme) {
    FILE *fp = fopen(filename, "r");
    if (fp == NULL) {
```

```c
        fprintf(stderr,"Couldn't open file");
        return -1;
    }
    mapping_t mappings_sc[2];
    mapping_t mappings_c[] = {
            {"functions", &(theme->functions)},
            {"structs", &(theme->structs)},
            {"vars", &(theme->variables)},
            {"conds", &(theme->conditionals)},
            {"loops", &(theme->loops)},
            {"main", &(theme->main_)},
    };
    //printf("file open\n") ;
    char line[256]; //if context == -1 go to next line else check subcontext and add to theme
    colour_t *context = NULL;
    SDL_Colour *subContext = NULL;
    while (fgets(line, 256, fp) != NULL) {
        //printf("%s", line);
        if (line[0] == '[') {
            char *name = line + 1;
            name = strtok(name, "]");
            stoLower(name);
            //printf("%s\n", name);
            for (int i = 0; i < 6; i++) {
                if (strcmp(name, mappings_c[i].key) == 0) {
                    context = (colour_t *) mappings_c[i].value;
                    break;
                }
            }
            if (context == NULL) {
                fprintf(stderr, "Unknown context: %s\n", name);
            }
            mappings_sc[0].key = "background";
            mappings_sc[0].value = &(context->background);
            mappings_sc[1].key = "text";
            mappings_sc[1].value = &(context->text);
            continue;
        }
        else if (line[0] != ';' && context != NULL)
        {
            char *value = strtok(line, "=");
            unsigned long long int val_len = strlen(value);
            //printf("%s", value);
            char *valend = value+val_len-1;
            while (isspace(*valend)){
                *valend = '\0';
                valend--;
            }
            for (int i = 0; i < 2; i++) {
                if (strcmp(value, mappings_sc[i].key) == 0) {
                    subContext = (SDL_Color *) mappings_sc[i].value;
                    break;
                }
            }
            if (subContext == NULL) {
                fprintf(stderr, "Unknown sub context: %s\n", value);
            }
            value = strtok(NULL, "=");
            while (isspace(*value))
                value++;
            set_rgba(value, subContext);
        }
    }
    fclose(fp);
    return 0;
}

void stoLower(char *str) {
    while (*str != '\0') {
        *str = (char) tolower(*str);
        str++;
    }
}

void set_rgba(char *hex, SDL_Colour *colour) {
    char rgba[3] = {hex[1], hex[2],'\0'};
    colour->r = strtoul(rgba, NULL, 16);
    rgba[0] = hex[3]; rgba[1] = hex[4];
    colour->g = strtoul(rgba, NULL, 16);
    rgba[0] = hex[5]; rgba[1] = hex[6];
    colour->b = strtoul(rgba, NULL, 16);
    rgba[0] = hex[7]; rgba[1] = hex[8];
    colour->a = strtoul(rgba, NULL, 16);
}
```

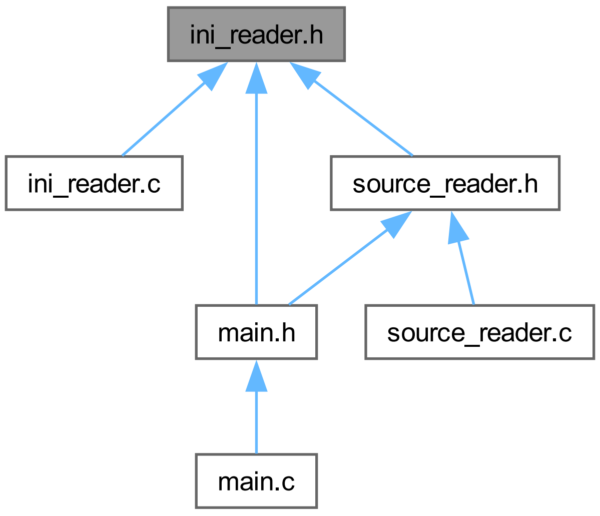
5.13 ini_reader.h File Reference

```
#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

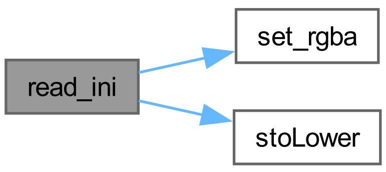
filename	name of the ini file (including the .ini)
theme	pointer to the theme variable

Returns

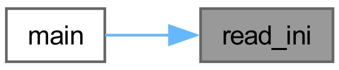
0 if ok, 1 if error

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 )
```

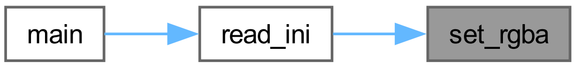
Sets the rgba of an SDL_Colour.

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 )
```

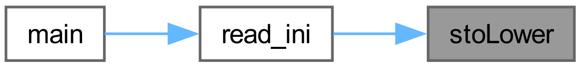
turns a string to lowercase

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

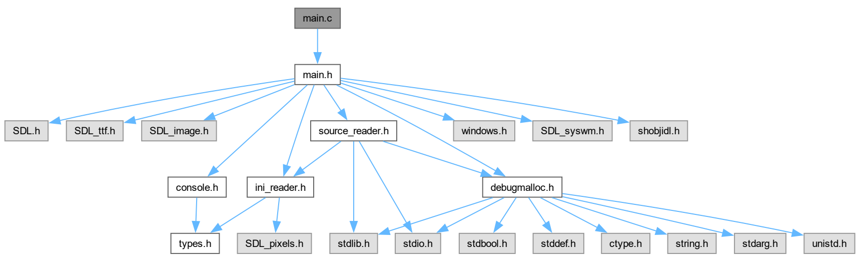
Go to the documentation of this file.

```
//
// Created by sziha on 16/10/2023.
//

#ifndef NHF_INI_READER_H
#define NHF_INI_READER_H

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

typedef enum {
    function,
    structs,
    variable,
    conditional,
    loop,
    main_
} context_e;
typedef enum {
    background,
    text
} sub_context_e;

typedef struct {
    SDL_Colour background;
    SDL_Colour text;
} colour_t;

typedef struct {
    colour_t functions;
    colour_t structs;
    colour_t variables;
    colour_t conditionals;
    colour_t loops;
    colour_t main_;
} theme_t;

int read_ini(const char *filename, theme_t *theme);
void stoLower(char *str);
void set_rgba(char *hex, SDL_Colour *colour);
#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 )
```

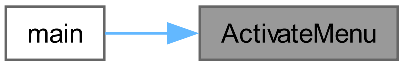
Creates a menu for the given window handle.

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

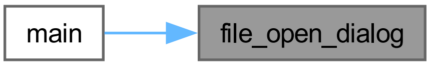
windowRef	

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

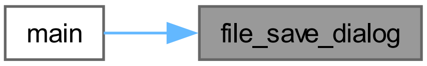
windowRef	

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

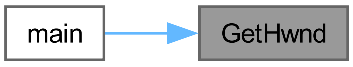
window	sdl window

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.

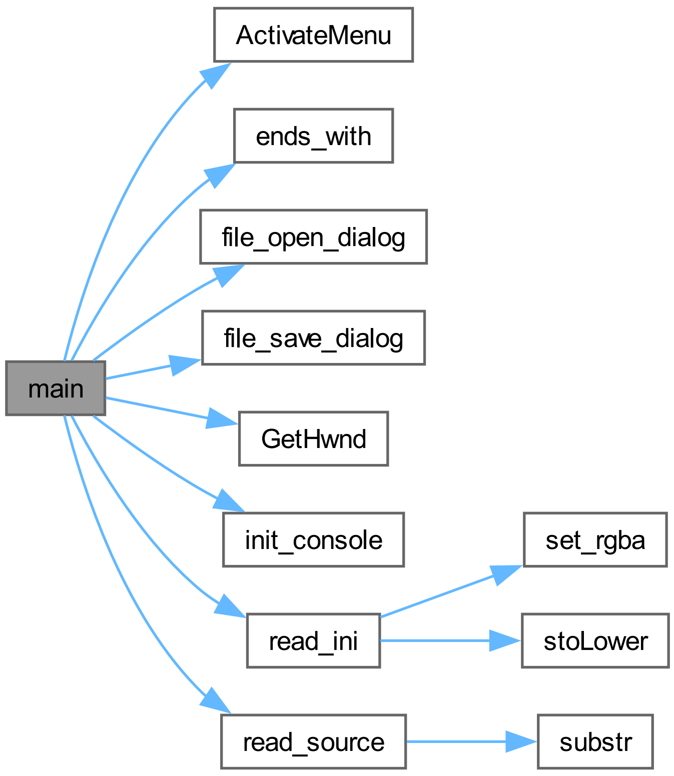
Parameters

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.

```
#include "main.h"

int main(int argc, char** argv) {
    char *theme_file = "theme.ini";
    char *input_file = "test.c";
    char *output_file = "test";
    theme_t *theme = &default_theme;
    printf("%u %u %u", theme->main_.background.r, theme->main_.background.g,
theme->main_.background.b);
    file_type_e output_type;
    file_type_e input_type;
    node_t *root = NULL;
    int quit = 0;
    HWND windowRef;
    SDL_Event event;

    if (argc != 1){
        if(init_console(argc, argv, theme_file, output_file, input_file) == -1) return -1;
```

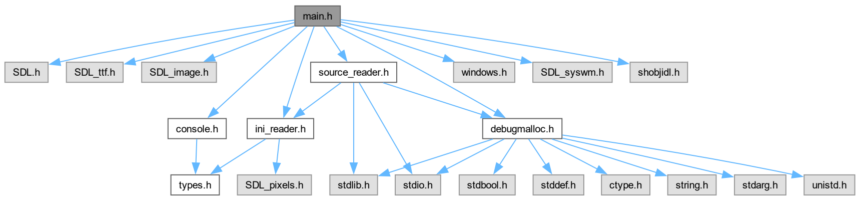
```c
        output_type = ends_with(output_file);
        input_type = ends_with(input_file);
    }
    if (read_ini(theme_file, theme) == -1) theme = &default_theme;
    SDL_Window *window = SDL_CreateWindow("Source to Flow", SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED, 640, 480, SDL_WINDOW_SHOWN | SDL_WINDOW_RESIZABLE);
    windowRef = GetHwnd(window);
    ActivateMenu(windowRef);
    SDL_Renderer *renderer = SDL_CreateRenderer(window, -1, 0);
    SDL_Surface *surface = SDL_CreateRGBSurface(1,640,480,32,0,0,0,0);
    SDL_Texture *texture = SDL_CreateTextureFromSurface(renderer, surface);
    SDL_FreeSurface(surface);
    SDL_EventState(SDL_SYSWMEVENT, SDL_ENABLE);
    if (input_type == file_type_c){
        root = read_source(input_file);
    }
    if (output_type == file_type_c){
        printf("HOW");
    }
    while (!quit) {
        SDL_PollEvent(&event);
        switch (event.type) {
            case SDL_WINDOWEVENT_CLOSE:
                event.type = SDL_QUIT;
                SDL_PushEvent(&event);
                break;
            case SDL_QUIT:
                quit = 1;
                break;
            case SDL_SYSWMEVENT:
                if (event.syswm.msg->msg.win.msg == WM_COMMAND){
                    char* temp;
                    switch (event.syswm.msg->msg.win.wParam){
                        case ID_EXIT:
                            quit = 1;
                            break;
                        case ID_OPEN_FILE:
                            ;
                            temp = file_open_dialog(windowRef);
                            if (temp == NULL) break;
                            input_file = temp;
                            input_type = ends_with(input_file);
                            root = read_source(input_file);
                            //printf("%s", input_file);
                            break;
                        case ID_SAVE_FLOW:
                            ;
                            temp = file_save_dialog(windowRef);
                            if (temp == NULL) break;
                            output_file = temp;
                            output_type = ends_with(output_file);
                            //printf("%s", output_file);
                            break;
                        case ID_LOAD_THEME:
                            break;
                        case ID_RESET_THEME:
                            theme = &default_theme;
                            //redarw call
                            break;
                        case ID_ZOOM_IN:
                            break;
                        case ID_ZOOM_OUT:
                            break;
                        case ID_ZOOM_RESET:
                            break;
                        default:
                            break;
                    }
                }
                break;
        }

        //SDL_SetRenderDrawColor(renderer, theme->main_.background.r, theme->main_.background.g, theme->main_.background.b, theme->main_.background.a);
        //SDL_RenderClear(renderer);
        SDL_RenderPresent(renderer);
    }
    SDL_DestroyTexture(texture);
    SDL_DestroyRenderer(renderer);
    SDL_DestroyWindow(window);
    SDL_Quit();
    //free(theme);
    //free(input_file);
    //free(output_file);
    free(root);
    return 0;
}
```

```

HWND GetHwnd(SDL_Window *window){
    SDL_SysWMinfo windowInfo;
    if(!SDL_GetWindowWMInfo(window,&windowInfo)) return NULL;
    return windowInfo.info.win.window;
}

void ActivateMenu(HWND windowRef)
{
    HMENU hMenuBar = CreateMenu();
    HMENU hFile = CreateMenu();
    HMENU hView = CreateMenu();

    AppendMenu(hMenuBar, MF_POPUP, (UINT_PTR)hFile, "File");
    AppendMenu(hMenuBar, MF_POPUP, (UINT_PTR)hView, "View");
    AppendMenu(hMenuBar, MF_STRING, ID_EXIT, "Exit");

    AppendMenu(hFile, MF_STRING, ID_OPEN_FILE, "Open File");
    AppendMenu(hFile, MF_STRING, ID_SAVE_FLOW, "Save flowchart");

    AppendMenu(hView, MF_STRING, ID_LOAD_THEME, "Load Theme");
    AppendMenu(hView, MF_STRING, ID_RESET_THEME, "Reset Theme");
    AppendMenu(hView, MF_STRING, ID_ZOOM_IN, "Zoom In");
    AppendMenu(hView, MF_STRING, ID_ZOOM_OUT, "Zoom Out");
    AppendMenu(hView, MF_STRING, ID_ZOOM_RESET, "Zoom Reset");

    SetMenu(windowRef, hMenuBar);
}

char* file_open_dialog(HWND windowRef)
{
    char *file_path = NULL;
    HRESULT hr = CoInitializeEx(NULL, COINIT_APARTMENTTHREADED |
                                      COINIT_DISABLE_OLE1DDE);
    COMDLG_FILTERSPEC filterspec = {L"Source Files", L"*.c;*.h"};
    if (SUCCEEDED(hr))
    {
        IFileOpenDialog *pFileOpen;

        // Create the FileOpenDialog object.
        hr = CoCreateInstance(&CLSID_FileOpenDialog, NULL, CLSCTX_ALL,
                              &IID_IFileOpenDialog, (void**)(&pFileOpen));
        if (SUCCEEDED(hr))
        {
            pFileOpen->lpVtbl->SetFileTypes(pFileOpen, 1, &filterspec);
            // Show the Open dialog box.
            hr = pFileOpen->lpVtbl->Show(pFileOpen, windowRef);

            // Get the file name from the dialog box.
            if (SUCCEEDED(hr))
            {
                IShellItem *pItem;
                hr = pFileOpen->lpVtbl->GetResult(pFileOpen, &pItem);
                if (SUCCEEDED(hr))
                {
                    PWSTR pszFilePath;
                    hr = pItem->lpVtbl->GetDisplayName(pItem,SIGDN_FILESYSPATH,&pszFilePath);

                    // Display the file name to the user.
                    if (SUCCEEDED(hr))
                    {
                        //MessageBoxW(NULL, pszFilePath, L"File Path", MB_OK);
                        file_path = (char *)malloc(lstrlenW(pszFilePath) + 1);
                        wcstombs(file_path, pszFilePath, lstrlenW(pszFilePath) + 1);
                        CoTaskMemFree(pszFilePath);
                    }
                    pItem->lpVtbl->Release((IShellItem *) &pItem);
                }
            }
            pFileOpen->lpVtbl->Release((IFileOpenDialog *) &pFileOpen);
        }
        CoUninitialize();
    }
    return file_path;
}

char* file_save_dialog(HWND windowRef)
{
    char *file_path = NULL;
    HRESULT hr = CoInitializeEx(NULL, COINIT_APARTMENTTHREADED |
                                      COINIT_DISABLE_OLE1DDE);
    COMDLG_FILTERSPEC filterspec[2] = {{L"png", L"*.png"}, {L"jpg", L"*.jpg"}}; //TODO: add md later
    if (SUCCEEDED(hr))
    {
        IFileSaveDialog *pFileSave;

        // Create the FileOpenDialog object.
        hr = CoCreateInstance(&CLSID_FileSaveDialog, NULL, CLSCTX_ALL,
                              &IID_IFileSaveDialog, (void**)(&pFileSave));
        if (SUCCEEDED(hr))
        {
            pFileSave->lpVtbl->SetFileTypes(pFileSave, 2, filterspec);
            pFileSave->lpVtbl->SetFileName(pFileSave, L"Flowchart.png");
            // Show the Open dialog box.
            hr = pFileSave->lpVtbl->Show(pFileSave, windowRef);
            // Get the file name from the dialog box.
            if (SUCCEEDED(hr))
            {
                IShellItem *pItem;
                unsigned int i;
                pFileSave->lpVtbl->GetFileTypeIndex(pFileSave, &i);
                hr = pFileSave->lpVtbl->GetResult(pFileSave, &pItem);
                if (SUCCEEDED(hr))
                {
                    PWSTR pszFilePath;
                    hr = pItem->lpVtbl->GetDisplayName(pItem,SIGDN_FILESYSPATH,&pszFilePath);

                    // Display the file name to the user.
                    if (SUCCEEDED(hr))
                    {
                        //MessageBoxW(NULL, pszFilePath, L"File Path", MB_OK);
                        file_path = (char *)malloc(lstrlenW(pszFilePath) + 5);
                        wcstombs(file_path, pszFilePath, lstrlenW(pszFilePath) + 1);
                        strcat(file_path, i == 1 ? ".png" : ".jpg");
                        CoTaskMemFree(pszFilePath);
                    }
                    pItem->lpVtbl->Release((IShellItem *) &pItem);
                }
            }
            pFileSave->lpVtbl->Release((IFileSaveDialog *) &pFileSave);
        }
        CoUninitialize();
    }
    return file_path;
}
```

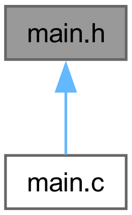
5.17 main.h File Reference

```
#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 )
```

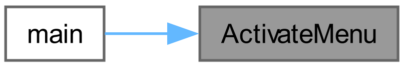
Creates a menu for the given window handle.

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

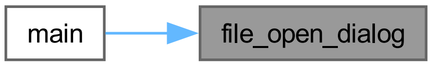
windowRef	

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

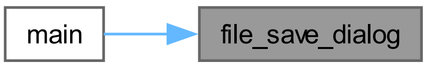
windowRef	

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

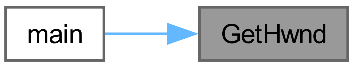
window	sdl window

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.

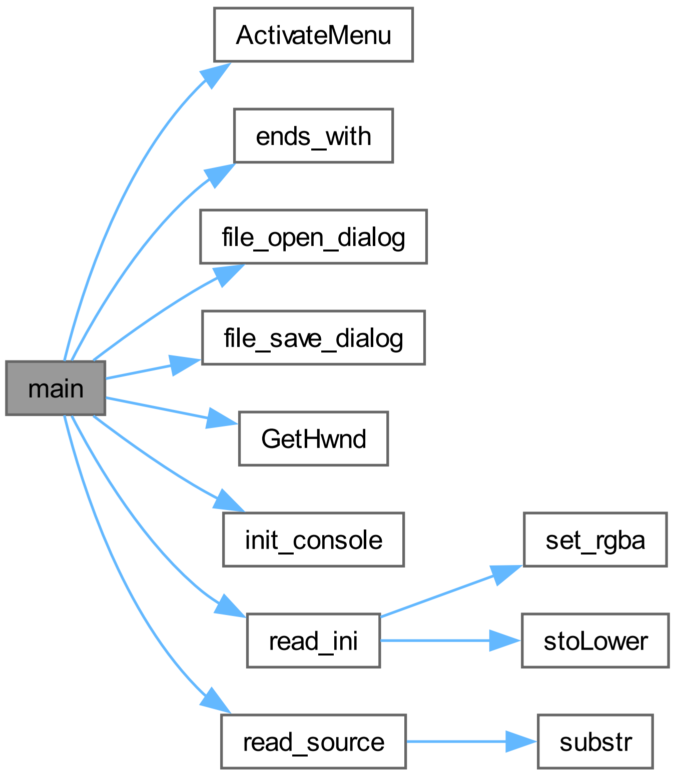
Parameters

argc	
argv	

Returns

Definition at line 3 of file main.c.

Here is the call graph for this function:



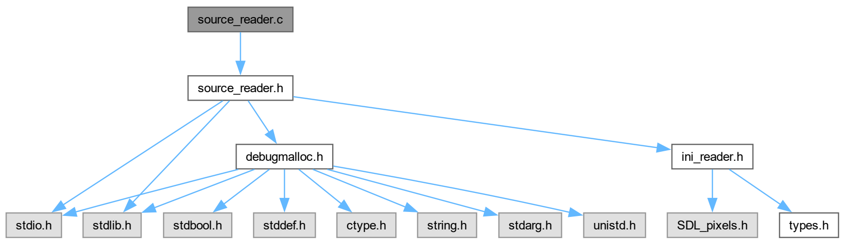
5.18 main.h

Go to the documentation of this file.

```c
//
// Created by sziha on 18/10/2023.
//

#ifndef NHF_MAIN_H
#define NHF_MAIN_H
#endif//NHF_MAIN_H

#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"
int main(int argc, char** argv);

#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

HWND GetHwnd(SDL_Window *window);
void ActivateMenu(HWND windowRef);
//code from
https://stackoverflow.com/questions/51250046/sdl2-win32-api-menubar-click-event-not-working
char* file_open_dialog(HWND windowRef);
char* file_save_dialog(HWND windowRef);

/*
 * Default theme for the viewport
 */
static theme_t default_theme = {
        .main_ = {
            .background = {.r = 255,.g = 255,.b = 255,.a = 255},
            .text = {.r = 0,.g = 0,.b = 0,.a = 255}},
        .functions = {
            .background = {.r = 255,.g = 255,.b = 255,.a = 255},
            .text = {.r = 0,.g = 0,.b = 0,.a = 255}},
        .structs = {
            .background = {.r = 255,.g = 255,.b = 255,.a = 255},
            .text = {.r = 0,.g = 0,.b = 0,.a = 255}},
        .variables = {
            .background = {.r = 255,.g = 255,.b = 255,.a = 255},
            .text = {.r = 0,.g = 0,.b = 0,.a = 255}},
        .conditionals = {
            .background = {.r = 255,.g = 255,.b = 255,.a = 255},
            .text = {.r = 0,.g = 0,.b = 0,.a = 255}},
        .loops = {
            .background = {.r = 255,.g = 255,.b = 255,.a = 255},
            .text = {.r = 0,.g = 0,.b = 0,.a = 255}}
};
```

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

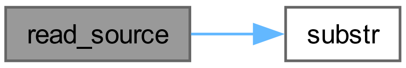
source_file	to read

Returns

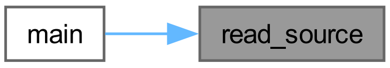
linked_list of all the lines

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

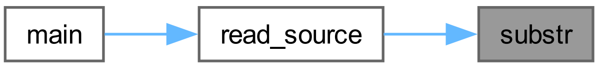
str	
start	
end	

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

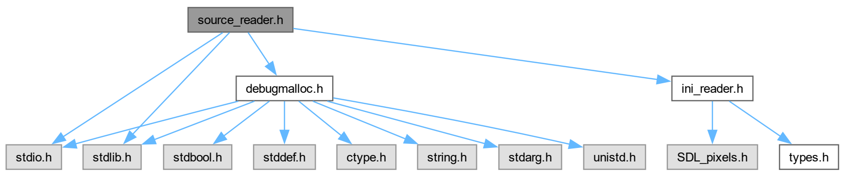
Go to the documentation of this file.

```c
//
// Created by sziha on 21/10/2023.
//

#include "source_reader.h"

node_t *read_source(char *filename){
    FILE *fp = fopen(filename, "r");
    if (fp == NULL) {
        fprintf(stderr, "Unable to open file %s\n", filename);
        return NULL;
    }
    node_t *first_node = (node_t *) malloc(sizeof(node_t));
    node_t *current_node = first_node;
    current_node->type = -1;
    char buffer[1000];
    int skip = 0;
    int c = 0;
    int c_bracket[2] = {0,0};
    for (int i = 0; c != EOF; i = (i == 999) ? 0 : i + 1) {
        c = getc(fp);
        if (skip == 1 && c != '\n') {
            continue;
        }
        if (c == '{')
        {
            c_bracket[1] = c_bracket[0];
            c_bracket[0]++;
        }
        else if (c == '}')
        {
            c_bracket[1] = c_bracket[0];
            c_bracket[0]--;
        }
        skip = 0;
        buffer[i] = (char) c;
        buffer[i + 1] = '\0';
        if (buffer[i] == '#' || buffer[i] == '\n' || (i > 1 && (buffer[i-1] == '/' && buffer[i] ==
'/'))) {
            i--;
            skip = 1;
            if (i > 1 && (buffer[i-1] == '/' && buffer[i] == '/')) i--;
            continue;
        }
        fprintf(stderr,"%s", buffer);
        switch (current_node->type) {
            case structs:
                if (current_node->name[0] == '\0') {
                    if (c_bracket[0] == 1)
                        i = 0;
                    if (c_bracket[0] == 0 && c_bracket[1] == 1) {
                        if (c != ';') continue;
                        char *name = substr(buffer, '}', ';');
                        if (name == NULL) continue;
                    }
                }
                break;
            default:
                if (strstr(buffer, "struct") != NULL) {
                    printf("test");
                    current_node->type = structs;
                    if (sscanf(buffer,"struct %s", current_node->name) != 1) { //struct name
                        if (sscanf(buffer, "struct %s{", current_node->name) != 1) current_node->name[0] = '\0'; //struct name{
                    }
                }
                break;
        }
    }
    fclose(fp);
    return first_node;
}

char* substr(char const *str, char start, char end) {
    char* s = strchr(str, start);
    char* e = strchr(str, end);
    if (s == NULL || e == NULL)
        return NULL;
    char* result = malloc((e - s) + 1);
    if (result == NULL)
        return NULL;
    strncpy(result, s, e-s);
    result[e-s+1] = '\0';
    return result;
}
```

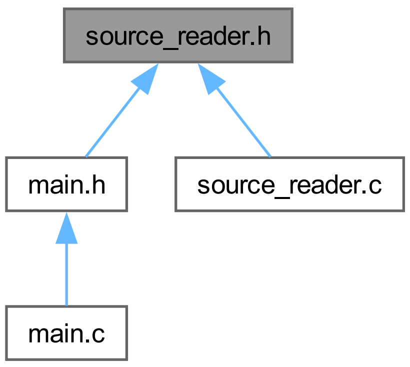
5.21 source_reader.h File Reference

```
#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

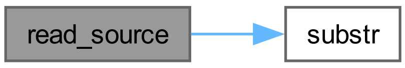
source_file	to read

Returns

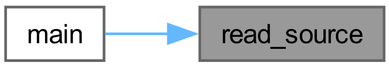
linked_list of all the lines

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

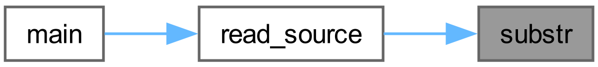
str	
start	
end	

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.

```
//
// Created by sziha on 21/10/2023.
//

#ifndef NHF_SOURCE_READER_H
#define NHF_SOURCE_READER_H
#include <stdio.h>
#include <stdlib.h>
#include "ini_reader.h"
#include "debugmalloc.h"
typedef struct {
    char *return_type;
    char **args;
} func_type_t;

typedef struct {
    char *args;
} struct_type_t;

typedef struct {
    char *value;
} variable_type_t;

typedef struct {
    char *condition;
} conditional_type_t;

typedef struct {
    char *condition;
} loop_type_t;
typedef struct node{
    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;
} node_t;
node_t *read_source(char *filename);
char* substr(char const *str, char start, char end);
#endif //NHF_SOURCE_READER_H
```

5.23 Specification.md File Reference

5.24 test.c File Reference

Data Structures

- struct test
- struct test_t

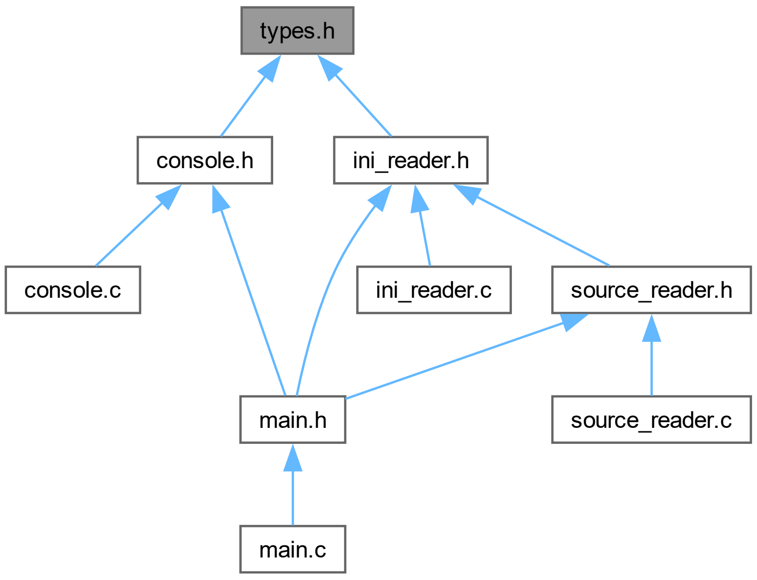
5.25 test.c

Go to the documentation of this file.

```
//
// Created by sziha on 25/10/2023.
//

struct test{
    int a;
};
typedef struct{
    int b;
} 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