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 box_t Struct Reference

type for boxes

`#include <graphics.h>`

Collaboration diagram for box_t:

Data Fields

- rect_t rect
- int type

  rectangle
- char * text

  type
- int radius

  text

4.1.1 Detailed Description

type for boxes

Definition at line 35 of file graphics.h.

4.1.2 Field Documentation

4.1.2.1 radius

```
int radius
```

text

Definition at line 39 of file graphics.h.

4.1.2.2 rect

```
rect_t rect
```

Definition at line 36 of file graphics.h.

4.1.2.3 text

```
char* text
```

type

Definition at line 38 of file graphics.h.

4.1.2.4 type

```
int type
```

rectangle

Definition at line 37 of file graphics.h.

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

- graphics.h

4.2 colour_t Struct Reference

colouring struct for theme

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

Data Fields

- SDL_Colour background
- SDL_Colour text

4.2.1 Detailed Description

colouring struct for theme

Definition at line 36 of file ini_reader.h.

4.2.2 Field Documentation

4.2.2.1 background

```
SDL_Colour background
```

Definition at line 37 of file ini_reader.h.

4.2.2.2 text

```
SDL_Colour text
```

Definition at line 38 of file ini_reader.h.

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

- ini_reader.h

4.3 func_type_t Struct Reference

types of nodes

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

Data Fields

- char * return_type
- char ** args

4.3.1 Detailed Description

types of nodes

Definition at line 33 of file source_reader.h.

4.3.2 Field Documentation

4.3.2.1 args

```
char** args
```

Definition at line 35 of file source_reader.h.

4.3.2.2 return_type

```
char* return_type
```

Definition at line 34 of file source_reader.h.

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

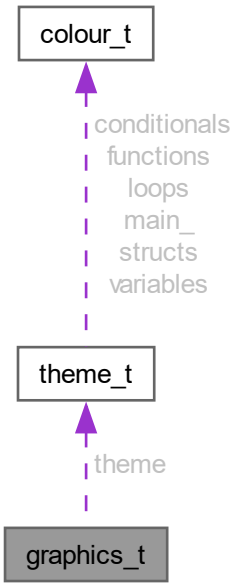
- source_reader.h

4.4 graphics_t Struct Reference

type for graphics

```
#include <graphics.h>
```

Collaboration diagram for graphics_t:



Data Fields

- SDL_Renderer * renderer
- theme_t * theme

  renderer
- SDL_PixelFormatEnum pixel_format

  theme
- TTF_Font * font

  pixel format
- double scale

  font

4.4.1 Detailed Description

type for graphics

Definition at line 18 of file graphics.h.

4.4.2 Field Documentation

4.4.2.1 font

```
TTF_Font* font
```

pixel format

Definition at line 22 of file graphics.h.

4.4.2.2 pixel_format

```
SDL_PixelFormatEnum pixel_format
```

theme

Definition at line 21 of file graphics.h.

4.4.2.3 renderer

```
SDL_Renderer* renderer
```

Definition at line 19 of file graphics.h.

4.4.2.4 scale

```
double scale
```

font

Definition at line 23 of file graphics.h.

4.4.2.5 theme

```
theme_t* theme
```

renderer

Definition at line 20 of file graphics.h.

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

- graphics.h

4.5 loop_cond_type_t Struct Reference

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

Data Fields

- loop_cond_e type
- char * condition
- bool finished

4.5.1 Detailed Description

Definition at line 47 of file source_reader.h.

4.5.2 Field Documentation

4.5.2.1 condition

```
char* condition
```

Definition at line 49 of file source_reader.h.

4.5.2.2 finished

```
bool finished
```

Definition at line 50 of file source_reader.h.

4.5.2.3 type

```
loop_cond_e type
```

Definition at line 48 of file source_reader.h.

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

- source_reader.h

4.6 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

4.6.1 Detailed Description

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

Definition at line 19 of file types.h.

4.6.2 Field Documentation

4.6.2.1 key

```
const char* key
```

Definition at line 20 of file types.h.

4.6.2.2 value

```
const void* value
```

Definition at line 21 of file types.h.

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

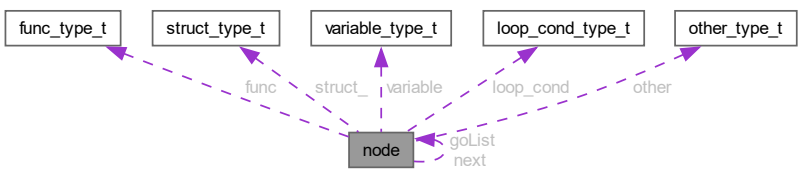
- types.h

4.7 node Struct Reference

linked list structure

`#include <source_reader.h>`

Collaboration diagram for node:



Data Fields

- int type
- char * name

  type of the node
- int indent

  name of the node
- int call_num

  indent of the node
- struct node ** goList

  number of time the node has been called
- union {
  func_type_t func
  struct_type_t struct_
  function
  variable_type_t variable
  struct
  loop_cond_type_t loop_cond
  variable
  other_type_t other
  loop or conditional statement
  };

  list of return to nodes
- struct node * next

4.7.1 Detailed Description

linked list structure

Definition at line 61 of file source_reader.h.

4.7.2 Field Documentation

4.7.2.1 [union]

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

list of return to nodes

4.7.2.2 call_num

```
int call_num
```

indent of the node

Definition at line 65 of file source_reader.h.

4.7.2.3 func

```
func_type_t func
```

Definition at line 68 of file source_reader.h.

4.7.2.4 goList

```
struct node** goList
```

number of time the node has been called

Definition at line 66 of file source_reader.h.

4.7.2.5 indent

```
int indent
```

name of the node

Definition at line 64 of file source_reader.h.

4.7.2.6 loop_cond

```
loop_cond_type_t loop_cond
```

variable

Definition at line 71 of file source_reader.h.

4.7.2.7 name

```
char* name
```

type of the node

Definition at line 63 of file source_reader.h.

4.7.2.8 next

```
struct node* next
```

Definition at line 74 of file source_reader.h.

4.7.2.9 other

```
other_type_t other
```

loop or conditional statement

Definition at line 72 of file source_reader.h.

4.7.2.10 struct_

```
struct_type_t struct_
```

function

Definition at line 69 of file source_reader.h.

4.7.2.11 type

```
int type
```

Definition at line 62 of file source_reader.h.

4.7.2.12 variable

```
variable_type_t variable
```

struct

Definition at line 70 of file source_reader.h.

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

- source_reader.h

4.8 other_type_t Struct Reference

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

Data Fields

- char * value

4.8.1 Detailed Description

Definition at line 53 of file source_reader.h.

4.8.2 Field Documentation

4.8.2.1 value

```
char* value
```

Definition at line 54 of file source_reader.h.

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

- source_reader.h

4.9 rect_t Struct Reference

type for rectangles

```
#include <graphics.h>
```

Data Fields

- int x
- int y
- int w
- int h

4.9.1 Detailed Description

type for rectangles

Definition at line 27 of file graphics.h.

4.9.2 Field Documentation

4.9.2.1 h

```
int h
```

Definition at line 31 of file graphics.h.

4.9.2.2 w

```
int w
```

Definition at line 30 of file graphics.h.

4.9.2.3 x

```
int x
```

Definition at line 28 of file graphics.h.

4.9.2.4 y

```
int y
```

Definition at line 29 of file graphics.h.

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

- graphics.h

4.10 struct_type_t Struct Reference

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

Data Fields

- char * args

4.10.1 Detailed Description

Definition at line 38 of file source_reader.h.

4.10.2 Field Documentation

4.10.2.1 args

```
char* args
```

Definition at line 39 of file source_reader.h.

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

- source_reader.h

4.11 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_

4.11.1 Detailed Description

struct for theme

Definition at line 44 of file ini_reader.h.

4.11.2 Field Documentation

4.11.2.1 conditionals

`colour_t conditionals`

Definition at line 48 of file ini_reader.h.

4.11.2.2 functions

```
colour_t functions
```

Definition at line 45 of file ini_reader.h.

4.11.2.3 loops

```
colour_t loops
```

Definition at line 49 of file ini_reader.h.

4.11.2.4 main_

```
colour_t main_
```

Definition at line 50 of file ini_reader.h.

4.11.2.5 structs

```
colour_t structs
```

Definition at line 46 of file ini_reader.h.

4.11.2.6 variables

```
colour_t variables
```

Definition at line 47 of file ini_reader.h.

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

- ini_reader.h

4.12 variable_type_t Struct Reference

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

Data Fields

- char * type
- char * value

4.12.1 Detailed Description

Definition at line 42 of file source_reader.h.

4.12.2 Field Documentation

4.12.2.1 type

```
char* type
```

Definition at line 43 of file source_reader.h.

4.12.2.2 value

```
char* value
```

Definition at line 44 of file source_reader.h.

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

- source_reader.h

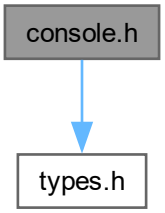
Chapter 5

File Documentation

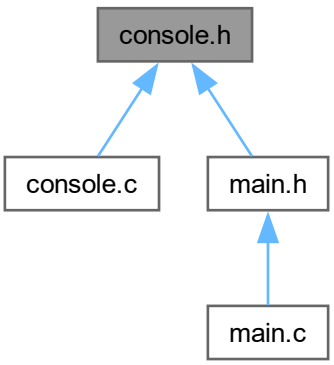
5.1 console.h File Reference

```
#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.

5.1.1 Function Documentation

5.1.1.1 ends_with()

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

This function is used to get the file type.

Parameters

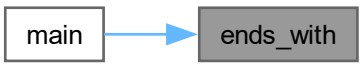
file	file path

Returns

file_type

Definition at line 39 of file console.c.

Here is the caller graph for this function:



5.1.1.2 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

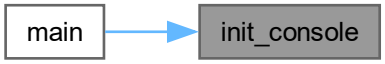
argc	argument_cont from main
argv	arguments list from main
theme_file	theme file path
output_file	output file path

Returns

0 if success, -1 if error

Definition at line 8 of file console.c.

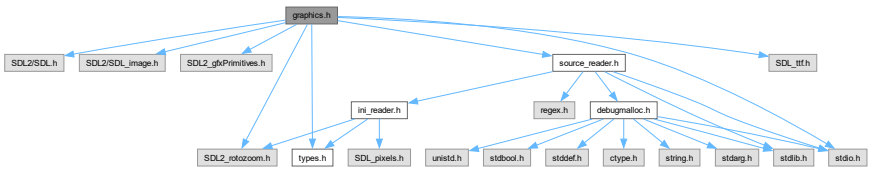
Here is the caller graph for this function:



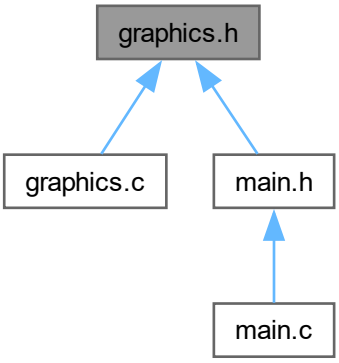
5.2 graphics.h File Reference

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <SDL2_gfxPrimitives.h>
#include <SDL2_rotozoom.h>
#include <SDL_ttf.h>
#include <stdio.h>
#include "types.h"
#include "source_reader.h"
```

Include dependency graph for graphics.h:



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



Data Structures

- struct graphics_t
  type for graphics
- struct rect_t
  type for rectangles
- struct box_t
  type for boxes

Functions

- int redraw (graphics_t *graphics, box_t *boxes, int num_boxes)
  Redraws the screen.
- int initGraphics (graphics_t *graphics, SDL_Window *window)
  intializes the graphics
- int initBoxes (graphics_t *graphics, node_t *start_node, box_t *boxes)
  intializes the boxes
- int drawArrow (box_t start, box_t end, graphics_t *graphics)
  draw an arrow
- int drawBox (box_t box, graphics_t *graphics)
  draw a box
- int drawRect (rect_t rect, graphics_t *graphics, SDL_Colour colour)
  draw a rectangle
- int nodeToBox (node_t *node, box_t *box)
- int saveToFile (graphics_t *graphics, const char *filename, file_type_e file_type)
  save the screen to a file
- char * cat (char *a, char *b)
  appends two strings with realloc
- void freeBoxes (box_t *boxes, int box_count)
  free the boxes

5.2.1 Function Documentation

5.2.1.1 cat()

```
char * cat (
            char * a,
            char * b )
```

appends two strings with realloc

Parameters

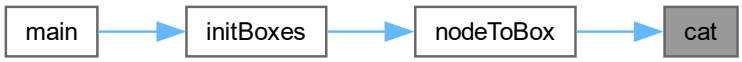
a	string to append to
b	string to append

Returns

a

Definition at line 225 of file graphics.c.

Here is the caller graph for this function:



5.2.1.2 drawArrow()

```
int drawArrow (
            box_t start,
            box_t end,
            graphics_t * graphics )
```

draw an arrow

Parameters

start	start box
end	end box
graphics	collection of the important variables for the graphics

Returns

0 if success, -1 if error

Definition at line 189 of file graphics.c.

5.2.1.3 drawBox()

```
int drawBox (
            box_t box,
            graphics_t * graphics )
```

draw a box

Parameters

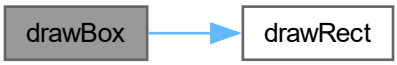
box	box to draw
graphics	collection of the important variables for the graphics

Returns

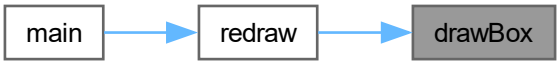
0 if success, -1 if error

Definition at line 161 of file graphics.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.1.4 drawRect()

```
int drawRect (
            rect_t rect,
            graphics_t * graphics,
            SDL_Colour colour )
```

draw a rectangle

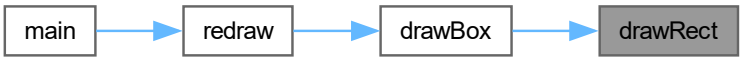
Parameters

rect	rectangle to draw
graphics	collection of the important variables for the graphics
colour	colour of the rectangle

Returns

Definition at line 151 of file graphics.c.

Here is the caller graph for this function:



5.2.1.5 freeBoxes()

```
void freeBoxes (
            box_t * boxes,
            int box_count )
```

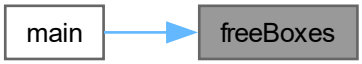
free the boxes

Parameters

boxes	list of boxes
box_count	number of boxes

Definition at line 231 of file graphics.c.

Here is the caller graph for this function:



5.2.1.6 initBoxes()

```
int initBoxes (
            graphics_t * graphics,
            node_t * start_node,
            box_t * boxes )
```

intializes the boxes

Parameters

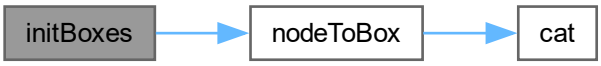
graphics	collection of the important variables for the graphics
start_node	the start node of the linked list
boxes	the list of boxes to append to

Returns

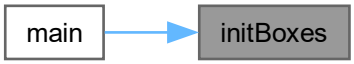
0 if success, -1 if error

Definition at line 38 of file graphics.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.1.7 initGraphics()

```
int initGraphics (
            graphics_t * graphics,
            SDL_Window * window )
```

intializes the graphics

Parameters

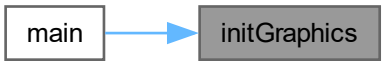
graphics	collection of the important variables for the graphics
window	the window

Returns

0 if success, -1 if error

Definition at line 21 of file graphics.c.

Here is the caller graph for this function:



5.2.1.8 nodeToBox()

```
int nodeToBox (
            node_t * node,
            box_t * box )
```

Parameters

node	start node
boc	box to fill

Returns

-1 if error, 0 if end_of nodes, 1 new box needed

Definition at line 67 of file graphics.c.

Here is the call graph for this function:

Here is the caller graph for this function:

5.2.1.9 redraw()

```
int redraw (
            graphics_t * graphics,
            box_t * boxes,
            int num_boxes )
```

Redraws the screen.

Parameters

graphics	collection of the important variables for the graphics
boxes	the boxes to draw
num_boxes	the number of boxes

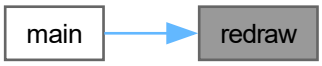
Returns

0 if success, -1 if error

Definition at line 7 of file graphics.c.

Here is the call graph for this function:

Here is the caller graph for this function:



5.2.1.10 saveToFile()

```
int saveToFile (
            graphics_t * graphics,
            const char * filename,
            file_type_e file_type )
```

save the screen to a file

Parameters

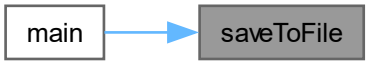
graphics	collection of the important variables for the graphics
filename	name of the file
file_type	the type of the file jpg or png

Returns

0 if success, -1 if error

Definition at line 206 of file graphics.c.

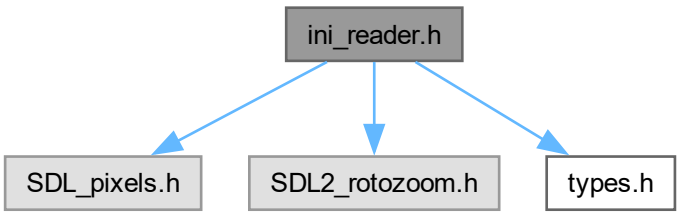
Here is the caller graph for this function:



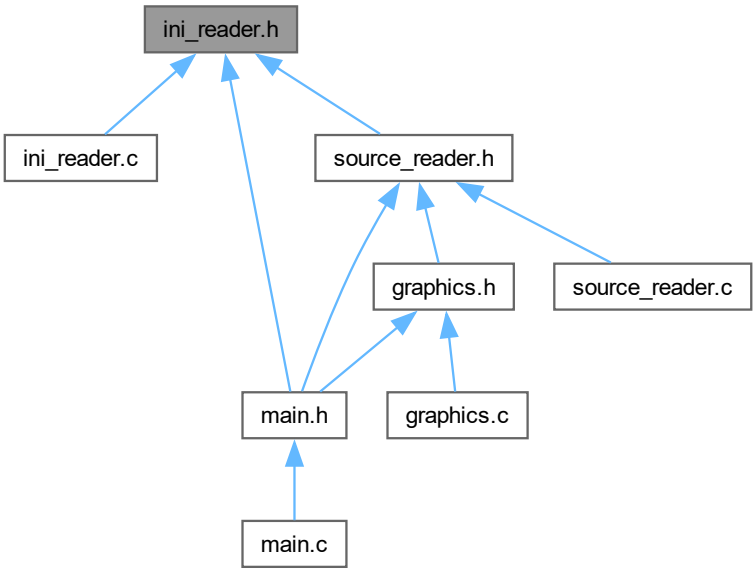
5.3 ini_reader.h File Reference

```
#include <SDL_pixels.h>
#include <SDL2_rotozoom.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 type
- enum sub_context_e { background , text }

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

Functions

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

  reads the theme ini file for custom themes
- void set_rgba (char *hex, SDL_Colour *colour)

  Sets the rgba of an SDL_Colour.

5.3.1 Enumeration Type Documentation

5.3.1.1 context_e

```
enum context_e
```

enum for ini file type

Enumerator

function	
structs	
variable	
conditional	
loop	
main_	

Definition at line 15 of file ini_reader.h.

5.3.1.2 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 type so it doesnt matter)

Enumerator

background	
text	

Definition at line 28 of file ini_reader.h.

5.3.2 Function Documentation

5.3.2.1 read_ini()

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

reads the theme ini file for custom themes

Parameters

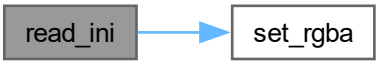
filename	name of the ini file (including the .ini)
theme	pointer to the theme variable

Returns

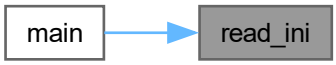
0 if ok, 1 if error

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:



5.3.2.2 set_rgba()

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

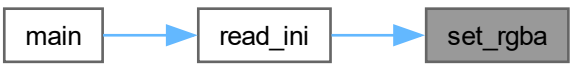
Sets the rgba of an SDL_Colour.

Parameters

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

Definition at line 81 of file ini_reader.c.

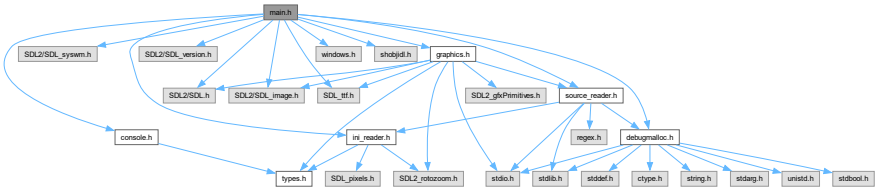
Here is the caller graph for this function:



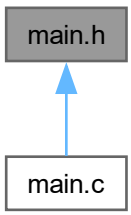
5.4 main.h File Reference

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_syswm.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_version.h>
#include <SDL_ttf.h>
#include "console.h"
#include "ini_reader.h"
#include <windows.h>
#include <shobjidl.h>
#include "source_reader.h"
#include "debugmalloc.h"
#include "graphics.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
- #define NHF_MAIN_H

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, const wchar_t *name, const wchar_t *file_spec)

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

  Creates a file save dialog for saving image files.

5.4.1 Macro Definition Documentation

5.4.1.1 ID_EXIT

```
#define ID_EXIT 8
```

Definition at line 27 of file main.h.

5.4.1.2 ID_LOAD_THEME

```
#define ID_LOAD_THEME 3
```

Definition at line 22 of file main.h.

5.4.1.3 ID_OPEN_FILE

```
#define ID_OPEN_FILE 1
```

Definition at line 20 of file main.h.

5.4.1.4 ID_RESET_THEME

```
#define ID_RESET_THEME 4
```

Definition at line 23 of file main.h.

5.4.1.5 ID_SAVE_FLOW

```
#define ID_SAVE_FLOW 2
```

Definition at line 21 of file main.h.

5.4.1.6 ID_ZOOM_IN

```
#define ID_ZOOM_IN 5
```

Definition at line 24 of file main.h.

5.4.1.7 ID_ZOOM_OUT

```
#define ID_ZOOM_OUT 6
```

Definition at line 25 of file main.h.

5.4.1.8 ID_ZOOM_RESET

```
#define ID_ZOOM_RESET 7
```

Definition at line 26 of file main.h.

5.4.1.9 NHF_MAIN_H

```
#define NHF_MAIN_H
```

Definition at line 88 of file main.h.

5.4.2 Function Documentation

5.4.2.1 ActivateMenu()

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

Creates a menu for the given window handle.

Parameters

windowRef	win32 window handle

Definition at line 171 of file main.c.

Here is the caller graph for this function:

5.4.2.2 file_open_dialog()

```
char * file_open_dialog (
            HWND windowRef,
            const wchar_t * name,
            const wchar_t * file_spec )
```

Creates a file open dialog for opening source files.

Parameters

windowRef	win32 window handle
name	filter name
file_spec	filter spec

Returns

file to open

Definition at line 193 of file main.c.

Here is the caller graph for this function:

5.4.2.3 file_save_dialog()

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

Creates a file save dialog for saving image files.

Parameters

windowRef	win32 window handle

Returns

file to save

Definition at line 240 of file main.c.

Here is the caller graph for this function:

5.4.2.4 GetHwnd()

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

Gets win32 window handle.

Parameters

window	sdl window

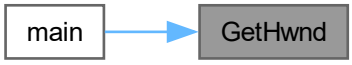
Returns

win32 window handle

NEEDED FOR SOME REASON

Definition at line 155 of file main.c.

Here is the caller graph for this function:



5.4.2.5 main()

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

Obvious.

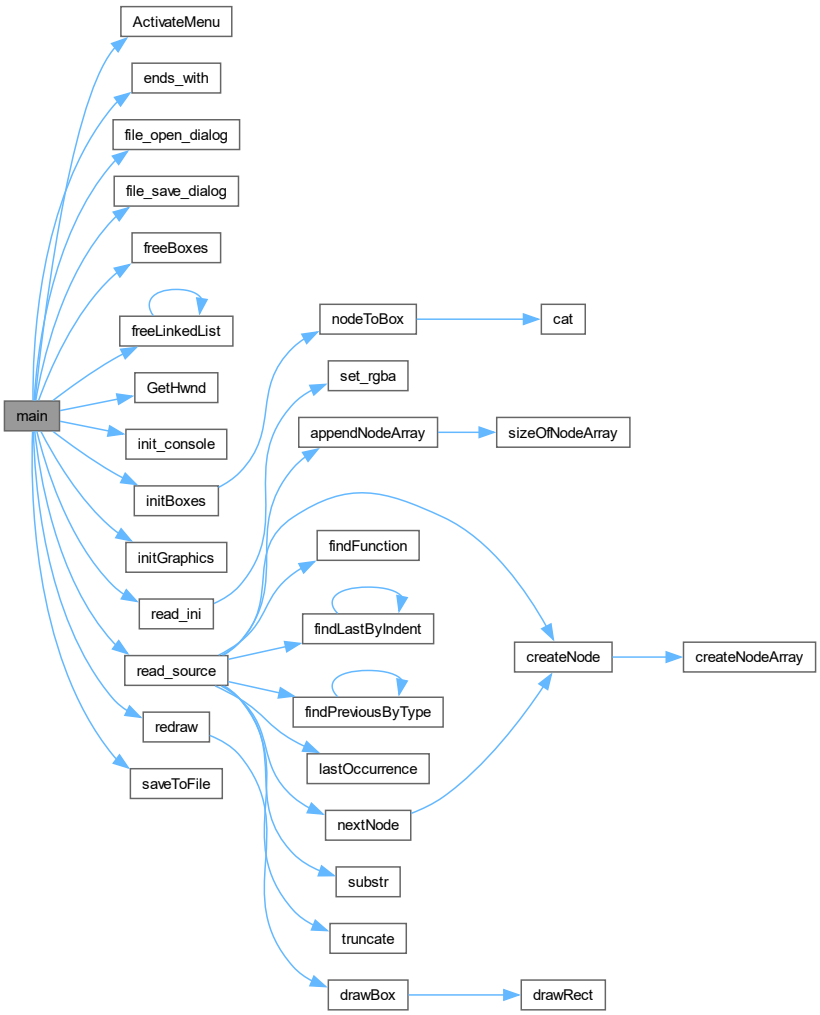
Parameters

argc	
argv	

Returns

Definition at line 3 of file main.c.

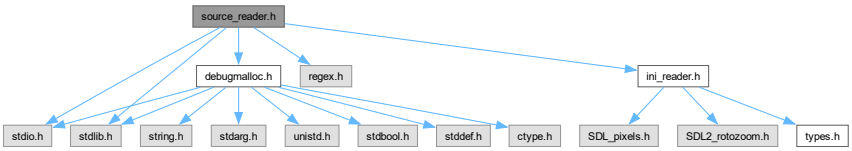
Here is the call graph for this function:



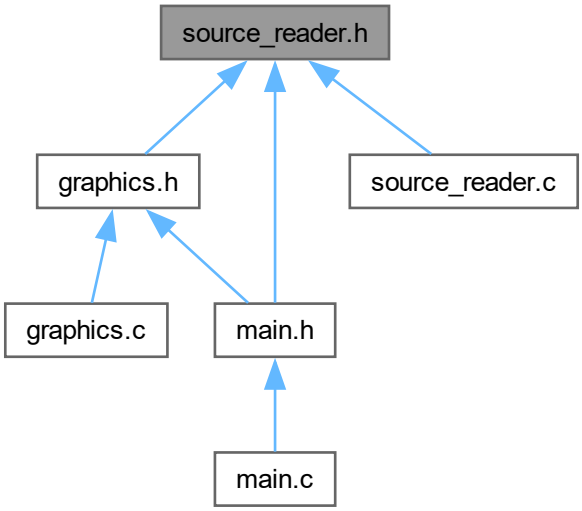
5.5 source_reader.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <regex.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

  types of nodes
- struct struct_type_t
- struct variable_type_t
- struct loop_cond_type_t
- struct other_type_t
- struct node

  linked list structure

Typedefs

- typedef struct node node_t

  linked list structure

Enumerations

- enum type_e { other = main_ + 1 , func_call = other + 1 }

  enum extension for node types
- enum loop_cond_e {
  if_ , else_ , else_if , switch_ ,
  case_ , while_ , for_ , do_while }

  extending type enum

Functions

- node_t * read_source (char *filename)

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

  Returns a substirng from the first appearance of start until the first appearance of end.
- char * truncate (char *str, int len)

  Truncates a string from spaces and endlines
  if len < 0, string length doesnt change
  if len > 0, truncates to len.
- int lastOccurrence (char *str, char c)

  find the last occurence of a char in a string
- node_t * createNode (int type)

  Creates a new node.
- node_t * nextNode (node_t *node)

  advances to the next node
- node_t * findFunction (node_t *node, node_t *functions[ ])

  finds a function by name in the fucntions list
- node_t * findByName (node_t *start, char *name, int indent)

  finds a node with a given name starting from a specified node.
- node_t * findByType (node_t *start, int type, int indent)

  finds a node with a specified type and indentation level starting from a specified node.
- node_t * findLastByIndent (node_t *start, int indent)

  finds the last node with a specified indentation level starting from a specified node.
- node_t * findPreviousByType (node_t *start, node_t *current, int type, int indent)

  finds the previous node of a specified type and indentation level from a given starting node up to the current node.
- int sizeOfNodeArray (node_t **arr)

  calculates the number of nodes in an array of node pointers.
- node_t ** createNodeArray (int size)

  Creates an array of node pointers with a specified size.
- node_t ** appendNodeArray (node_t **arr, node_t *node)

  Appends a node to the end of an array of node pointers.
- void freeLinkedList (node_t *start)

  Frees all the nodes in a linked list starting from the given node.

5.5.1 Typedef Documentation

5.5.1.1 node_t

```
typedef struct node node_t
```

linked list structure

5.5.2 Enumeration Type Documentation

5.5.2.1 loop_cond_e

```
enum loop_cond_e
```

extending type enum

enum for loop conditions

Enumerator

if_	
else_	if
else_if	else
switch_	else if
case_	switch
while_	case
for_	while
do_while	for

Definition at line 20 of file source_reader.h.

5.5.2.2 type_e

```
enum type_e
```

enum extension for node types

Enumerator

other	
func_call	

Definition at line 14 of file source_reader.h.

5.5.3 Function Documentation

5.5.3.1 appendNodeArray()

```
node_t ** appendNodeArray (
            node_t ** arr,
            node_t * node )
```

Appends a node to the end of an array of node pointers.

Parameters

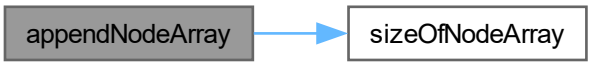
arr	The array of node pointers to which the node will be appended.
node	The node pointer that will be appended to the array.

Returns

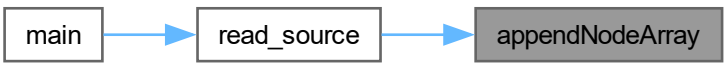
A pointer to the array of node pointers with the new node appended, or NULL if the operation fails.

Definition at line 677 of file source_reader.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.2 createNode()

```
node_t * createNode (
            int type )
```

Creates a new node.

Parameters

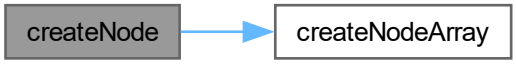
type	type of the node

Returns

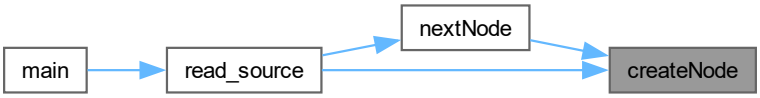
new node

Definition at line 584 of file source_reader.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.3 createNodeArray()

```
node_t ** createNodeArray (
            int size )
```

Creates an array of node pointers with a specified size.

Parameters

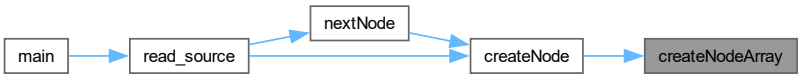
size	The number of node pointers to allocate in the array.

Returns

A pointer to the newly allocated array of node pointers, or NULL if allocation fails.

Definition at line 669 of file source_reader.c.

Here is the caller graph for this function:



5.5.3.4 findByName()

```
node_t * findByName (
            node_t * start,
            char * name,
            int indent )
```

finds a node with a given name starting from a specified node.

Parameters

start	The starting node for the search.
name	The name of the node to find.
indent	The indentation level to consider for the search.

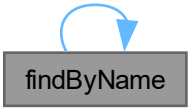
Returns

A pointer to the found node, or NULL if no node with the given name is found.

Definition at line 613 of file source_reader.c.

Here is the call graph for this function:

Here is the caller graph for this function:



5.5.3.5 findByType()

```
node_t * findByType (
            node_t * start,
            int type,
            int indent )
```

finds a node with a specified type and indentation level starting from a specified node.

Parameters

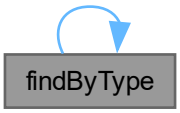
start	The starting node for the search.
type	The type of the node to find.
indent	The indentation level to consider for the search.

Returns

A pointer to the found node with the specified type and indentation level, or NULL if no such node is found.

Definition at line 620 of file source_reader.c.

Here is the call graph for this function:



Here is the caller graph for this function:

5.5.3.6 findFunction()

```
node_t * findFunction (
            node_t * node,
            node_t * functions[] )
```

finds a function by name in the fucntions list

Parameters

node	node to get name from
functions	functions list

Returns

function node

Definition at line 604 of file source_reader.c.

Here is the caller graph for this function:

5.5.3.7 findLastByIndent()

```
node_t * findLastByIndent (
            node_t * start,
            int indent )
```

finds the last node with a specified indentation level starting from a specified node.

Parameters

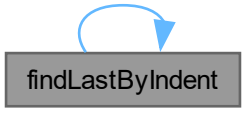
start	The starting node for the search.
indent	The indentation level to consider for the search.

Returns

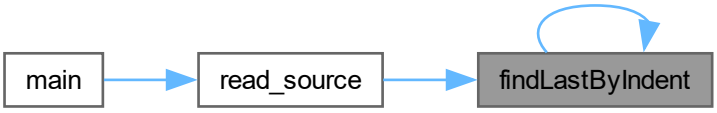
A pointer to the last node with the specified indentation level, or NULL if no such node is found.

Definition at line 627 of file source_reader.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.8 findPreviousByType()

```
node_t * findPreviousByType (
            node_t * start,
            node_t * current,
            int type,
            int indent )
```

finds the previous node of a specified type and indentation level from a given starting node up to the current node.

Parameters

start	The starting node for the search.
current	The current node, acting as the endpoint for the search.
type	The type of the node to find.
indent	The indentation level to consider for the search.

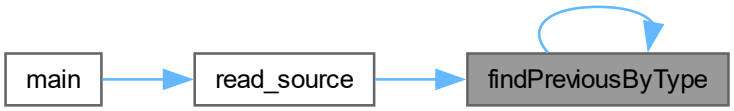
Returns

A pointer to the previous node with the specified type and indentation level, or NULL if no such node is found.

Definition at line 644 of file source_reader.c.

Here is the call graph for this function:

Here is the caller graph for this function:



5.5.3.9 freeLinkedList()

```
void freeLinkedList (
            node_t * start )
```

Frees all the nodes in a linked list starting from the given node.

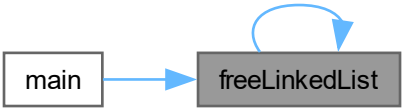
Parameters

start	The starting node of the linked list to be freed.

Definition at line 685 of file source_reader.c.

Here is the call graph for this function:

Here is the caller graph for this function:



5.5.3.10 lastOccurrence()

```
int lastOccurrence (
            char * str,
            char c )
```

find the last occurence of a char in a string

Parameters

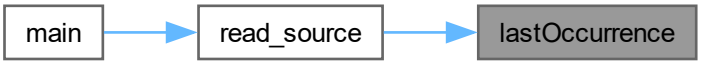
str	string to search
c	char to search for

Returns

-1 if not found, else the index

Definition at line 576 of file source_reader.c.

Here is the caller graph for this function:



5.5.3.11 nextNode()

```
node_t * nextNode (
            node_t * node )
```

advances to the next node

Parameters

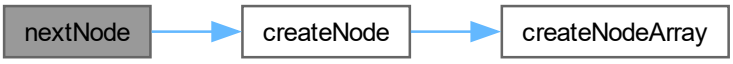
node	current node

Returns

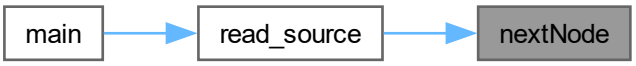
next node

Definition at line 597 of file source_reader.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.12 read_source()

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

NOT FULLY IMPLEMENTED YET.

Parameters

source_file	to read

Returns

linked_list of all the lines

Structures

Function definitions

Variables

Function calls

Conditionals

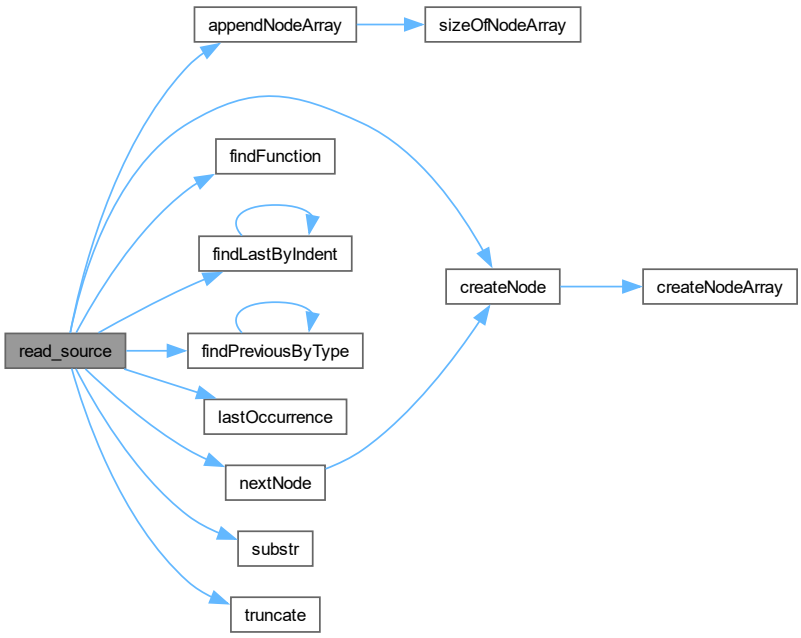
Loops

Others

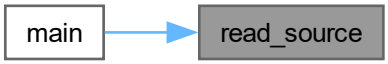
args

Definition at line 8 of file source_reader.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.13 sizeOfNodeArray()

```
int sizeOfNodeArray (
            node_t ** arr )
```

calculates the number of nodes in an array of node pointers.

Parameters

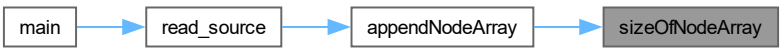
arr	The array of node pointers.

Returns

The number of nodes in the array.

Definition at line 661 of file source_reader.c.

Here is the caller graph for this function:



5.5.3.14 substr()

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

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

Parameters

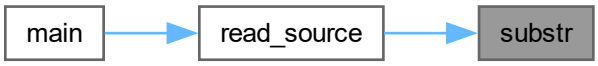
str	string to get substring from
start	starting char
end	ending char

Returns

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

Definition at line 540 of file source_reader.c.

Here is the caller graph for this function:



5.5.3.15 truncate()

```
char * truncate (
            char * str,
            int len )
```

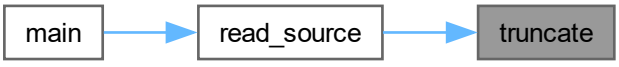
Truncates a string from spaces and endlines
if len $<$ 0, string length doesnt change
if len $>$ 0, truncates to len.

Parameters

str	string to get truncated
len	truncation length

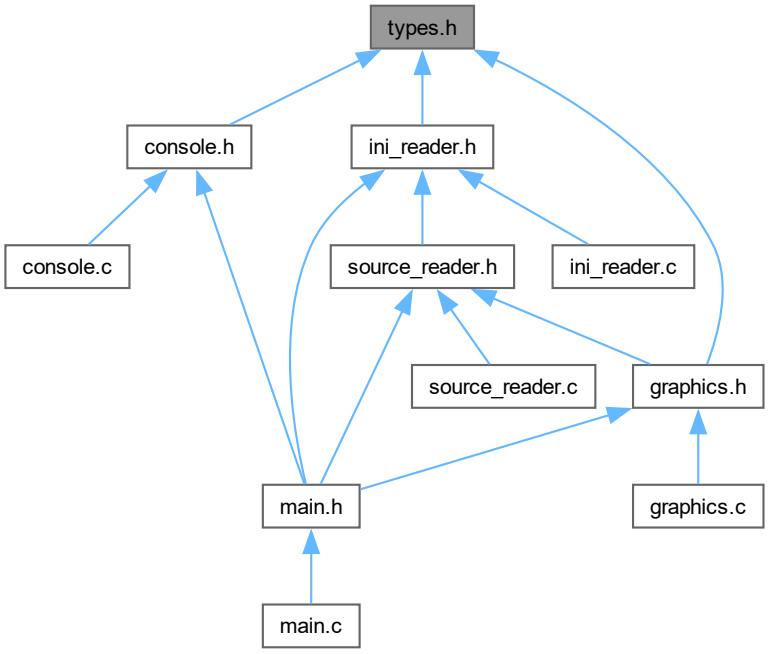
Definition at line 554 of file source_reader.c.

Here is the caller graph for this function:



5.6 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_jpg , file_type_png }

  Input file enum.

5.6.1 Macro Definition Documentation

5.6.1.1 DEFAULT_FILE_TYPE

```
#define DEFAULT_FILE_TYPE file_type_png
```

Definition at line 24 of file types.h.

5.6.2 Enumeration Type Documentation

5.6.2.1 file_type_e

```
enum file_type_e
```

Input file enum.

Enumerator

file_type_jpg	
file_type_png	jpg file

Definition at line 10 of file types.h.

Index