

Source to Flow

0.2

Generated by Doxygen 1.9.8

1 Specification	1
1.1 Source to Flow specifikáció	1
1.1.1 Parancssori irányítás	1
1.1.2 Grafikus Irányítás	1
1.1.3 Témák	2
1.1.4 File mentés	2
1.1.5 Kilépés	2
2 Data Structure Index	3
2.1 Data Structures	3
3 File Index	5
3.1 File List	5
4 Data Structure Documentation	7
4.1 box_t Struct Reference	7
4.1.1 Detailed Description	8
4.1.2 Field Documentation	8
4.1.2.1 radius	8
4.1.2.2 rect	8
4.1.2.3 text	8
4.1.2.4 type	8
4.2 colour_t Struct Reference	8
4.2.1 Detailed Description	9
4.2.2 Field Documentation	9
4.2.2.1 background	9
4.2.2.2 text	9
4.3 func_type_t Struct Reference	9
4.3.1 Detailed Description	9
4.3.2 Field Documentation	10
4.3.2.1 args	10
4.3.2.2 return_type	10
4.4 graphics_t Struct Reference	10
4.4.1 Detailed Description	11
4.4.2 Field Documentation	11
4.4.2.1 font	11
4.4.2.2 pixel_format	11
4.4.2.3 renderer	11
4.4.2.4 scale	11
4.4.2.5 theme	12
4.5 loop_cond_type_t Struct Reference	12
4.5.1 Detailed Description	12
4.5.2 Field Documentation	12

4.5.2.1 condition	12
4.5.2.2 finished	12
4.5.2.3 type	12
4.6 mapping_t Struct Reference	13
4.6.1 Detailed Description	13
4.6.2 Field Documentation	13
4.6.2.1 key	13
4.6.2.2 value	13
4.7 node Struct Reference	13
4.7.1 Detailed Description	14
4.7.2 Field Documentation	14
4.7.2.1 [union]	14
4.7.2.2 call_num	14
4.7.2.3 func	15
4.7.2.4 goList	15
4.7.2.5 indent	15
4.7.2.6 loop_cond	15
4.7.2.7 name	15
4.7.2.8 next	15
4.7.2.9 other	16
4.7.2.10 struct_	16
4.7.2.11 type	16
4.7.2.12 variable	16
4.8 other_type_t Struct Reference	16
4.8.1 Detailed Description	16
4.8.2 Field Documentation	17
4.8.2.1 value	17
4.9 rect_t Struct Reference	17
4.9.1 Detailed Description	17
4.9.2 Field Documentation	17
4.9.2.1 h	17
4.9.2.2 w	17
4.9.2.3 x	18
4.9.2.4 y	18
4.10 struct_type_t Struct Reference	18
4.10.1 Detailed Description	18
4.10.2 Field Documentation	18
4.10.2.1 args	18
4.11 theme_t Struct Reference	19
4.11.1 Detailed Description	19
4.11.2 Field Documentation	19
4.11.2.1 conditionals	19

4.11.2.2 functions	20
4.11.2.3 loops	20
4.11.2.4 main_	20
4.11.2.5 structs	20
4.11.2.6 variables	20
4.12 variable_type_t Struct Reference	20
4.12.1 Detailed Description	21
4.12.2 Field Documentation	21
4.12.2.1 type	21
4.12.2.2 value	21
5 File Documentation	23
5.1 console.h File Reference	23
5.1.1 Function Documentation	24
5.1.1.1 ends_with()	24
5.1.1.2 init_console()	24
5.2 graphics.h File Reference	25
5.2.1 Function Documentation	27
5.2.1.1 cat()	27
5.2.1.2 drawArrow()	27
5.2.1.3 drawBox()	28
5.2.1.4 drawRect()	29
5.2.1.5 freeBoxes()	29
5.2.1.6 initBoxes()	30
5.2.1.7 initGraphics()	31
5.2.1.8 nodeToBox()	31
5.2.1.9 redraw()	32
5.2.1.10 saveToFile()	33
5.3 ini_reader.h File Reference	34
5.3.1 Enumeration Type Documentation	36
5.3.1.1 context_e	36
5.3.1.2 sub_context_e	37
5.3.2 Function Documentation	37
5.3.2.1 read_ini()	37
5.3.2.2 set_rgba()	38
5.4 main.h File Reference	39
5.4.1 Macro Definition Documentation	40
5.4.1.1 ID_EXIT	40
5.4.1.2 ID_LOAD_THEME	40
5.4.1.3 ID_OPEN_FILE	40
5.4.1.4 ID_RESET_THEME	40
5.4.1.5 ID_SAVE_FLOW	40

5.4.1.6 ID_ZOOM_IN	41
5.4.1.7 ID_ZOOM_OUT	41
5.4.1.8 ID_ZOOM_RESET	41
5.4.1.9 NHF_MAIN_H	41
5.4.2 Function Documentation	41
5.4.2.1 ActivateMenu()	41
5.4.2.2 file_open_dialog()	42
5.4.2.3 file_save_dialog()	42
5.4.2.4 GetHwnd()	43
5.4.2.5 main()	44
5.5 source_reader.h File Reference	45
5.5.1 Typedef Documentation	47
5.5.1.1 node_t	47
5.5.2 Enumeration Type Documentation	47
5.5.2.1 loop_cond_e	47
5.5.2.2 type_e	47
5.5.3 Function Documentation	48
5.5.3.1 appendNodeArray()	48
5.5.3.2 createNode()	48
5.5.3.3 createNodeArray()	49
5.5.3.4 findByName()	50
5.5.3.5 findByType()	51
5.5.3.6 findFunction()	52
5.5.3.7 findLastByIndent()	52
5.5.3.8 findPreviousByType()	53
5.5.3.9 freeLinkedList()	54
5.5.3.10 lastOccurrence()	55
5.5.3.11 nextNode()	56
5.5.3.12 read_source()	57
5.5.3.13 sizeOfNodeArray()	58
5.5.3.14 substr()	58
5.5.3.15 truncate()	59
5.6 types.h File Reference	60
5.6.1 Macro Definition Documentation	60
5.6.1.1 DEFAULT_FILE_TYPE	60
5.6.2 Enumeration Type Documentation	61
5.6.2.1 file_type_e	61

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 fileből 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 kimentti 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 rákérdezzünk 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:

box_t	Type for boxes	7
colour_t	Colouring struct for theme	8
func_type_t	Types of nodes	9
graphics_t	Type for graphics	10
loop_cond_type_t	12
mapping_t	Hash-map like struct for mapping strings to anything (not very safe)	13
node	Linked list structure	13
other_type_t	16
rect_t	Type for rectangles	17
struct_type_t	18
theme_t	Struct for theme	19
variable_type_t	20

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

console.h	23
graphics.h	25
ini_reader.h	34
main.h	39
source_reader.h	45
types.h	60

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:

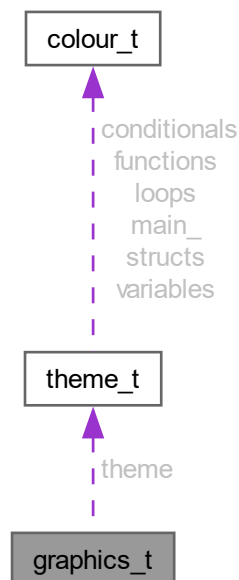
- [source_reader.h](#)

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:

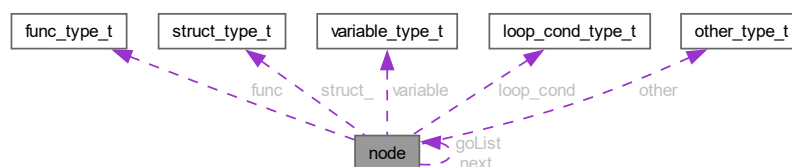
- [types.h](#)

4.7 node Struct Reference

linked list structure

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

Collaboration diagram for node:



Data Fields

- int `type`
type of the node
- char * `name`
name of the node
- int `indent`
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](#)

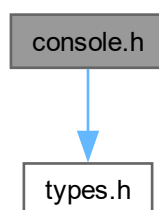
Chapter 5

File Documentation

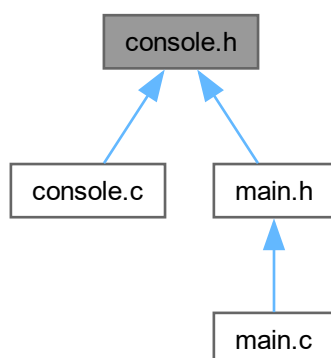
5.1 console.h File Reference

```
#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

<i>file</i>	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

<i>argc</i>	argument_cont from main
<i>argv</i>	arguments list from main
<i>theme_file</i>	theme file path
<i>output_file</i>	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:



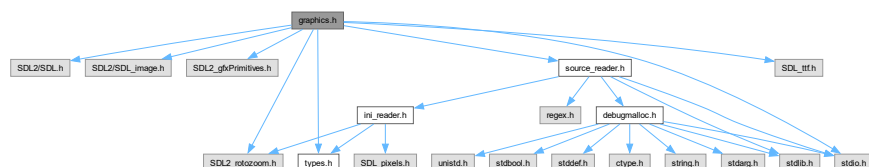
5.2 graphics.h File Reference

```

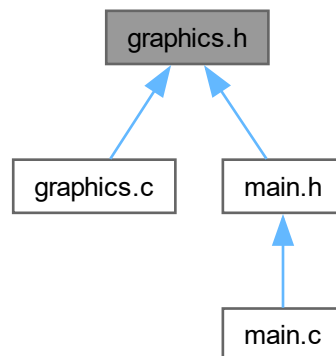
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <SDL2_gfxPrimitives.h>
#include <SDL2_rotozoom.h>
#include <SDL_ttf.h>
#include <stdio.h>
#include "types.h"
#include "source_reader.h"

```

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)
initializes the graphics
- int [initBoxes](#) ([graphics_t](#) *graphics, [node_t](#) *start_node, [box_t](#) *boxes)
initializes 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

<i>a</i>	string to append to
<i>b</i>	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

<i>start</i>	start box
<i>end</i>	end box
<i>graphics</i>	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

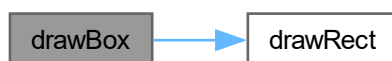
<i>box</i>	box to draw
<i>graphics</i>	collection of the important variables for the graphics

Returns

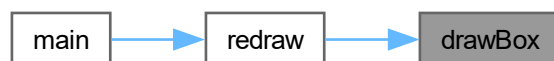
0 if success, -1 if error

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

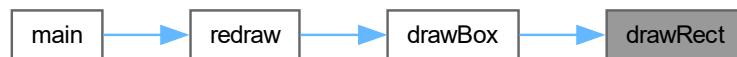
Parameters

<i>rect</i>	rectangle to draw
<i>graphics</i>	collection of the important variables for the graphics
<i>colour</i>	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

<i>boxes</i>	list of boxes
<i>box_count</i>	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 )
```

initializes the boxes

Parameters

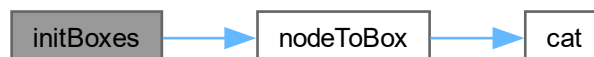
<i>graphics</i>	collection of the important variables for the graphics
<i>start_node</i>	the start node of the linked list
<i>boxes</i>	the list of boxes to append to

Returns

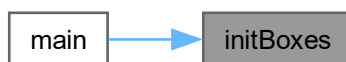
0 if success, -1 if error

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 )
```

initializes the graphics

Parameters

<i>graphics</i>	collection of the important variables for the graphics
<i>window</i>	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

<i>node</i>	start node
<i>boc</i>	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

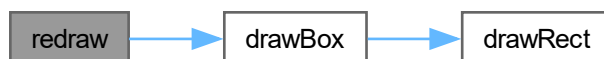
<i>graphics</i>	collection of the important variables for the graphics
<i>boxes</i>	the boxes to draw
<i>num_boxes</i>	the number of boxes

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

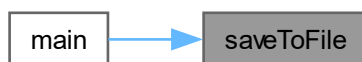
<i>graphics</i>	collection of the important variables for the graphics
<i>filename</i>	name of the file
<i>file_type</i>	the type of the file jpg or png

Returns

0 if success, -1 if error

Definition at line 206 of file [graphics.c](#).

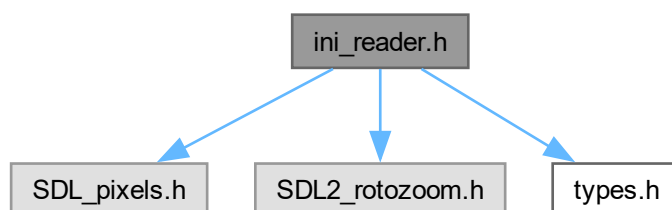
Here is the caller graph for this function:



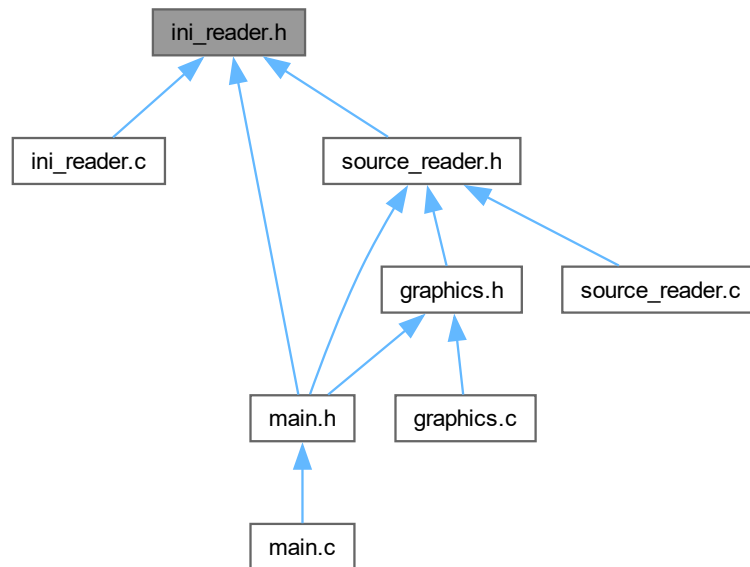
5.3 ini_reader.h File Reference

```
#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

<i>filename</i>	name of the ini file (including the .ini)
<i>theme</i>	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 )
```

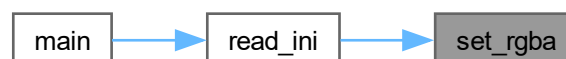
Sets the rgba of an `SDL_Colour`.

Parameters

<i>hex</i>	hexadecimal string beginning with an #
<i>colour</i>	pointer to the <code>SDL_Colour</code>

Definition at line 81 of file [ini_reader.c](#).

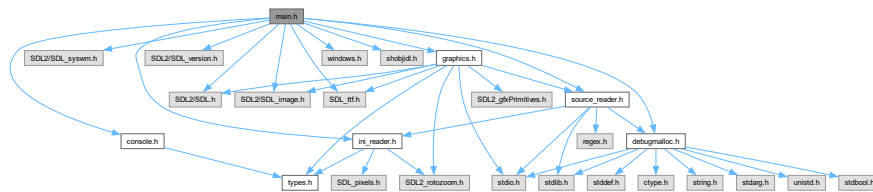
Here is the caller graph for this function:



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

<i>windowRef</i>	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

<i>windowRef</i>	win32 window handle
<i>name</i>	filter name
<i>file_spec</i>	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

<i>windowRef</i>	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

<i>window</i>	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.

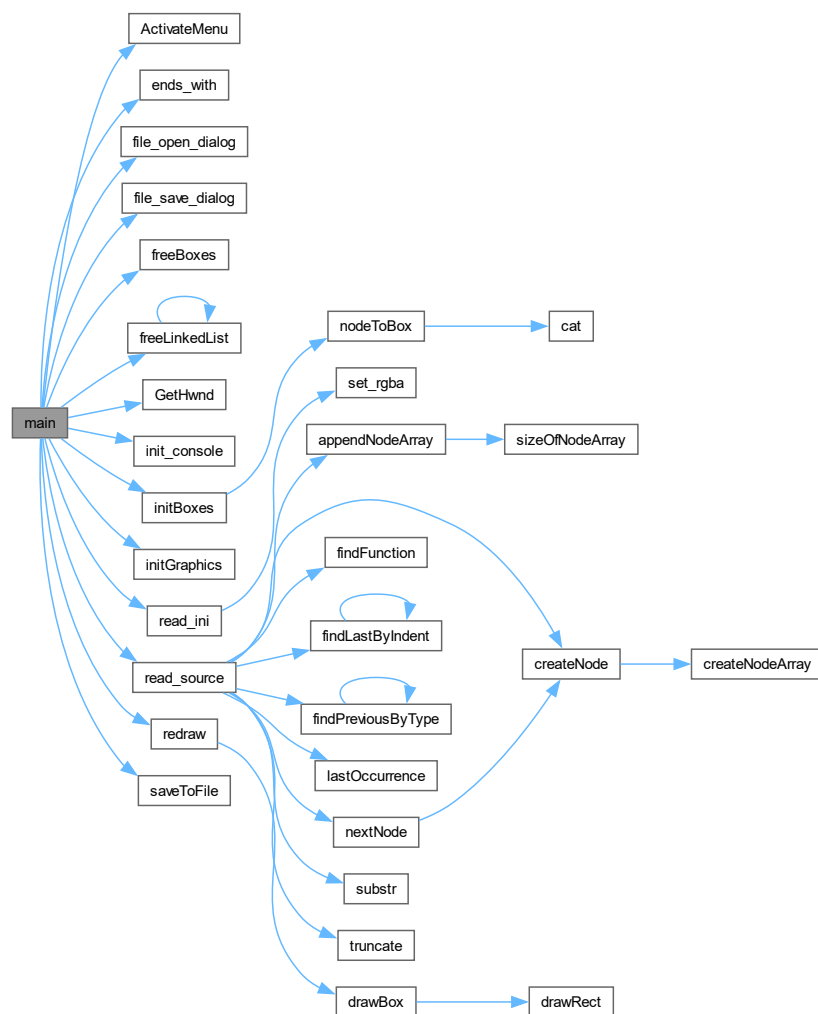
Parameters

<i>argc</i>	
<i>argv</i>	

Returns

Definition at line 3 of file [main.c](#).

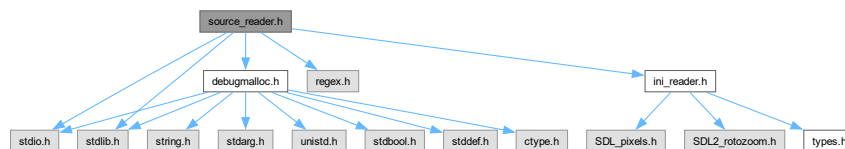
Here is the call graph for this function:



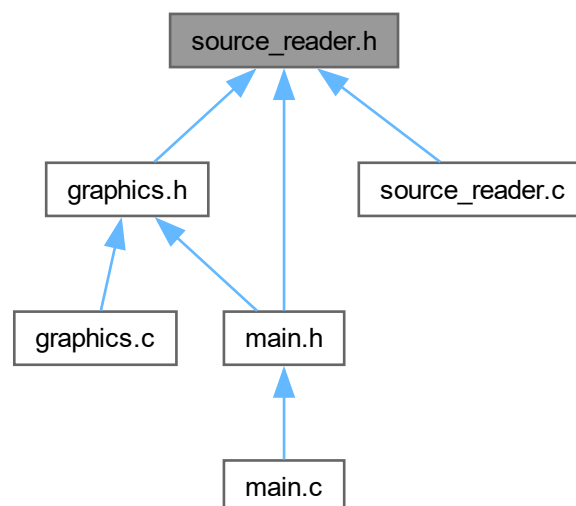
5.5 source_reader.h File Reference

```
#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 substrirng 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

<i>arr</i>	The array of node pointers to which the node will be appended.
<i>node</i>	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

<i>type</i>	type of the node
-------------	------------------

Returns

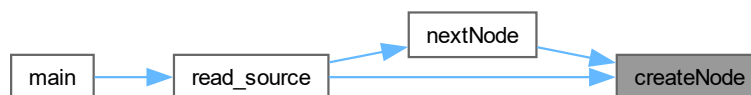
new node

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

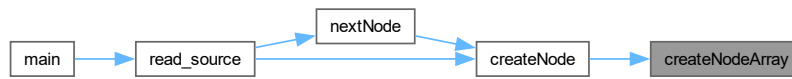
<i>size</i>	The number of node pointers to allocate in the array.
-------------	---

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

<i>start</i>	The starting node for the search.
<i>name</i>	The name of the node to find.
<i>indent</i>	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

<i>start</i>	The starting node for the search.
<i>type</i>	The type of the node to find.
<i>indent</i>	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

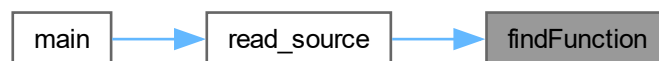
<i>node</i>	node to get name from
<i>functions</i>	functions list

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

<i>start</i>	The starting node for the search.
<i>indent</i>	The indentation level to consider for the search.

Returns

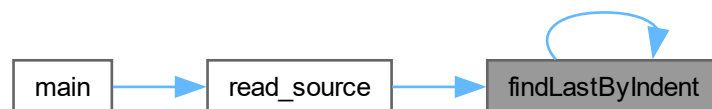
A pointer to the last node with the specified indentation level, or NULL if no such node is found.

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

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

Returns

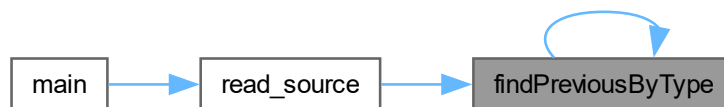
A pointer to the previous node with the specified type and indentation level, or NULL if no such node is found.

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

<i>start</i>	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 occurrence of a char in a string

Parameters

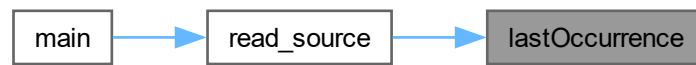
<i>str</i>	string to search
<i>c</i>	char to search for

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

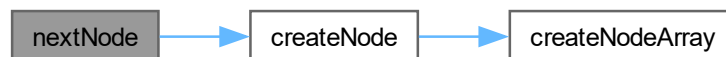
<i>node</i>	current node
-------------	--------------

Returns

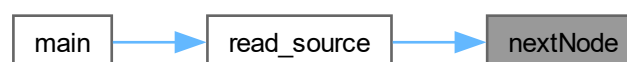
next node

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

<i>source_file</i>	to read
--------------------	---------

Returns

linked_list of all the lines

Structures

Function definitions

Variables

Function calls

Conditionals

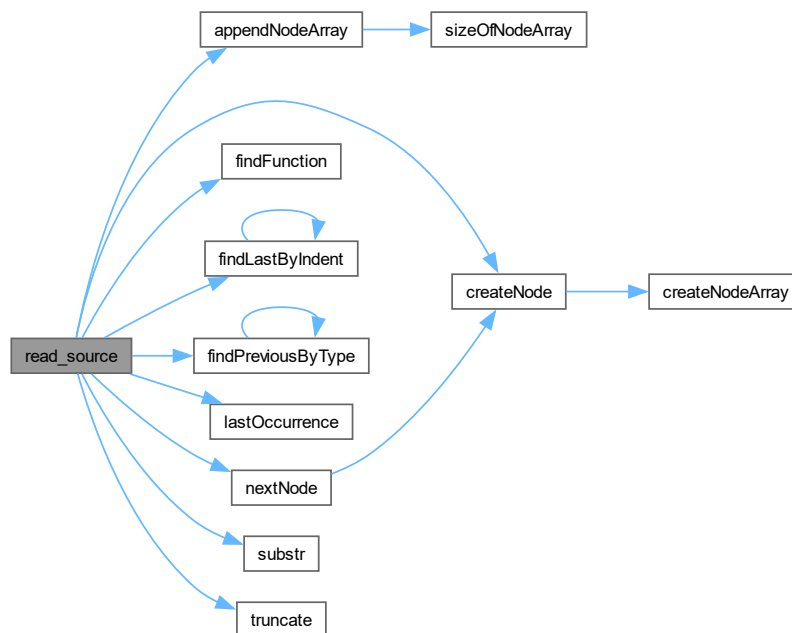
Loops

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

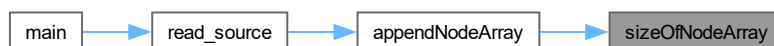
<i>arr</i>	The array of node pointers.
------------	-----------------------------

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 substring from the first appearance of `start` until the first appearance of `end`.

Parameters

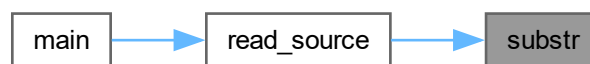
<i>str</i>	string to get substring from
<i>start</i>	starting char
<i>end</i>	ending char

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 )
```

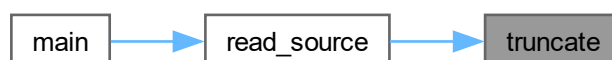
Truncates a string from spaces and endlines
if `len < 0`, string length doesn't change
if `len > 0`, truncates to `len`.

Parameters

<i>str</i>	string to get truncated
<i>len</i>	truncation length

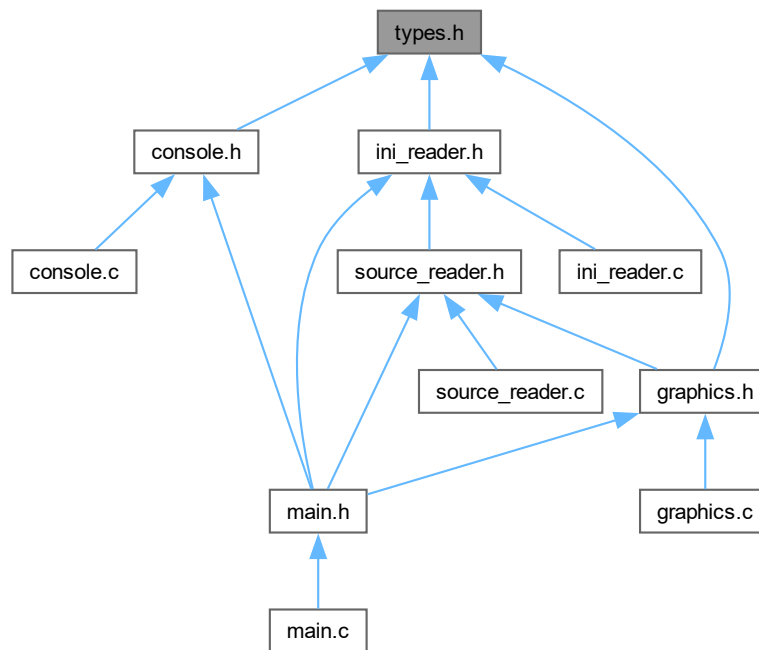
Definition at line 554 of file [source_reader.c](#).

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

- ActivateMenu
 - main.h, [41](#)
- appendNodeArray
 - source_reader.h, [48](#)
- args
 - func_type_t, [10](#)
 - struct_type_t, [18](#)
- background
 - colour_t, [9](#)
 - ini_reader.h, [37](#)
- box_t, [7](#)
 - radius, [8](#)
 - rect, [8](#)
 - text, [8](#)
 - type, [8](#)
- call_num
 - node, [14](#)
- case_
 - source_reader.h, [47](#)
- cat
 - graphics.h, [27](#)
- colour_t, [8](#)
 - background, [9](#)
 - text, [9](#)
- condition
 - loop_cond_type_t, [12](#)
- conditional
 - ini_reader.h, [37](#)
- conditionals
 - theme_t, [19](#)
- console.h, [23](#)
 - ends_with, [24](#)
 - init_console, [24](#)
- context_e
 - ini_reader.h, [36](#)
- createNode
 - source_reader.h, [48](#)
- createNodeArray
 - source_reader.h, [49](#)
- DEFAULT_FILE_TYPE
 - types.h, [60](#)
- do_while
 - source_reader.h, [47](#)
- drawArrow
 - graphics.h, [27](#)
- drawBox
 - graphics.h, [28](#)
- drawRect
 - graphics.h, [28](#)
- else_
 - source_reader.h, [47](#)
- else_if
 - source_reader.h, [47](#)
- ends_with
 - console.h, [24](#)
- file_open_dialog
 - main.h, [42](#)
- file_save_dialog
 - main.h, [42](#)
- file_type_e
 - types.h, [61](#)
- file_type_jpg
 - types.h, [61](#)
- file_type_png
 - types.h, [61](#)
- findByName
 - source_reader.h, [50](#)
- findByType
 - source_reader.h, [51](#)
- findFunction
 - source_reader.h, [52](#)
- findLastByIndent
 - source_reader.h, [52](#)
- findPreviousByType
 - source_reader.h, [53](#)
- finished
 - loop_cond_type_t, [12](#)
- font
 - graphics_t, [11](#)
- for_
 - source_reader.h, [47](#)
- freeBoxes
 - graphics.h, [29](#)
- freeLinkedList
 - source_reader.h, [54](#)
- func
 - node, [14](#)
- func_call
 - source_reader.h, [47](#)
- func_type_t, [9](#)
 - args, [10](#)
 - return_type, [10](#)
- function
 - ini_reader.h, [37](#)
- functions

- theme_t, 19
- GetHwnd
 - main.h, 43
- goList
 - node, 15
- graphics.h, 25
 - cat, 27
 - drawArrow, 27
 - drawBox, 28
 - drawRect, 28
 - freeBoxes, 29
 - initBoxes, 30
 - initGraphics, 31
 - nodeToBox, 31
 - redraw, 32
 - saveToFile, 33
- graphics_t, 10
 - font, 11
 - pixel_format, 11
 - renderer, 11
 - scale, 11
 - theme, 11
- h
 - rect_t, 17
- ID_EXIT
 - main.h, 40
- ID_LOAD_THEME
 - main.h, 40
- ID_OPEN_FILE
 - main.h, 40
- ID_RESET_THEME
 - main.h, 40
- ID_SAVE_FLOW
 - main.h, 40
- ID_ZOOM_IN
 - main.h, 40
- ID_ZOOM_OUT
 - main.h, 41
- ID_ZOOM_RESET
 - main.h, 41
- if_
 - source_reader.h, 47
- indent
 - node, 15
- ini_reader.h, 34
 - background, 37
 - conditional, 37
 - context_e, 36
 - function, 37
 - loop, 37
 - main_, 37
 - read_ini, 37
 - set_rgba, 38
 - structs, 37
 - sub_context_e, 37
 - text, 37
 - variable, 37
- init_console
 - console.h, 24
- initBoxes
 - graphics.h, 30
- initGraphics
 - graphics.h, 31
- key
 - mapping_t, 13
- lastOccurrence
 - source_reader.h, 55
- loop
 - ini_reader.h, 37
- loop_cond
 - node, 15
- loop_cond_e
 - source_reader.h, 47
- loop_cond_type_t, 12
 - condition, 12
 - finished, 12
 - type, 12
- loops
 - theme_t, 20
- main
 - main.h, 43
- main.h, 39
 - ActivateMenu, 41
 - file_open_dialog, 42
 - file_save_dialog, 42
 - GetHwnd, 43
 - ID_EXIT, 40
 - ID_LOAD_THEME, 40
 - ID_OPEN_FILE, 40
 - ID_RESET_THEME, 40
 - ID_SAVE_FLOW, 40
 - ID_ZOOM_IN, 40
 - ID_ZOOM_OUT, 41
 - ID_ZOOM_RESET, 41
 - main, 43
 - NHF_MAIN_H, 41
- main_
 - ini_reader.h, 37
 - theme_t, 20
- mapping_t, 13
 - key, 13
 - value, 13
- name
 - node, 15
- next
 - node, 15
- nextNode
 - source_reader.h, 56
- NHF_MAIN_H
 - main.h, 41
- node, 13

- call_num, 14
- func, 14
- goList, 15
- indent, 15
- loop_cond, 15
- name, 15
- next, 15
- other, 15
- struct_, 16
- type, 16
- variable, 16
- node_t
 - source_reader.h, 47
- nodeToBox
 - graphics.h, 31
- other
 - node, 15
 - source_reader.h, 47
- other_type_t, 16
 - value, 17
- pixel_format
 - graphics_t, 11
- radius
 - box_t, 8
- read_ini
 - ini_reader.h, 37
- read_source
 - source_reader.h, 56
- rect
 - box_t, 8
- rect_t, 17
 - h, 17
 - w, 17
 - x, 17
 - y, 18
- redraw
 - graphics.h, 32
- renderer
 - graphics_t, 11
- return_type
 - func_type_t, 20
- saveToFile
 - graphics.h, 33
- scale
 - graphics_t, 11
- set_rgba
 - ini_reader.h, 38
- sizeofNodeArray
 - source_reader.h, 58
- source_reader.h, 45
 - appendNodeArray, 48
 - case_, 47
 - createNode, 48
 - createNodeArray, 49
 - do_while, 47
 - else_, 47
 - else_if, 47
 - findByName, 50
 - findByType, 51
 - findFunction, 52
 - findLastByIndent, 52
 - findPreviousByType, 53
 - for_, 47
 - freeLinkedList, 54
 - func_call, 47
 - if_, 47
 - lastOccurrence, 55
 - loop_cond_e, 47
 - nextNode, 56
 - node_t, 47
 - other, 47
 - read_source, 56
 - sizeofNodeArray, 58
 - substr, 58
 - switch_, 47
 - truncate, 59
 - type_e, 47
 - while_, 47
- Specification, 1
- struct_
 - node, 16
- struct_type_t, 18
 - args, 18
- structs
 - ini_reader.h, 37
 - theme_t, 20
- sub_context_e
 - ini_reader.h, 37
- substr
 - source_reader.h, 58
- switch_
 - source_reader.h, 47
- text
 - box_t, 8
 - colour_t, 9
 - ini_reader.h, 37
- theme
 - graphics_t, 11
- theme_t, 19
 - conditionals, 19
 - functions, 19
 - loops, 20
 - main_, 20
 - structs, 20
 - variables, 20
- truncate
 - source_reader.h, 59
- type
 - box_t, 8
 - loop_cond_type_t, 12
 - node, 16
 - variable_type_t, 21
- type_e

- source_reader.h, [47](#)
- types.h, [60](#)
 - DEFAULT_FILE_TYPE, [60](#)
 - file_type_e, [61](#)
 - file_type_jpg, [61](#)
 - file_type_png, [61](#)
- value
 - mapping_t, [13](#)
 - other_type_t, [17](#)
 - variable_type_t, [21](#)
- variable
 - ini_reader.h, [37](#)
 - node, [16](#)
- variable_type_t, [20](#)
 - type, [21](#)
 - value, [21](#)
- variables
 - theme_t, [20](#)
- w
 - rect_t, [17](#)
- while_
 - source_reader.h, [47](#)
- x
 - rect_t, [17](#)
- y
 - rect_t, [18](#)