Csuper - Compteur de Score Universel Permettant l’Exemption de Reflexion

4.0.0

Generated by Doxygen 1.8.6

Thu Jun 26 2014 23:37:58

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 csuStruct Struct Reference

```
#include <csu_struct.h>
```

Data Fields

- float version
- float size_max_name
- float day
- float month
- float year
- float nb_player
- game_config config
- char ** player_names
- float * total_points
- float * rank
- float * nb_turn
- float distributor
- float ** point

3.1.1 Detailed Description

Represent a csu file

3.1.2 Field Documentation

3.1.2.1 game_config config

The game configuration.

3.1.2.2 float day

Day of the structure creation.

3.1.2.3 float distributor

Index of the distributor.

3.1.2.4 float month

Month of the structure creation.

3.1.2.5 float nb_player

Number of player.

3.1.2.6 float∗ nb_turn

Array containing the number of turn of all players.

3.1.2.7 char∗∗ player_names

Array containing the name of all players.

3.1.2.8 float∗∗ point

Array containing the points of all players in each turn.

3.1.2.9 float∗ rank

Array containing the rank of all players.

3.1.2.10 float size_max_name

Maximum size that can reach a player name.

3.1.2.11 float∗ total_points

Array containing the total score of all players.

3.1.2.12 float version

Version of the structure.

3.1.2.13 float year

Year of the structure creation.

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

- csu_struct.h

3.2 game_config Struct Reference

```
#include <csu_struct.h>
```

Data Fields

- float nb_max
- char first_way
- char turn_by_turn
- char use_distributor
- char decimal_place
- char max
- char name [SIZE_MAX_NAME]
- float begin_score

3.2.1 Detailed Description

Represent a game configuration

3.2.2 Field Documentation

3.2.2.1 float begin_score

The score of all players in the beginning of the game

3.2.2.2 char decimal_place

The number of decimal place which are display

3.2.2.3 char first_way

Is 1 if the first those has the maximum of points, -1 otherwise

3.2.2.4 char max

Is 1 if the game use a maximum, 0 if it's a minimum

3.2.2.5 char name[SIZE_MAX_NAME]

The name of the game configuration

3.2.2.6 float nb_max

Number maximum or minimum that can reach a player.

3.2.2.7 char turn_by_turn

Is 1 if the game is on turn by turn, 0 otherwise

3.2.2.8 char use_distributor

Is 1 if the game use a distributor, 0 otherwise

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

- csu_struct.h

3.3 list_game_config Struct Reference

```
#include <game_config.h>
```

Data Fields

- int nb_config
- char ** name_game_config

3.3.1 Detailed Description

Represent a list of game configuration

3.3.2 Field Documentation

3.3.2.1 char** name_game_config

The list of the game configuration.

3.3.2.2 int nb_config

Number of game configuration.

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

- game_config.h

3.4 main_window_size Struct Reference

```
#include <preferences_files.h>
```

Data Fields

- int width
- int height
- int is_maximize

3.4.1 Detailed Description

All component of the man window size

3.4.2 Field Documentation

3.4.2.1 int height

The height of the main window

3.4.2.2 int is_maximize

Said if the main window is maximize or not

3.4.2.3 int width

The width of the main window

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

- preferences_files.h

3.5 toolbar_button_preferences_struct Struct Reference

```
#include <preferences_files.h>
```

Data Fields

- int new
- int open
- int save_as
- int separator_1
- int undo
- int redo
- int separator_2
- int cut
- int copy
- int paste
- int delete
- int separator_3
- int properties
- int separator_4
- int preferences
- int game_configuration_preferences
- int toolbar_button_preferences
- int separator_5
- int about

3.5.1 Detailed Description

Represent the toolbar button preferences

3.5.2 Field Documentation

3.5.2.1 int about

The about button

3.5.2.2 int copy

The copy button

3.5.2.3 int cut

The cut button

3.5.2.4 int delete

The delete button

3.5.2.5 int game_configuration_preferences

The game configuration preferences button

3.5.2.6 int new

The new button

3.5.2.7 int open

The open button

3.5.2.8 int paste

The paste button

3.5.2.9 int preferences

The preferences button

3.5.2.10 int properties

The properties button

3.5.2.11 int redo

The redo button

3.5.2.12 int save_as

The save_as button

3.5.2.13 int separator_1

The separator 1

3.5.2.14 int separator_2

The separator 2

3.5.2.15 int separator_3

The separator 3

3.5.2.16 int separator_4

The separator 4

3.5.2.17 int separator_5

The separator 5

3.5.2.18 int toolbar_button_preferences

The toolbar button preferences button

3.5.2.19 int undo

The undo button

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

- preferences_files.h

Chapter 4

File Documentation

4.1 csu_files.c File Reference

Files management.

```
#include "csu_files.h"
```

Functions

- FILE ∗ openFileCsuExtension (char file_name[ ], char mode[ ])
- csuStruct ∗ readCsuFile (char ∗file_name)
- int writeCsuFile (char ∗file_name, csuStruct ∗ptr_csu_struct)
- int writeFileNewTurn (char ∗file_name, csuStruct ∗ptr_csu_struct)

4.1.1 Detailed Description

Files management.

Author

Remi BERTHO

Date

27/04/14

Version

2.2.0

4.1.2 Function Documentation

4.1.2.1 FILE ∗ openFileCsuExtension (char *file_name[ ]*, char *mode[ ]*)

Open a file with his name and with a specific mode and add the file extension if necessary.

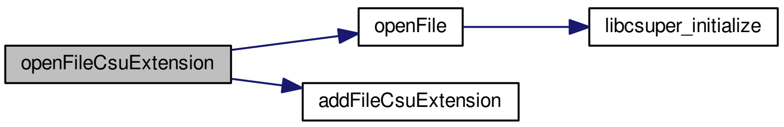
Parameters

`in`	*file_name[]*	the filename
`in`	*mode[]*	the mode

Returns

a pointer on the open file, NULL if there is a problem

Here is the call graph for this function:



4.1.2.2 csuStruct ∗ readCsuFile (char ∗ *file_name*)

Read the file with the name file_name and copy the result in a new csu structure.

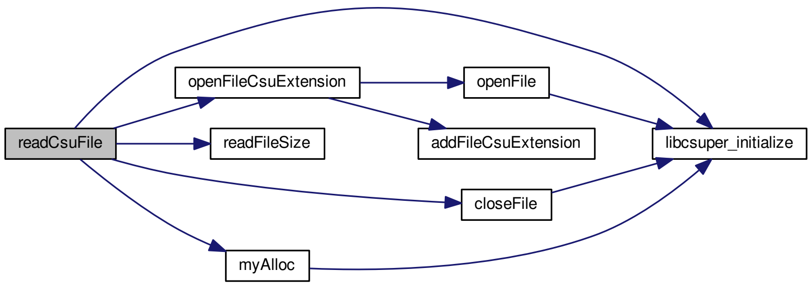
Parameters

`in`	*file_name[]*	the filename

Returns

a pointer on the new csu structure, NULL if there is a problem

Here is the call graph for this function:



4.1.2.3 int writeCsuFile (char ∗ *file_name,* csuStruct ∗ *ptr_csu_struct*)

Write a csu file

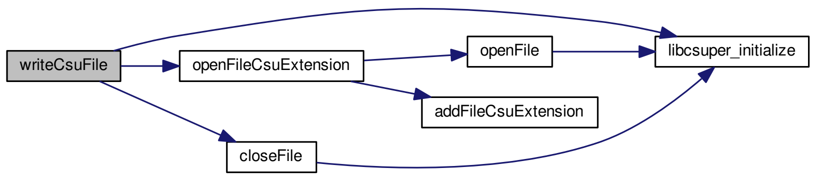
Parameters

in	**file_name*	the filename
in	**ptr_csu_struct*	a pointer on a csuStruct

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.1.2.4 void writeFileNewTurn (char ∗ *file_name*, csuStruct ∗ *ptr_csu_struct*)

Update the file with the new scores

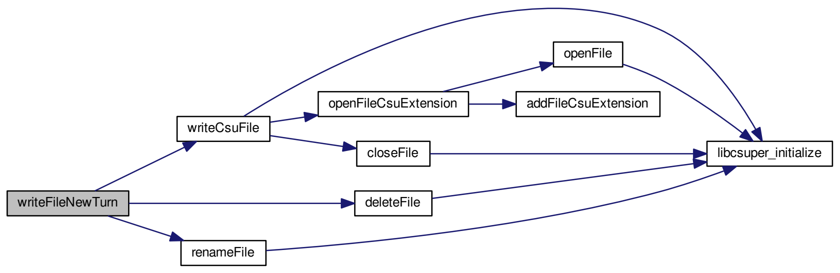
Parameters

in	**file_name*	the filename
in	**ptr_csu_struct*	a pointer on a csuStruct

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.2 csu_files.h File Reference

Files management.

```
#include "csu_struct.h"
#include <unistd.h>
```

Macros

- #define SIZE_MAX_FILE_NAME 1024
- #define FILE_EXTENSION "csu"
- #define STRING_CHECK_CSU_FILE "CompteurScoreUniversel"

Functions

- FILE ∗ openFileCsuExtension (char file_name[ ], char mode[ ])
- csuStruct ∗ readCsuFile (char ∗file_name)
- int writeCsuFile (char ∗file_name, csuStruct ∗ptr_csu_struct)
- int writeFileNewTurn (char ∗file_name, csuStruct ∗ptr_csu_struct)

4.2.1 Detailed Description

Files management.

Author

Remi BERTHO

Date

16/04/14

Version

2.2.0

4.2.2 Macro Definition Documentation

4.2.2.1 #define FILE_EXTENSION "csu"

Define the file extension to "csu"

4.2.2.2 #define SIZE_MAX_FILE_NAME 1024

Define the size maximum of a filename to 1024

4.2.2.3 #define STRING_CHECK_CSU_FILE "CompteurScoreUniversel"

String for checking if the file is a csu file.

4.2.3 Function Documentation

4.2.3.1 FILE∗ openFileCsuExtension (char *file_name[ ],* char *mode[ ]*)

Open a file with his name and with a specific mode and add the file extension if necessary.

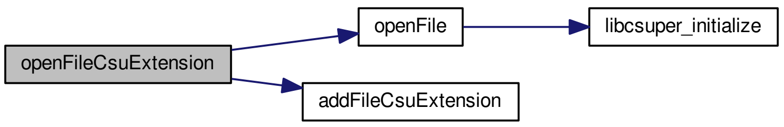
Parameters

`in`	*file_name[]*	the filename
`in`	*mode[]*	the mode

Returns

a pointer on the open file, NULL if there is a problem

Here is the call graph for this function:



4.2.3.2 csuStruct∗ readCsuFile (char ∗ *file_name*)

Read the file with the name file_name and copy the result in a new csu structure.

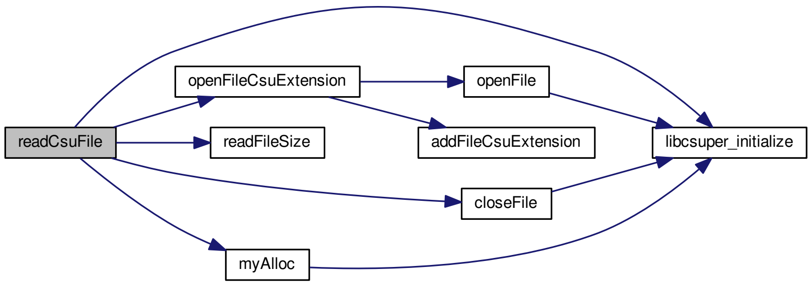
Parameters

`in`	*file_name[]*	the filename

Returns

a pointer on the new csu structure, NULL if there is a problem

Here is the call graph for this function:



4.2.3.3 int writeCsuFile (char ∗ *file_name,* csuStruct ∗ *ptr_csu_struct*)

Write a csu file

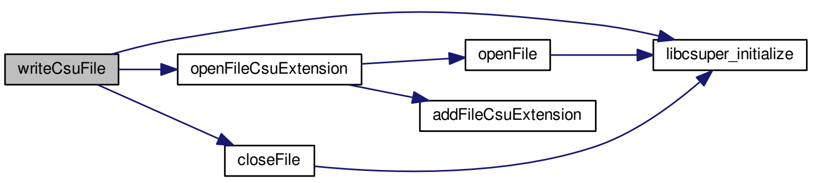
Parameters

in	**file_name*	the filename
in	**ptr_csu_struct*	a pointer on a csuStruct

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.2.3.4 int writeFileNewTurn (char ∗ *file_name*, csuStruct ∗ *ptr_csu_struct*)

Update the file with the new scores

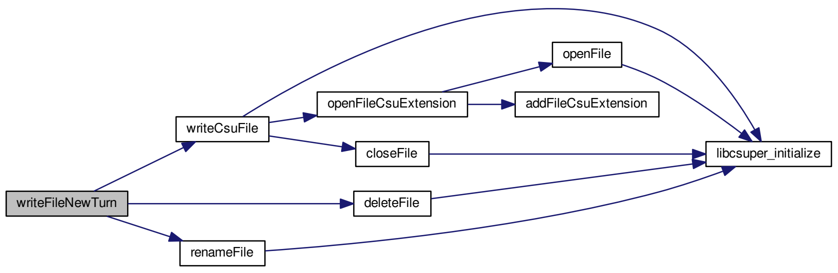
Parameters

in	**file_name*	the filename
in	**ptr_csu_struct*	a pointer on a csuStruct

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.3 csu_struct.c File Reference

Management of the csu files.

```
#include "csu_struct.h"
```

Functions

- csuStruct ∗ newCsuStruct (float nb_player, game_config config)
- void closeCsuStruct (csuStruct ∗ptr_csu_struct)
- void startNewTurn (csuStruct ∗ptr_csu_struct, int index_player)
- void endNewTurn (csuStruct ∗ptr_csu_struct, int index_player)
- void rankCalculation (csuStruct ∗ptr_csu_struct)
- int searchIndexFromPosition (csuStruct ∗ptr_csu_struct, int position, int ∗nb)
- void addDistributorCsuStruct (csuStruct ∗ptr_csu_struct, char ∗distributor_name)
- int exceedMaxNumber (csuStruct ∗ptr_csu_struct)
- int maxNbTurn (csuStruct ∗ptr_csu_struct)
- int searchPlayerIndex (csuStruct ∗ptr_csu_struct, char ∗player_name)
- int differentsPlayerName (csuStruct ∗ptr_csu_struct)
- csuStruct ∗ copyCsuStruct (csuStruct ∗ptr_csu_struct)

4.3.1 Detailed Description

Management of the csu files.

Author

Remi BERTHO

Date

15/06/14

Version

4.0.4

4.3.2 Function Documentation

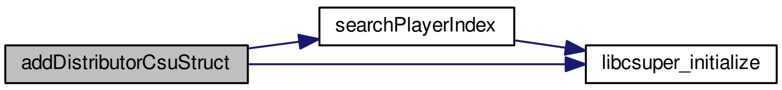
4.3.2.1 void addDistributorCsuStruct (**csuStruct** ∗ *ptr_csu_struct,* char ∗ *distributor_name*)

Add the distributor on the structure

Parameters

in	∗*distributor_-name*	the name of the distributor
in	∗*ptr_csu_struct*	a pointer on a csuStruct

Here is the call graph for this function:



4.3.2.2 void closeCsuStruct (**csuStruct** ∗ *ptr_csu_struct*)

Free a csuStruct

Parameters

`in,out`	**ptr_csu_struct*	a pointer to the csuStruct

4.3.2.3 csuStruct ∗ copyCsuStruct (csuStruct ∗ *ptr_csu_struct*)

Copy a csu structure

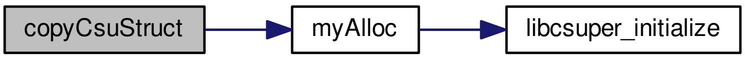
Parameters

`in`	**ptr_csu_struct*	a pointer on a csuStruct

Returns

a pointer on the new csu structure

Here is the call graph for this function:



4.3.2.4 int differentsPlayerName (csuStruct ∗ *ptr_csu_struct*)

Search the index of a person

Parameters

`in`	**ptr_csu_struct*	a pointer on a csuStruct

Returns

MY_TRUE if all player names are different, MY_FALSE otherwise

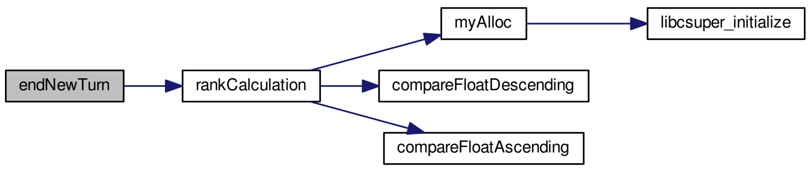
4.3.2.5 void endNewTurn (csuStruct ∗ *ptr_csu_struct,* int *index_player*)

Update the total points, the number of turn, the distributor and the rank for a new turn

Parameters

`in,out`	**ptr_csu_struct*	a pointer on a csuStruct
`in,out`	*index_player*	index_player the index of the player who begin a new turn, -1 if everybody begin a new turn

Here is the call graph for this function:



4.3.2.6 int exceedMaxNumber (csuStruct ∗ *ptr_csu_struct*)

Check if someone exceed the maximum number

Parameters

in	∗*ptr_csu_struct*	a pointer on a csuStruct

Returns

MY_TRUE if someone exceed, MY_FALSE otherwise

4.3.2.7 int maxNbTurn (csuStruct ∗ *ptr_csu_struct*)

Search the maximal number of turn

Parameters

in	∗*ptr_csu_struct*	a pointer on a csuStruct

Returns

the maximal number of turn

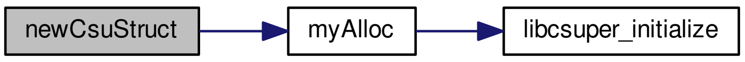
4.3.2.8 csuStruct ∗ newCsuStruct (float *nb_player*, game_config *config*)

Create a new csuStruct from a game configuration and the number of player.

Parameters

in	*nb_player*	the number of player
in	*config*	the game configuration

Here is the call graph for this function:



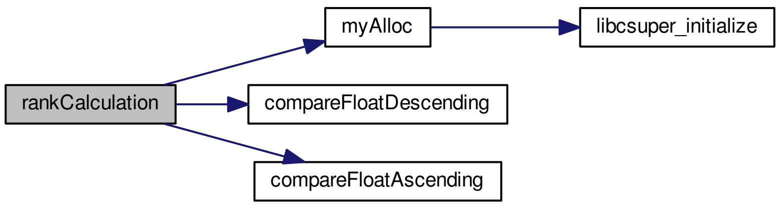
4.3.2.9 void rankCalculation (csuStruct ∗ *ptr_csu_struct*)

Calculate the rank

Parameters

`in,out`	**ptr_csu_struct*	a pointer on a csuStruct

Here is the call graph for this function:



4.3.2.10 int searchIndexFromPosition (csuStruct ∗ *ptr_csu_struct,* int *position,* int ∗ *nb*)

Search the index in the array of the person who is the 'position' position

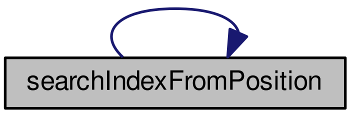
Parameters

`in,out`	**ptr_csu_struct*	a pointer on a csuStruct
`in,out`	*position*	the position
`in,out`	*nb*	the nbth player who have the position will be selected

Returns

the index or NULL if the position doesn't exist

Here is the call graph for this function:



4.3.2.11 int searchPlayerIndex (csuStruct ∗ *ptr_csu_struct,* char ∗ *player_name*)

Search the index of a person

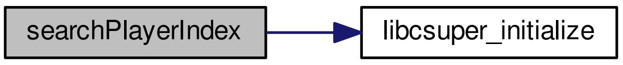
Parameters

in	*player_name	the name of the player
in	*ptr_csu_struct	a pointer on a csuStruct

Returns

the index, -1 if there is not found

Here is the call graph for this function:



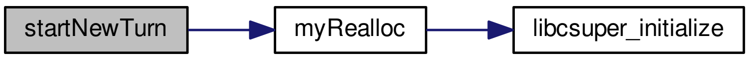
4.3.2.12 void startNewTurn (csuStruct * *ptr_csu_struct*, int *index_player*)

Reallocate the memory for the point to begin a new turn.

Parameters

in,out	*ptr_csu_struct	a pointer on a csuStruct
in,out	index_player	the index of the player who begin a new turn, -1 if everybody begin a new turn

Here is the call graph for this function:



4.4 csu_struct.h File Reference

Management of the csu files header.

```
#include <time.h>
#include <float.h>
#include "share.h"
```

Data Structures

- struct game_config
- struct csuStruct

Macros

- #define SIZE_MAX_NAME 30
- #define VERSION 1.4

Functions

- csuStruct ∗ newCsuStruct (float nb_player, game_config config)
- void closeCsuStruct (csuStruct ∗ptr_csu_struct)
- void startNewTurn (csuStruct ∗ptr_csu_struct, int index_player)
- void endNewTurn (csuStruct ∗ptr_csu_struct, int index_player)
- void rankCalculation (csuStruct ∗ptr_csu_struct)
- int searchIndexFromPosition (csuStruct ∗ptr_csu_struct, int position, int ∗nb)
- void addDistributorCsuStruct (csuStruct ∗ptr_csu_struct, char ∗distributor_name)
- int exceedMaxNumber (csuStruct ∗ptr_csu_struct)
- int maxNbTurn (csuStruct ∗ptr_csu_struct)
- int searchPlayerIndex (csuStruct ∗ptr_csu_struct, char ∗player_name)
- int differentsPlayerName (csuStruct ∗ptr_csu_struct)
- csuStruct ∗ copyCsuStruct (csuStruct ∗ptr_csu_struct)

4.4.1 Detailed Description

Management of the csu files header.

Author

Remi BERTHO

Date

16/06/14

Version

4.0.0

4.4.2 Macro Definition Documentation

4.4.2.1 #define SIZE_MAX_NAME 30

Define size max of name to 30

4.4.2.2 #define VERSION 1.4

Define the version to 1.4

4.4.3 Function Documentation

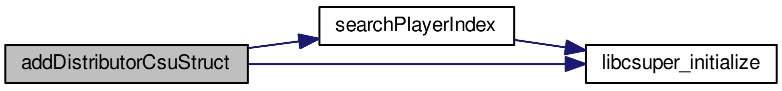
4.4.3.1 void addDistributorCsuStruct (csuStruct ∗ *ptr_csu_struct,* char ∗ *distributor_name*)

Add the distributor on the structure

Parameters

`in`	**distributor_-name*	the name of the distributor
`in`	**ptr_csu_struct*	a pointer on a csuStruct

Here is the call graph for this function:



4.4.3.2 void closeCsuStruct (**csuStruct** ∗ *ptr_csu_struct*)

Free a csuStruct

Parameters

`in,out`	**ptr_csu_struct*	a pointer to the csuStruct

4.4.3.3 **csuStruct**∗ copyCsuStruct (**csuStruct** ∗ *ptr_csu_struct*)

Copy a csu structure

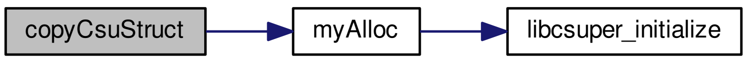
Parameters

`in`	**ptr_csu_struct*	a pointer on a csuStruct

Returns

a pointer on the new csu structure

Here is the call graph for this function:



4.4.3.4 int differentsPlayerName (**csuStruct** ∗ *ptr_csu_struct*)

Search the index of a person

Parameters

in	**ptr_csu_struct*	a pointer on a csuStruct

Returns

MY_TRUE if all player names are different, MY_FALSE otherwise

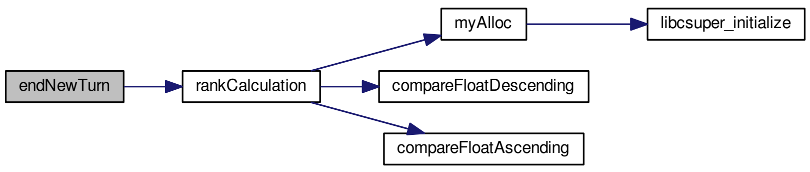
4.4.3.5 void endNewTurn (**csuStruct** ∗ *ptr_csu_struct,* int *index_player*)

Update the total points, the number of turn, the distributor and the rank for a new turn

Parameters

in,out	**ptr_csu_struct*	a pointer on a csuStruct
in,out	*index_player*	index_player the index of the player who begin a new turn, -1 if everybody begin a new turn

Here is the call graph for this function:



4.4.3.6 int exceedMaxNumber (**csuStruct** ∗ *ptr_csu_struct*)

Check if someone exceed the maximum number

Parameters

in	**ptr_csu_struct*	a pointer on a csuStruct

Returns

MY_TRUE if someone exceed, MY_FALSE otherwise

4.4.3.7 int maxNbTurn (**csuStruct** ∗ *ptr_csu_struct*)

Search the maximal number of turn

Parameters

in	**ptr_csu_struct*	a pointer on a csuStruct

Returns

the maximal number of turn

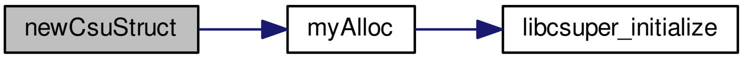
4.4.3.8 **csuStruct**∗ newCsuStruct (float *nb_player,* **game_config** *config*)

Create a new csuStruct from a game configuration and the number of player.

Parameters

in	*nb_player*	the number of player
in	*config*	the game configuration

Here is the call graph for this function:



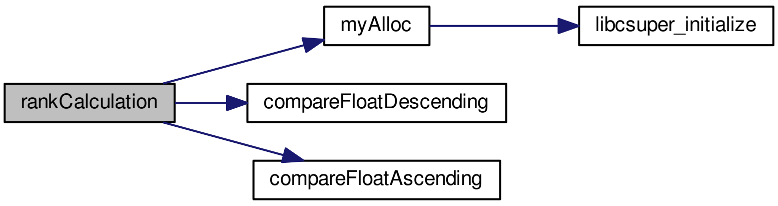
4.4.3.9 void rankCalculation (**csuStruct** ∗ *ptr_csu_struct*)

Calculate the rank

Parameters

in,out	∗*ptr_csu_struct*	a pointer on a csuStruct

Here is the call graph for this function:



4.4.3.10 int searchIndexFromPosition (**csuStruct** ∗ *ptr_csu_struct,* int *position,* int ∗ *nb*)

Search the index in the array of the person who is the 'position' position

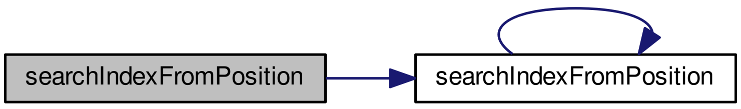
Parameters

in,out	∗*ptr_csu_struct*	a pointer on a csuStruct
in,out	*position*	the position
in,out	*nb*	the nbth player who have the position will be selected

Returns

the index or NULL if the position doesn't exist

Here is the call graph for this function:



4.4.3.11 int searchPlayerIndex (**csuStruct** ∗ *ptr_csu_struct,* char ∗ *player_name*)

Search the index of a person

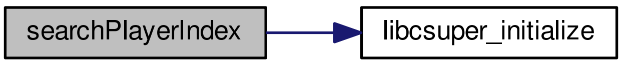
Parameters

`in`	∗*player_name*	the name of the player
`in`	∗*ptr_csu_struct*	a pointer on a csuStruct

Returns

the index, -1 if there is not found

Here is the call graph for this function:



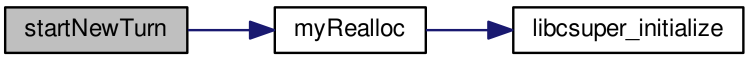
4.4.3.12 void startNewTurn (**csuStruct** ∗ *ptr_csu_struct,* int *index_player*)

Reallocate the memory for the point to begin a new turn.

Parameters

`in,out`	∗*ptr_csu_struct*	a pointer on a csuStruct
`in,out`	*index_player*	the index of the player who begin a new turn, -1 if everybody begin a new turn

Here is the call graph for this function:



4.5 game_config.c File Reference

Game configuration.

```
#include "game_config.h"
```

Functions

- list_game_config ∗ newListGameConfig (int nb_config)
- void closeListGameConfig (list_game_config ∗ptr_list_config)
- int makeConfigListFile (char ∗home_path)
- list_game_config ∗ readConfigListFile (char ∗home_path)
- int addConfigListFile (char ∗new_config_name, char ∗home_path)
- int removeConfigListFile (int index_delete, list_game_config ∗ptr_list_config, char ∗home_path)
- int newConfigFile (game_config config, char ∗home_path)
- int removeConfigFile (char ∗config_name, char ∗home_path)
- int readConfigFile (int index_read, list_game_config ∗ptr_list_config, game_config ∗ptr_config, char ∗home-_path)
- int exportConfigFile (char ∗home_path, char ∗file_name)
- int importConfigFile (char ∗home_path, char ∗file_name)

4.5.1 Detailed Description

Game configuration.

Author

Remi BERTHO

Date

29/04/14

Version

2.4.0

4.5.2 Function Documentation

4.5.2.1 int addConfigListFile (char ∗ *new_config_name,* char ∗ *home_path*)

Add a new game configuration into the file which contain the list of game configuration.

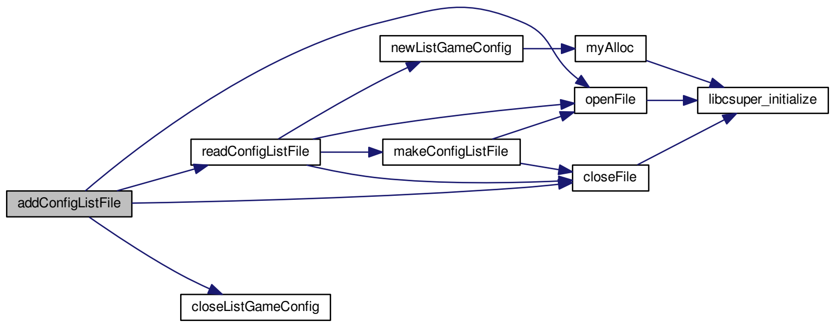
Parameters

in	*new_config_-name*	the name of the new game configuration
in	*home_path*	the path to the home directory

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.5.2.2 void closeListGameConfig (list_game_config ∗ *ptr_list_config*)

Free a list of game configuration

Parameters

in	*∗ptr_list_config*	a pointer on a list of game configuration

4.5.2.3 int exportConfigFile (char ∗ *home_path,* char ∗ *file_name*)

Export all config file into a file.

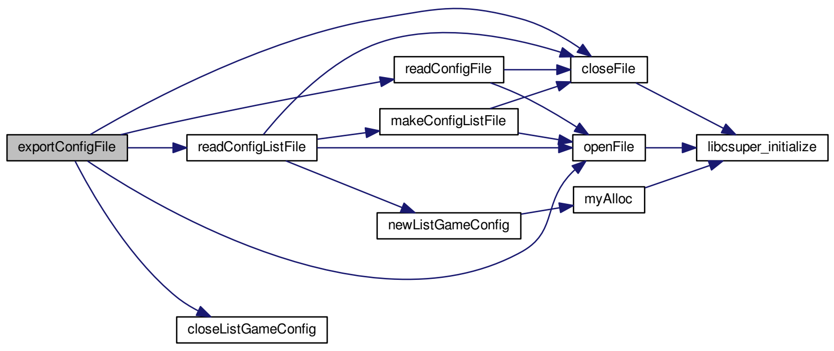
Parameters

in	*file_name*	the filename of the exported file.
in	*home_path*	the path to the home directory

Returns

a list_game_config

Here is the call graph for this function:



4.5.2.4 int importConfigFile (char ∗ *home_path,* char ∗ *file_name*)

Import all config file from a file.

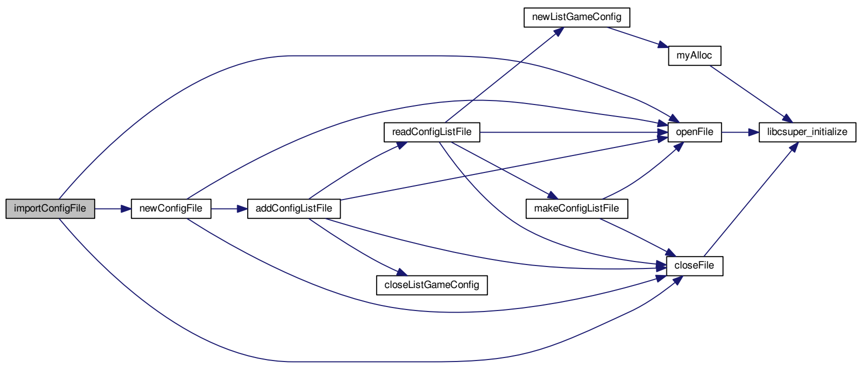
Parameters

in	*file_name*	the filename of the exported file.
in	*home_path*	the path to the home directory

Returns

a list_game_config

Here is the call graph for this function:



4.5.2.5 int makeConfigListFile (char ∗ *home_path*)

Create the folder which contain the games configurations and the files which contain the list of games configurations

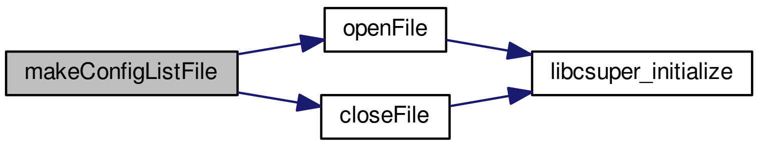
Parameters

in	∗*home_path*	the path to the home directory

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.5.2.6 int newConfigFile (**game_config** *config,* char ∗ *home_path*)

Create a game configuration file and put it into the game configuration file list.

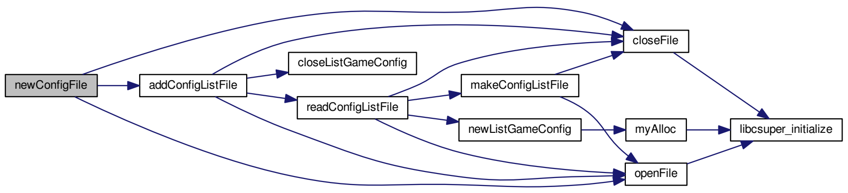
Parameters

in	*config*	the gale configuration
in	*home_path*	the path to the home directory

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.5.2.7 **list_game_config** ∗ newListGameConfig (int *nb_config*)

Create a list of game configuration.

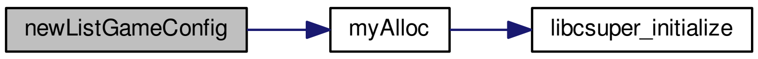
Parameters

in	*nb_config*	the number of game configuration

Returns

une list_game_config

Here is the call graph for this function:



4.5.2.8 int readConfigFile (int *index_read*, list_game_config ∗ *ptr_list_config*, game_config ∗ *ptr_config*, char ∗ *home_path*)

Read a game configuration file.

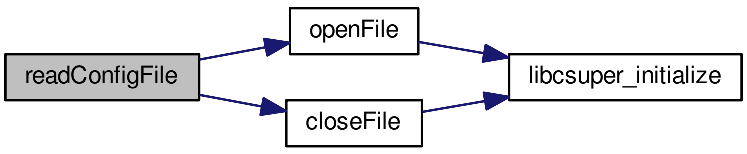
Parameters

in	*index_read*	the index of the game configuration to be read
in	*ptr_list_config*	a pointer on the game configration list
in	*ptr_config*	a pointer on a game configuration
in	*home_path*	the path to the home directory

Returns

a list_game_config

Here is the call graph for this function:



4.5.2.9 list_game_config ∗ readConfigListFile (char ∗ *home_path*)

Read the file which contain the list of game configuration.

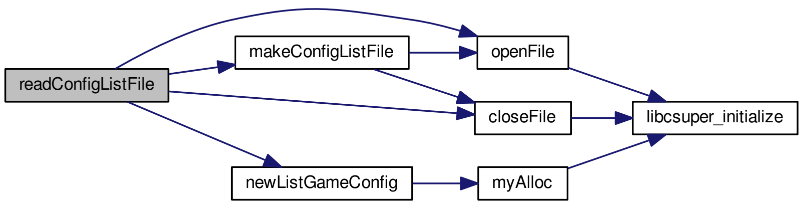
Parameters

in	∗*home_path*	the path to the home directory

Returns

a list_game_config

Here is the call graph for this function:



4.5.2.10 int removeConfigFile (char ∗ *config_name*, char ∗ *home_path*)

Delete a game configuration.

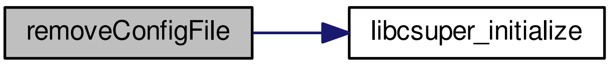
Parameters

in	*config_name*	the name of the game configuration which will be deleted
in	*home_path*	the path to the home directory

Returns

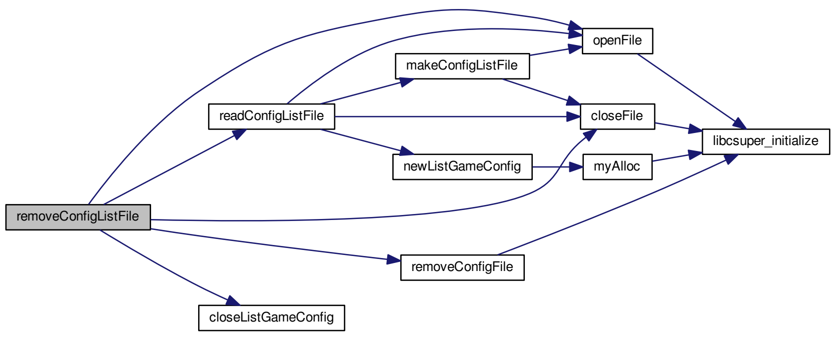
MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.5.2.11 int removeConfigListFile (int *index_delete,* list_game_config ∗ *ptr_list_config,* char ∗ *home_path*)

Here is the call graph for this function:



4.6 game_config.h File Reference

Game configurations.

```
#include <math.h>
#include "csu_struct.h"
#include "preferences_files.h"
```

Data Structures

- struct list_game_config

Macros

- #define CONFIGURATION_FOLDER_NAME "config"
- #define CONFIGURATION_FILE_NAME "configuration"
- #define STRING_CHECK_GAME_CONFIG "Csuper_Game_Configuration"

Functions

- list_game_config ∗ newListGameConfig (int nb_config)
- void closeListGameConfig (list_game_config ∗ptr_list_config)
- int makeConfigListFile (char ∗home_path)
- list_game_config ∗ readConfigListFile (char ∗home_path)
- int addConfigListFile (char ∗new_config_name, char ∗home_path)
- int removeConfigListFile (int index_delete, list_game_config ∗ptr_list_config, char ∗home_path)
- int newConfigFile (game_config config, char ∗home_path)
- int removeConfigFile (char ∗config_name, char ∗home_path)
- int readConfigFile (int index_read, list_game_config ∗ptr_list_config, game_config ∗ptr_config, char ∗home-_path)
- int exportConfigFile (char ∗home_path, char ∗file_name)
- int importConfigFile (char ∗home_path, char ∗file_name)

4.6.1 Detailed Description

Game configurations.

Author

Remi BERTHO

Date

29/04/14

Version

2.4.0

4.6.2 Macro Definition Documentation

4.6.2.1 #define CONFIGURATION_FILE_NAME "configuration"

Define the name of the file which contain the list of the game configurations

4.6.2.2 #define CONFIGURATION_FOLDER_NAME "config"

Define the name of the folder which contain the game configurations

4.6.2.3 #define STRING_CHECK_GAME_CONFIG "Csuper_Game_Configuration"

String for checking if the file is game configuration file.

4.6.3 Function Documentation

4.6.3.1 int addConfigListFile (char ∗ *new_config_name*, char ∗ *home_path*)

Add a new game configuration into the file which contain the list of game configuration.

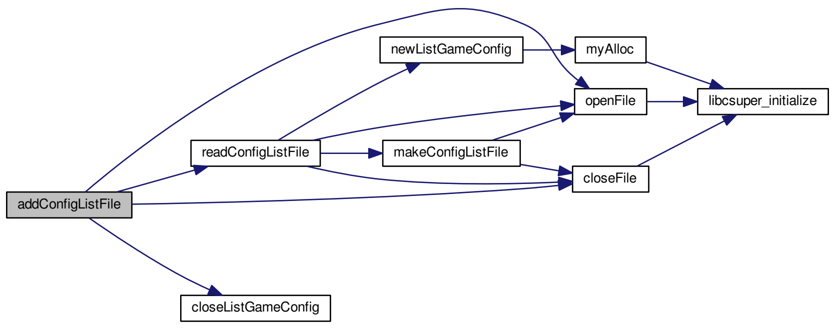
Parameters

`in`	*new_config_-name*	the name of the new game configuration
`in`	*home_path*	the path to the home directory

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.6.3.2 void closeListGameConfig (list_game_config ∗ *ptr_list_config*)

Free a list of game configuration

Parameters

`in`	∗*ptr_list_config*	a pointer on a list of game configuration

4.6.3.3 int exportConfigFile (char ∗ *home_path,* char ∗ *file_name*)

Export all config file into a file.

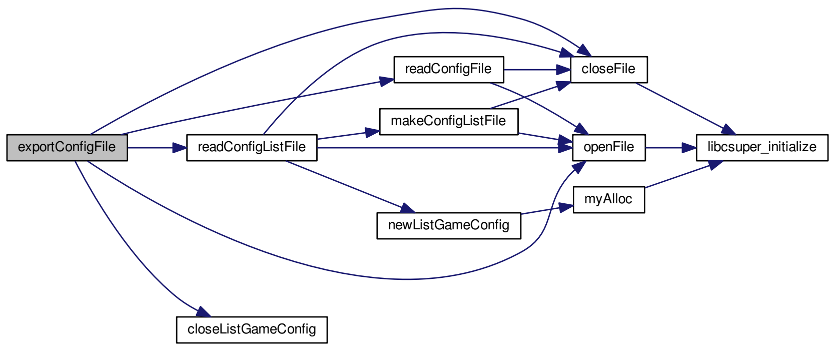
Parameters

`in`	*file_name*	the filename of the exported file.
`in`	*home_path*	the path to the home directory

Returns

a list_game_config

Here is the call graph for this function:



4.6.3.4 int importConfigFile (char ∗ *home_path,* char ∗ *file_name*)

Import all config file from a file.

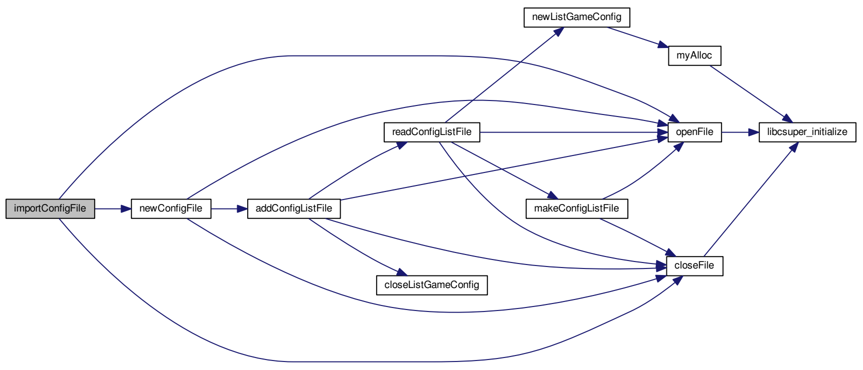
Parameters

in	*file_name*	the filename of the exported file.
in	*home_path*	the path to the home directory

Returns

a list_game_config

Here is the call graph for this function:



4.6.3.5 int makeConfigListFile (char ∗ *home_path*)

Create the folder which contain the games configurations and the files which contain the list of games configurations

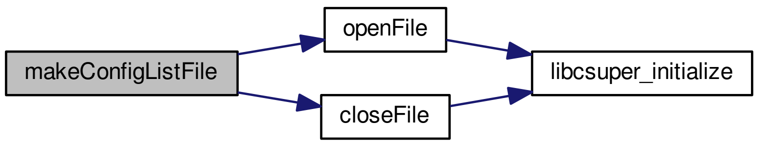
Parameters

in	*∗home_path*	the path to the home directory

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.6.3.6 int newConfigFile (**game_config** *config,* char ∗ *home_path*)

Create a game configuration file and put it into the game configuration file list.

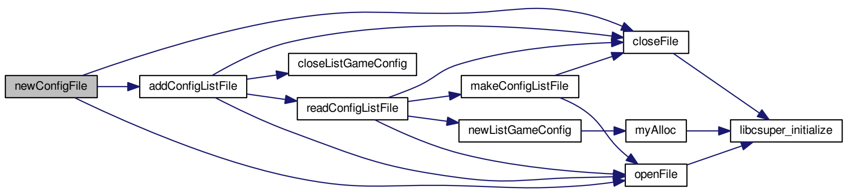
Parameters

in	*config*	the gale configuration
in	*home_path*	the path to the home directory

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.6.3.7 **list_game_config**∗ newListGameConfig (int *nb_config*)

Create a list of game configuration.

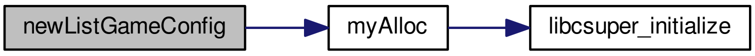
Parameters

`in`	*nb_config*	the number of game configuration

Returns

une list_game_config

Here is the call graph for this function:



4.6.3.8 int readConfigFile (int *index_read,* list_game_config ∗ *ptr_list_config,* game_config ∗ *ptr_config,* char ∗ *home_path*)

Read a game configuration file.

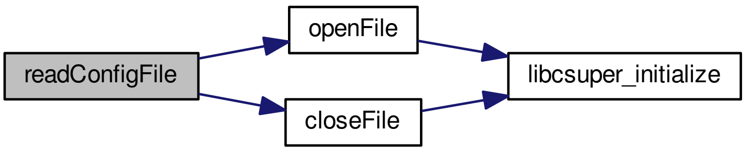
Parameters

`in`	*index_read*	the index of the game configuration to be read
`in`	*ptr_list_config*	a pointer on the game configration list
`in`	*ptr_config*	a pointer on a game configuration
`in`	*home_path*	the path to the home directory

Returns

a list_game_config

Here is the call graph for this function:



4.6.3.9 list_game_config∗ readConfigListFile (char ∗ *home_path*)

Read the file which contain the list of game configuration.

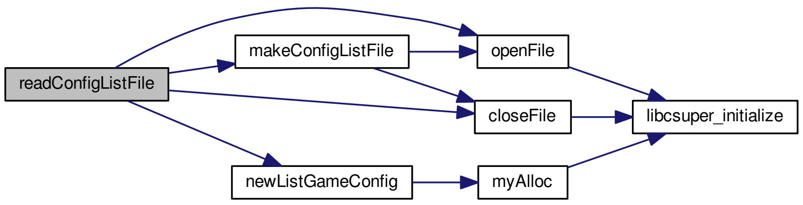
Parameters

in	∗*home_path*	the path to the home directory

Returns

a list_game_config

Here is the call graph for this function:



4.6.3.10 int removeConfigFile (char ∗ *config_name,* char ∗ *home_path*)

Delete a game configuration.

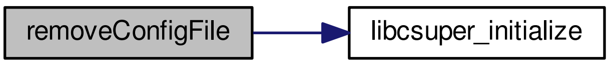
Parameters

in	*config_name*	the name of the game configuration which will be deleted
in	*home_path*	the path to the home directory

Returns

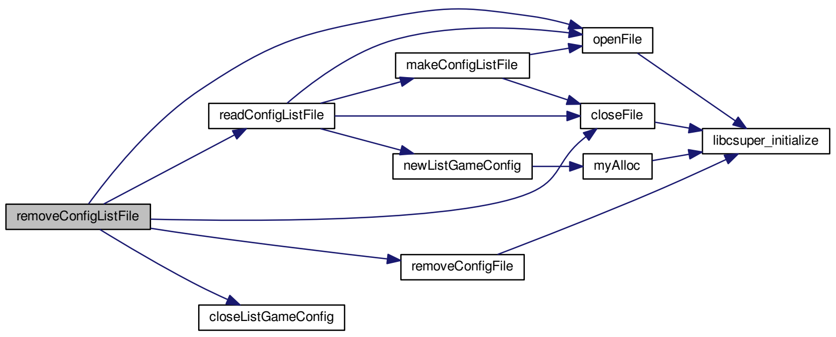
MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.6.3.11 int removeConfigListFile (int *index_delete,* list_game_config ∗ *ptr_list_config,* char ∗ *home_path*)

Here is the call graph for this function:



4.7 libcsuper.h File Reference

Inclusion of all header files of libcsuper.

```
#include "csu_struct.h"
#include "share.h"
#include "csu_files.h"
#include "preferences_files.h"
#include "main_argument.h"
#include "game_config.h"
```

4.7.1 Detailed Description

Inclusion of all header files of libcsuper.

Author

Remi BERTHO

Date

05/04/14

Version

2.2.0

4.8 main_argument.c File Reference

Begin csuper.

```
#include "main_argument.h"
```

Functions

- int searchArgument (int argc, char ∗argv[ ], int ∗function, int ∗file_place)
- void displayHelp ()

4.8.1 Detailed Description

Begin csuper.

Author

Remi BERTHO

Date

16/04/14

Version

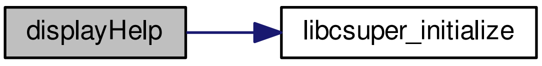
2.2.0

4.8.2 Function Documentation

4.8.2.1 void displayHelp ( )

Display the help

Here is the call graph for this function:



4.8.2.2 int searchArgument (int *argc*, char ∗ *argv[ ]*, int ∗ *function*, int ∗ *file_place*)

Search the argument passed to the main function

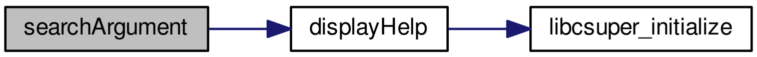
Parameters

`in`	*argc*	the number of argument
`in`	*argv*	the array of argument
`in`	*function*	integer which determine which function run
`in`	*file_place*	integer which determine the index of the filename

Returns

MY_TRUE if the function founded an argument, MY_FALSE otherwise

Here is the call graph for this function:



4.9 main_argument.h File Reference

Begin csuper.

`#include "share.h"`

Macros

- #define STRING_READ_FILE "--read"
- #define STRING_READ_FILE_RED "-r"
- #define READ_FILE 0
- #define STRING_OPEN_FILE "--open"
- #define STRING_OPEN_FILE_RED "-o"
- #define OPEN_FILE 1
- #define STRING_HELP "--help"
- #define STRING_HELP_RED "-h"
- #define HELP 2

Functions

- int searchArgument (int argc, char ∗argv[ ], int ∗function, int ∗file_place)
- void displayHelp ()

4.9.1 Detailed Description

Begin csuper.

Author

Remi BERTHO

Date

16/04/14

Version

2.2.0

4.9.2 Macro Definition Documentation

4.9.2.1 #define HELP 2

Define the call help to 2

4.9.2.2 #define OPEN_FILE 1

Define the call to read a file to 1

4.9.2.3 #define READ_FILE 0

Define the call to read a file to 0

4.9.2.4 #define STRING_HELP "--help"

Define the argument which call help to "--help"

4.9.2.5 #define STRING_HELP_RED "-h"

Define the reduce argument which call help to "-h"

4.9.2.6 #define STRING_OPEN_FILE "--open"

Define the argument which call to open a file to "--open"

4.9.2.7 #define STRING_OPEN_FILE_RED "-o"

Define the reduce argument which call to open a file to "-o"

4.9.2.8 #define STRING_READ_FILE "--read"

Define the argument which call to read a file to "--read"

4.9.2.9 #define STRING_READ_FILE_RED "-r"

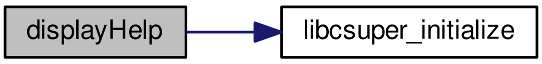
Define the reduce argument which call to read a file to "-r"

4.9.3 Function Documentation

4.9.3.1 void displayHelp ( )

Display the help

Here is the call graph for this function:



4.9.3.2 int searchArgument (int *argc,* char ∗ *argv[ ],* int ∗ *function,* int ∗ *file_place*)

Search the argument passed to the main function

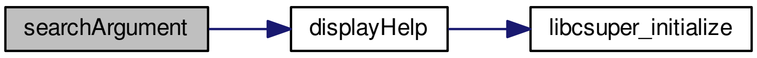
Parameters

`in`	*argc*	the number of argument
`in`	*argv*	the array of argument
`in`	*function*	integer which determine which function run
`in`	*file_place*	integer which determine the index of the filename

Returns

MY_TRUE if the function founded an argument, MY_FALSE otherwise

Here is the call graph for this function:



4.10 preferences_files.c File Reference

Function which store preferences into files.

```
#include "preferences_files.h"
```

Functions

- void createPreferencesFolder (char ∗home_path)
- int createFileToolbarButtonPreferences (char ∗home_path, toolbar_button_preferences_struct toolbar)
- int readFileToolbarButtonPreferences (char ∗home_path, toolbar_button_preferences_struct ∗toolbar)
- int differentsToolbarButtonPreferencesStruct (toolbar_button_preferences_struct toolbar1, toolbar_button_preferences_struct toolbar2)
- int createFileMainWidowSize (char ∗home_path, main_window_size size)
- int readFileMainWidowSize (char ∗home_path, main_window_size ∗size)

- int createFileSystemPath ()
- int readFileSystemPath (char ∗file_name)
- int readSystemPath (char ∗file_name)
- int changeSystemPath (char ∗new_path)
- int checkPath (char ∗path)
- void readHomePath (char ∗path)
- void readHomePathSlash (char ∗path)

4.10.1 Detailed Description

Function which store preferences into files.

Author

Remi BERTHO

Date

24/06/14

Version

4.0.0

4.10.2 Function Documentation

4.10.2.1 int changeSystemPath (char ∗ *new_path*)

Change the system path

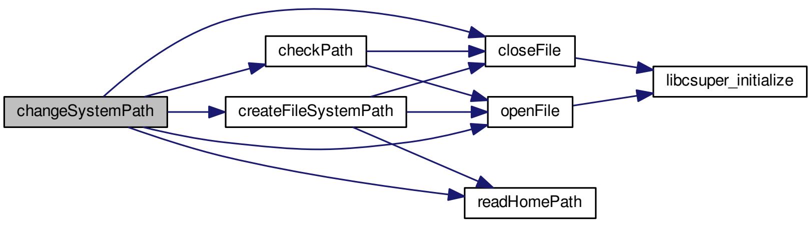
Parameters

in,out	∗*new_path*	the new path

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.10.2.2 int checkPath (char ∗ *path*)

Test if the path is valid

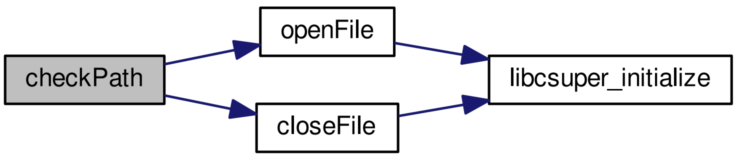
Parameters

`in,out`	∗*path*	the path

Returns

MY_TRUE if the path is valid OK, MY_FALSE otherwise

Here is the call graph for this function:



4.10.2.3 int createFileMainWidowSize (char ∗ *home_path*, main_window_size *size*)

Create the file which contain the main window size

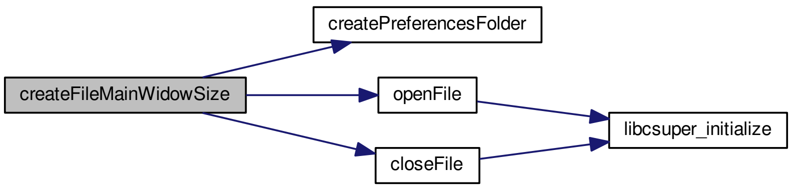
Parameters

`in`	*home_path*	the path to the home directory
`in`	*size*	the size of the main window

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



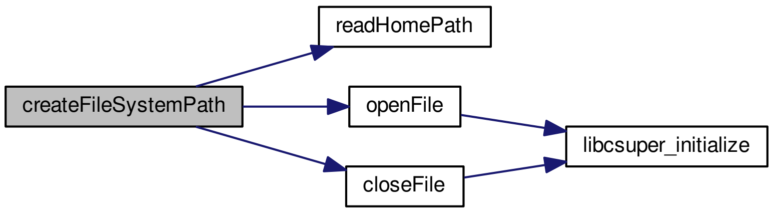
4.10.2.4 void createFileSystemPath ( )

Create the folder and the file which contain the system path

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.10.2.5 int createFileToolbarButtonPreferences (char * *home_path*, toolbar_button_preferences_struct *toolbar*)

Create the file which contain the preferences for the toolbar button

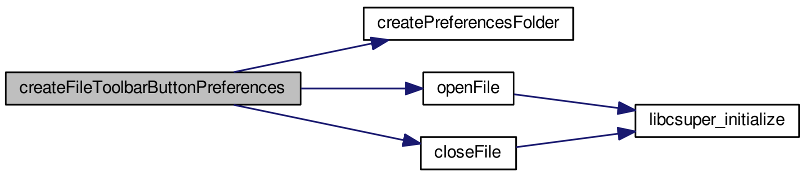
Parameters

`in`	*home_path*	the path to the home directory
`in`	*toolbar*	the toolbar button preferences

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.10.2.6 void createPreferencesFolder (char * *home_path*)

Create the folder which contain all preferences

Parameters

in	*home_path*	the path to the home directory

4.10.2.7 int differentsToolbarButtonPreferencesStruct (**toolbar_button_preferences_struct** *toolbar1*, **toolbar_button_preferences_struct** *toolbar2*)

Test if the two toolbar button preferences are different

Parameters

in	*toolbar1*	the first toolbar button preferences
in	*toolbar2*	the second toolbar button preferences

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

4.10.2.8 int readFileMainWidowSize (char ∗ *home_path*, **main_window_size** ∗ *size*)

Read the file which contain the main window size

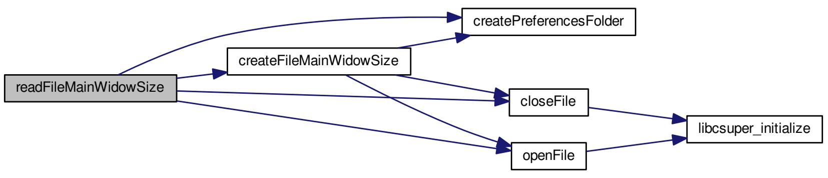
Parameters

in	*home_path*	the path to the home directory
in	*size*	the size of the main window

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.10.2.9 int readFileSystemPath (char ∗ *file_name*)

Read the system path and the path read to the filename

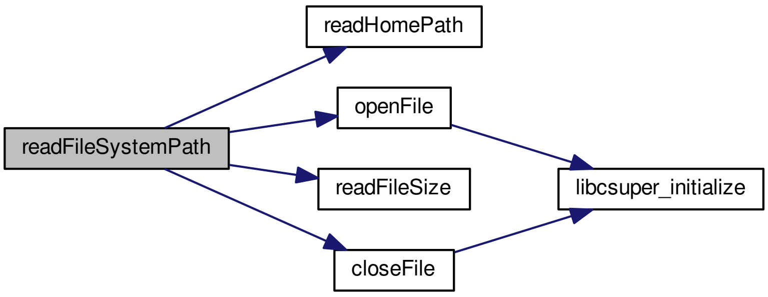
Parameters

in,out	∗*file_name*	the filename

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.10.2.10 int readFileToolbarButtonPreferences (char ∗ *home_path,* toolbar_button_preferences_struct ∗ *toolbar*)

Read the file which contain the preferences for the toolbar button

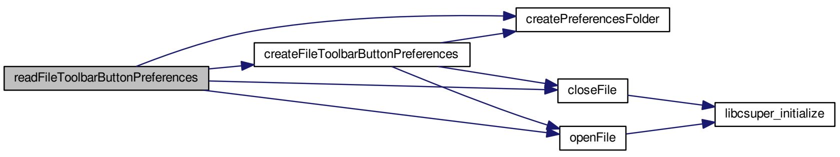
Parameters

`in`	*home_path*	the path to the home directory
`in`	*toolbar*	the toolbar button preferences

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:


createPreferencesFolder
createFileToolbarButtonPreferences
readFileToolbarButtonPreferences
closeFile
libcsuper_initialize
openFile

4.10.2.11 void readHomePath (char ∗ *path*)

Read the home path

Parameters

in,out	*path*	the path

Read the home path with a slash at the end

Parameters

in,out	*path*	the path

4.10.2.12 void readHomePathSlash (char ∗ *path*)

4.10.2.13 int readSystemPath (char ∗ *file_name*)

Add the system path, if the file system path doesn't exist, it create it.

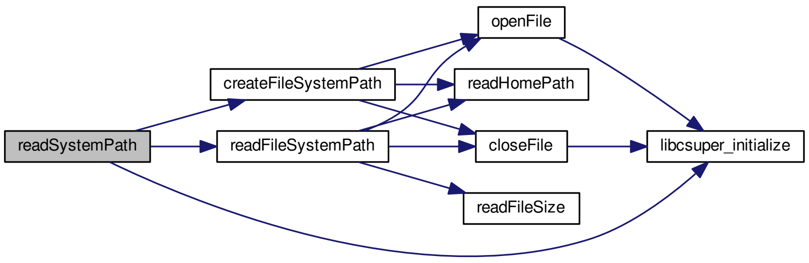
Parameters

in,out	∗*file_name*	the filename

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.11 preferences_files.h File Reference

Prototypes des fonctions qui l'emrankment des fichiers sauvegardes.

```
#include <sys/stat.h>
#include <sys/types.h>
#include "csu_struct.h"
#include "csu_files.h"
```

Data Structures

- struct toolbar_button_preferences_struct
- struct main_window_size

Macros

- #define FILENAME_SYSTEM_PATH "system_path.txt"

- #define FILENAME_TOOLBAR_BUTTON_PREFERENCES "toolbar_button_preferences.txt"
- #define FILENAME_MAIN_WINDOW_SIZE "main_window_size.txt"
- #define PREFERENCES_FOLDER_NAME ".csuper"

Functions

- void createPreferencesFolder (char ∗home_path)
- int createFileToolbarButtonPreferences (char ∗home_path, toolbar_button_preferences_struct toolbar)
- int readFileToolbarButtonPreferences (char ∗home_path, toolbar_button_preferences_struct ∗toolbar)
- int differentsToolbarButtonPreferencesStruct (toolbar_button_preferences_struct toolbar1, toolbar_button_-preferences_struct toolbar2)
- int createFileMainWidowSize (char ∗home_path, main_window_size size)
- int readFileMainWidowSize (char ∗home_path, main_window_size ∗size)
- int createFileSystemPath ()
- int readFileSystemPath (char ∗file_name)
- int readSystemPath (char ∗file_name)
- int changeSystemPath (char ∗new_path)
- int checkPath (char ∗path)
- void readHomePath (char ∗path)
- void readHomePathSlash (char ∗path)

4.11.1 Detailed Description

Prototypes des fonctions qui l'emrankment des fichiers sauvegardes.

Author

Remi BERTHO

Date

24/06/14

Version

4.0.0

4.11.2 Macro Definition Documentation

4.11.2.1 #define FILENAME_MAIN_WINDOW_SIZE "main_window_size.txt"

4.11.2.2 #define FILENAME_SYSTEM_PATH "system_path.txt"

Define filename of the file which contain the system path

4.11.2.3 #define FILENAME_TOOLBAR_BUTTON_PREFERENCES "toolbar_button_preferences.txt"

Define filename of the file which contain the toolbar button preferences

4.11.2.4 #define PREFERENCES_FOLDER_NAME ".csuper"

Define the folder name of the csuper preferences

4.11.3 Function Documentation

4.11.3.1 int changeSystemPath (char ∗ *new_path*)

Change the system path

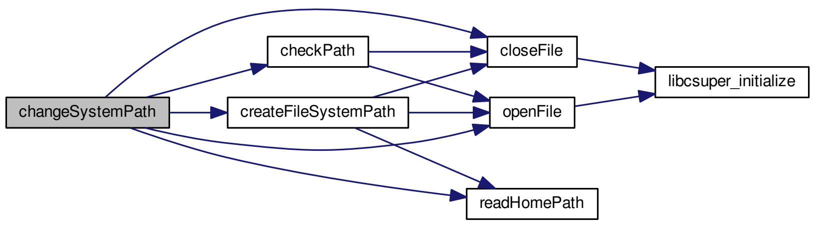
Parameters

`in,out`	*∗new_path*	the new path

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.11.3.2 int checkPath (char ∗ *path*)

Test if the path is valid

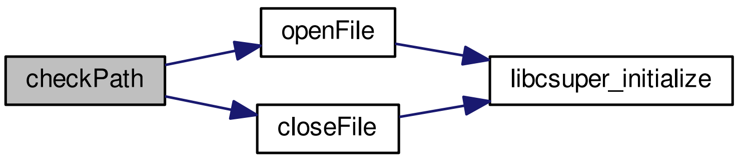
Parameters

`in,out`	*∗path*	the path

Returns

MY_TRUE if the path is valid OK, MY_FALSE otherwise

Here is the call graph for this function:



4.11.3.3 int createFileMainWidowSize (char ∗ *home_path*, main_window_size *size*)

Create the file which contain the main window size

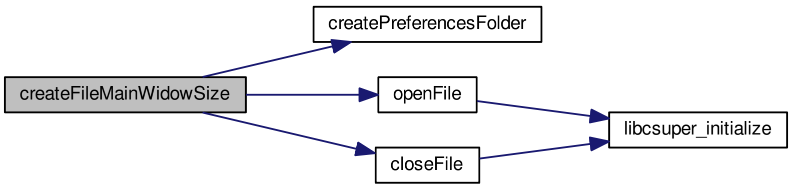
Parameters

in	*home_path*	the path to the home directory
in	*size*	the size of the main window

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



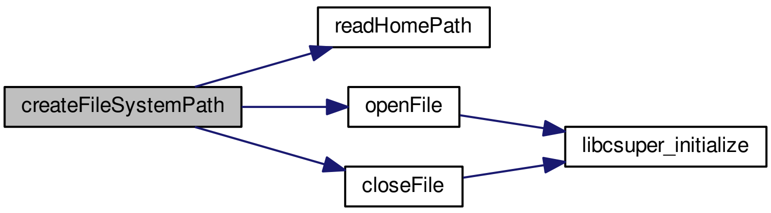
4.11.3.4 int createFileSystemPath ( )

Create the folder and the file which contain the system path

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.11.3.5 int createFileToolbarButtonPreferences (char ∗ *home_path,* toolbar_button_preferences_struct *toolbar*)

Create the file which contain the preferences for the toolbar button

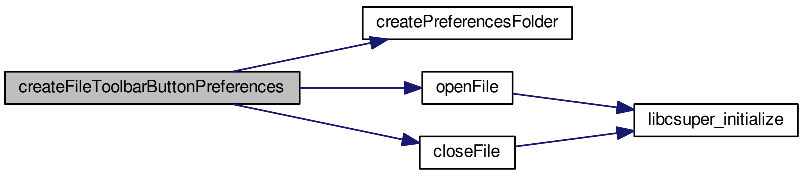
Parameters

in	*home_path*	the path to the home directory
in	*toolbar*	the toolbar button preferences

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.11.3.6 void createPreferencesFolder (char ∗ *home_path*)

Create the folder which contain all preferences

Parameters

in	*home_path*	the path to the home directory

4.11.3.7 int differentsToolbarButtonPreferencesStruct (toolbar_button_preferences_struct *toolbar1*, toolbar_button_preferences_struct *toolbar2*)

Test if the two toolbar button preferences are different

Parameters

in	*toolbar1*	the first toolbar button preferences
in	*toolbar2*	the second toolbar button preferences

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

4.11.3.8 int readFileMainWidowSize (char ∗ *home_path*, main_window_size ∗ *size*)

Read the file which contain the main window size

Parameters

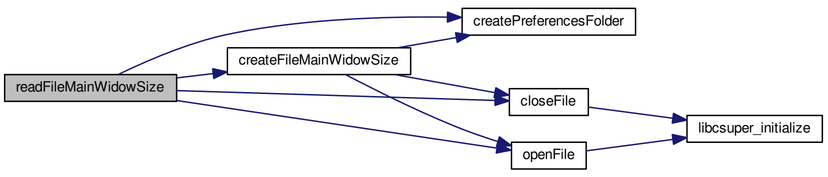
in	*home_path*	the path to the home directory

`in`	*size*	the size of the main window

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.11.3.9 int readFileSystemPath (char ∗ *file_name*)

Read the system path and the path read to the filename

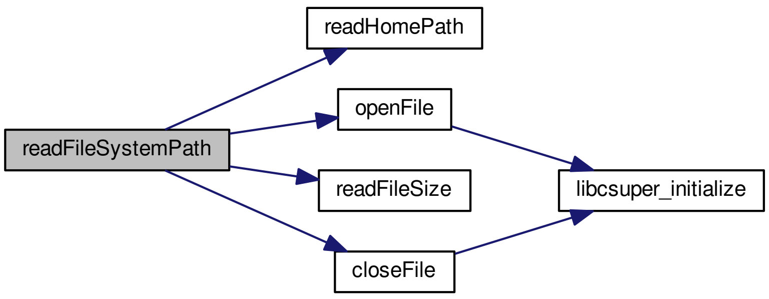
Parameters

`in,out`	∗*file_name*	the filename

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.11.3.10 int readFileToolbarButtonPreferences (char ∗ *home_path,* toolbar_button_preferences_struct ∗ *toolbar*)

Read the file which contain the preferences for the toolbar button

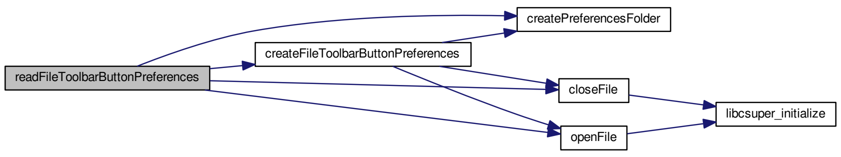
Parameters

in	*home_path*	the path to the home directory
in	*toolbar*	the toolbar button preferences

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:


createPreferencesFolder
createFileToolbarButtonPreferences
readFileToolbarButtonPreferences
closeFile
libcsuper_initialize
openFile

4.11.3.11 void readHomePath (char ∗ *path*)

Read the home path

Parameters

in,out	*path*	the path

Read the home path with a slash at the end

Parameters

in,out	*path*	the path

4.11.3.12 void readHomePathSlash (char ∗ *path*)

4.11.3.13 int readSystemPath (char ∗ *file_name*)

Add the system path, if the file system path doesn't exist, it create it.

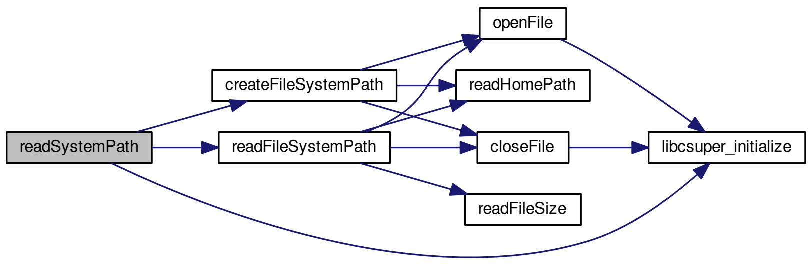
Parameters

in,out	∗*file_name*	the filename

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.12 share.c File Reference

Essential function of libcsuper.

```
#include "share.h"
#include "csu_files.h"
```

Functions

- void libcsuper_initialize ()
- void wrongChoice ()
- void clearScreen ()
- int compareFloatAscending (void const ∗a, void const ∗b)
- int compareFloatDescending (void const ∗a, void const ∗b)
- FILE ∗ openFile (char file_name[], char mode[])
- int closeFile (FILE ∗ptr_file)
- int readFileSize (FILE ∗ptr_file)
- void ∗ myAlloc (int size_alloue)
- void myRealloc (void ∗∗ptr, int size_alloue)
- void addFileCsuExtension (char ∗file_name)
- int deleteFile (char ∗file_name)
- int renameFile (char ∗old_name, char ∗new_name)
- char ∗ integerToYesNo (int i, char ∗yes, char ∗no)
- int getFolderFromFilename (char ∗file_name_to_folder)
- int getSimpleFilenameFromFullFilename (char ∗full_filename, char ∗simple_filename)

4.12.1 Detailed Description

Essential function of libcsuper.

Author

Remi BERTHO

Date

15/06/14

Version

4.0.0

4.12.2 Function Documentation

4.12.2.1 void addFileCsuExtension (char ∗ *file_name*)

Add the csu file extension

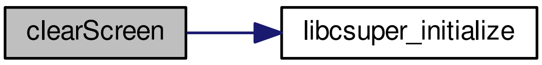
Parameters

`in`	*file_name*	the filename

4.12.2.2 void clearScreen ( )

Clear the terminal.

Here is the call graph for this function:



4.12.2.3 int closeFile (FILE ∗ *ptr_file*)

Close the file

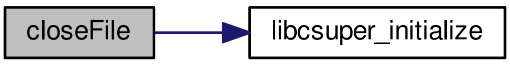
Parameters

`in`	∗*ptr_file*	the file

Returns

0 if everything is OK, 1 otherwise

Here is the call graph for this function:



4.12.2.4 int compareFloatAscending (void const ∗ *a*, void const ∗ *b*)

Compare 2 float

Parameters

in	∗*a*	a pointer on a float
in	∗*b*	a pointer on a float

Returns

1 if a>b, 0 if a=b and -1 if a<b

4.12.2.5 int int compareFloatDescending (void const ∗ *a*, void const ∗ *b*)

Compare 2 float

Parameters

in	∗*a*	a pointer on a float
in	∗*b*	a pointer on a float

Returns

1 if a<b, 0 if a=b and -1 if a>b

4.12.2.6 int deleteFile (char ∗ *file_name*)

Delete a file

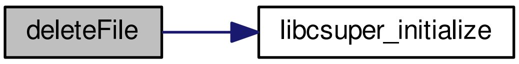
Parameters

in	∗*file_name*	the filename

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.12.2.7 int getFolderFromFilename (char ∗ *file_name_to_folder*)

Transform a filename into his folder

Parameters

in	*file_name_to_-folder*	the filename

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

4.12.2.8 int getSimpleFilenameFromFullFilename (char ∗ *full_filename,* char ∗ *simple_filename*)

Transform a full filename into his simple filename (without the folder)

Parameters

in	*full_filename*	the full filename
in	*simple_filename*	the full filename

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

4.12.2.9 char ∗ integerToYesNo (int *i,* char ∗ *yes,* char ∗ *no*)

Transform an integer to yes or no

Parameters

in	*i*	the integer
in	*yes*	the yes string
in	*no*	the no string

Returns

yes if i $>$ 0, no otherwise

4.12.2.10 void libcsuper_initialize ( )

Initialize libcsuper with gettext.

4.12.2.11 void ∗ myAlloc (int *size_alloue*)

Allocate a memory block and check if everything is OK.

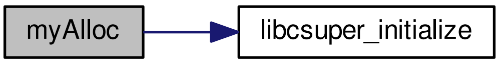
Parameters

in	*size_alloue*	the size

Returns

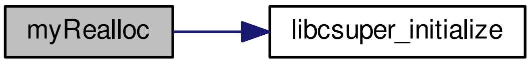
a pointer on the allocate memory block

Here is the call graph for this function:



4.12.2.12 void myRealloc (void ∗∗ *ptr,* int *size_alloue*)

Here is the call graph for this function:



4.12.2.13 FILE ∗ openFile (char *file_name[],* char *mode[]*)

Open a file with his name and with a specific mode.

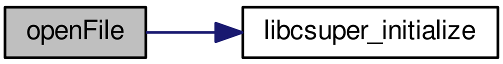
Parameters

`in`	*file_name[]*	the filename
`in`	*mode[]*	the mode

Returns

a pointer to the open file, NULL if there was a problem

Here is the call graph for this function:



4.12.2.14 int readFileSize (FILE ∗ *ptr_file*)

Read the size of the file

Parameters

in	*∗ptr_file*	the file

Returns

the size of the file

4.12.2.15 int renameFile (char ∗ *old_name,* char ∗ *new_name*)

Rename a file.

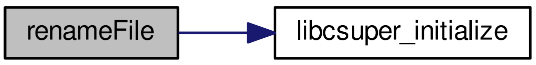
Parameters

in	*∗old_name*	the old name of the file
in	*∗new_name*	the new name of the file

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

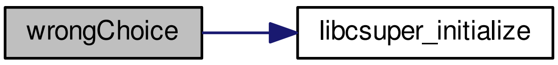
Here is the call graph for this function:



4.12.2.16 void wrongChoice ( )

Display an error message.

Here is the call graph for this function:



4.13 share.h File Reference

Header for the essential function of libcsuper.

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <libintl.h>
```

Macros

- #define MY_TRUE 1
- #define MY_FALSE 0
- #define _(String) dgettext ("libcsuper", String)

Functions

- void libcsuper_initialize ()
- void wrongChoice ()
- void clearScreen ()
- int compareFloatDescending (void const ∗a, void const ∗b)
- int compareFloatAscending (void const ∗a, void const ∗b)
- FILE ∗ openFile (char nome[ ], char mode[ ])
- int closeFile (FILE ∗ptr_file)
- int readFileSize (FILE ∗ptr_file)
- void ∗ myAlloc (int size_alloue)
- void myRealloc (void ∗∗ptr, int size_alloue)
- void addFileCsuExtension (char ∗file_name)
- int deleteFile (char ∗file_name)
- int renameFile (char ∗old_name, char ∗new_name)
- char ∗ integerToYesNo (int i, char ∗yes, char ∗no)
- int getFolderFromFilename (char ∗file_name_to_folder)
- int getSimpleFilenameFromFullFilename (char ∗full_filename, char ∗simple_filename)

4.13.1 Detailed Description

Header for the essential function of libcsuper.

Author

Remi BERTHO

Date

15/06/14

Version

4.0.0

4.13.2 Macro Definition Documentation

4.13.2.1 #define _(*String*) dgettext ("libcsuper", String)

Define the _ for gettext.

4.13.2.2 #define MY_FALSE 0

Definit MY_FALSE a 0

4.13.2.3 #define MY_TRUE 1

Definit MY_TRUE a 1

4.13.3 Function Documentation

4.13.3.1 void addFileCsuExtension (char ∗ *file_name*)

Add the csu file extension

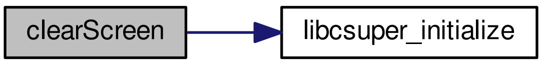
Parameters

in	*file_name*	the filename

4.13.3.2 void clearScreen ( )

Clear the terminal.

Here is the call graph for this function:



4.13.3.3 int closeFile (FILE ∗ *ptr_file*)

Close the file

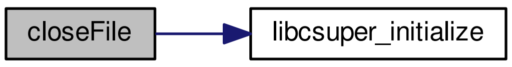
Parameters

in	∗*ptr_file*	the file

Returns

0 if everything is OK, 1 otherwise

Here is the call graph for this function:



4.13.3.4 int compareFloatAscending (void const $*$ *a*, void const $*$ *b*)

Compare 2 float

Parameters

in	$*a$	a pointer on a float
in	$*b$	a pointer on a float

Returns

1 if a>b, 0 if a=b and -1 if a<b

4.13.3.5 int compareFloatDescending (void const $*$ *a*, void const $*$ *b*)

Compare 2 float

Parameters

in	$*a$	a pointer on a float
in	$*b$	a pointer on a float

Returns

1 if a<b, 0 if a=b and -1 if a>b

4.13.3.6 int deleteFile (char $*$ *file_name*)

Delete a file

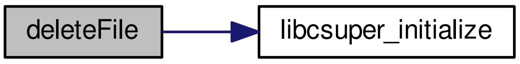
Parameters

in	$*$*file_name*	the filename

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

Here is the call graph for this function:



4.13.3.7 int getFolderFromFilename (char ∗ *file_name_to_folder*)

Transform a filename into his folder

Parameters

in	*file_name_to_-folder*	the filename

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

4.13.3.8 int getSimpleFilenameFromFullFilename (char ∗ *full_filename,* char ∗ *simple_filename*)

Transform a full filename into his simple filename (without the folder)

Parameters

in	*full_filename*	the full filename
in	*simple_filename*	the full filename

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

4.13.3.9 char∗ integerToYesNo (int *i,* char ∗ *yes,* char ∗ *no*)

Transform an integer to yes or no

Parameters

in	*i*	the integer
in	*yes*	the yes string
in	*no*	the no string

Returns

yes if i $>$ 0, no otherwise

4.13.3.10 void libcsuper_initialize ( )

Initialize libcsuper with gettext.

4.13.3.11 void∗ myAlloc (int *size_alloue*)

Allocate a memory block and check if everything is OK.

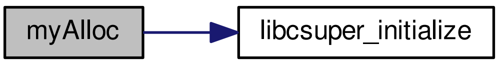
Parameters

in	*size_alloue*	the size

Returns

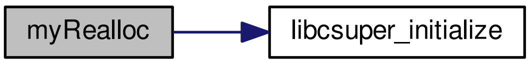
a pointer on the allocate memory block

Here is the call graph for this function:



4.13.3.12 void myRealloc (void ∗∗ *ptr,* int *size_alloue*)

Here is the call graph for this function:



4.13.3.13 FILE∗ openFile (char *file_name[],* char *mode[]*)

Open a file with his name and with a specific mode.

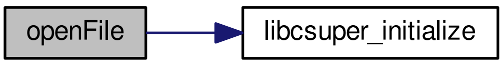
Parameters

in	*file_name[]*	the filename
in	*mode[]*	the mode

Returns

a pointer to the open file, NULL if there was a problem

Here is the call graph for this function:



4.13.3.14 int readFileSize (FILE ∗ *ptr_file*)

Read the size of the file

Parameters

in	∗*ptr_file*	the file

Returns

the size of the file

4.13.3.15 int renameFile (char ∗ *old_name*, char ∗ *new_name*)

Rename a file.

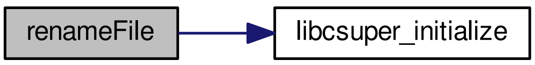
Parameters

in	∗*old_name*	the old name of the file
in	∗*new_name*	the new name of the file

Returns

MY_TRUE if everything is OK, MY_FALSE otherwise

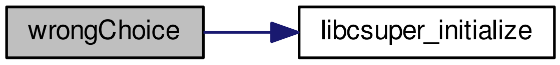
Here is the call graph for this function:



4.13.3.16 void wrongChoice ( )

Display an error message.

Here is the call graph for this function:



Index