calory-counter

1

Generated by Doxygen 1.8.5

Thu Oct 9 2014 11:51:26

Contents

1 Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

2 File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

3 Data Structure Documentation

3.1 client_config Struct Reference

Client config structure.

Data Fields

- char ∗ host
- unsigned int port

3.1.1 Detailed Description

Client config structure.

3.1.2 Field Documentation

3.1.2.1 char∗ client_config::host

Hostname to connect to

3.1.2.2 unsigned int client_config::port

Port to connect to

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

- diet-client.c

3.2 food Struct Reference

food structure for representing a food item

Data Fields

- char name [MAX_NAME_LEN]
- char measure [MAX_NAME_LEN]
- int weight
- int kcal
- int fat
- int carbo
- int protein

3.2.1 Detailed Description

food structure for representing a food item

3.2.2 Field Documentation

3.2.2.1 int food::carbo

Carbo (g) of the food.

3.2.2.2 int food::fat

Fat (g) of the food.

3.2.2.3 int food::kcal

kCal of the food.

3.2.2.4 char food::measure[MAX_NAME_LEN]

Name of the food.

3.2.2.5 char food::name[MAX_NAME_LEN]

Name of the food.

3.2.2.6 int food::protein

Protein (g) of the food.

3.2.2.7 int food::weight

Weight (g) of the food.

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

- food.c

3.3 foodlist Struct Reference

foodlist structure for representing a food item

Data Fields

- pthread_mutex_t rw_mutex
- pthread_mutex_t r_mutex
- int read_count
- foodlistnode ∗ data
- char ∗ file

3.3.1 Detailed Description

foodlist structure for representing a food item

3.3.2 Field Documentation

3.3.2.1 foodlistnode∗ foodlist::data

First node of this list

3.3.2.2 char∗ foodlist::file

Filename for loading/saving data from/to file

3.3.2.3 pthread_mutex_t foodlist::r_mutex

< Mutex for thread safe write access

3.3.2.4 int foodlist::read_count

Integer for thread safe read access

3.3.2.5 pthread_mutex_t foodlist::rw_mutex

Mutex for thread safe read-write access

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

- foodlist.c

3.4 foodlistnode Struct Reference

foodlistnode structure for representing a foodlistnode item

Data Fields

- food ∗ item
- foodlistnode ∗ next

3.4.1 Detailed Description

foodlistnode structure for representing a foodlistnode item

3.4.2 Field Documentation

3.4.2.1 food∗ foodlistnode::item

Item of this foodlistnode

3.4.2.2 foodlistnode∗ foodlistnode::next

Next foodlistnode of this foodlistnode

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

- foodlistnode.c

3.5 sockethandler Struct Reference

sockethandler structure for representing a sockethandler item

Data Fields

- unsigned int listen_port
- pthread_t thread_pool [MAX_THREADS]
- int client_sockets [MAX_SOCKETS]
- bool shutdown
- foodlist ∗ foodlist
- pthread_mutex_t mutex
- sem_t empty
- sem_t full
- size_t in
- size_t out
- size_t count

3.5.1 Detailed Description

sockethandler structure for representing a sockethandler item

3.5.2 Field Documentation

3.5.2.1 int sockethandler::client_sockets[MAX_SOCKETS]

Array of client sockets for consumer/producer principle

3.5.2.2 size_t sockethandler::count

number of unconsumed items

3.5.2.3 sem_t sockethandler::empty

Semaphore to block on empty socket list.

3.5.2.4 foodlist∗ sockethandler::foodlist

List of foods to work with

3.5.2.5 sem_t sockethandler::full

Semaphore to block on full socket list.

3.5.2.6 size_t sockethandler::in

position of producer in buffer

3.5.2.7 unsigned int sockethandler::listen_port

Listen port for the server socket

3.5.2.8 pthread_mutex_t sockethandler::mutex

Mutex to mutual exclude the client_socket array.

3.5.2.9 size_t sockethandler::out

position of consumer in buffer

3.5.2.10 bool sockethandler::shutdown

Flag to notifying all threads to shut down

3.5.2.11 pthread_t sockethandler::thread_pool[MAX_THREADS]

Thread pool for handling client connections

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

- sockethandler.c

4 File Documentation

4.1 diet-client.c File Reference

Main program file with main() entry point.

```
#include <pthread.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <ctype.h>
#include <unistd.h>
#include <errno.h>
#include "../lib/sock.h"
#include "../lib/food.h"
```

Data Structures

- struct client_config

  Client config structure.

Typedefs

- typedef struct client_config client_config

  Forward declaration of client_config.

Functions

- void usage (char ∗pname)

  Prints the help for diet-client to the console.
- bool ask_user_cancel ()

  Method for getting a true/false user input.
- food ∗ get_food_from_user ()

  Method for creating a food object with help of user input.
- void client_loop (client_config ∗c)

  Loop function with handles the client connection and user input stuff.
- int main (int argc, char ∗∗argv)

  Main entry point of diet-client.

Variables

- bool client_exit = false

  This is set to exit when the application should exit gracefully.

4.1.1 Detailed Description

Main program file with main() entry point.

Author

Lukas Elsner

Date

01-09-2014 The calory-counter client application.

4.1.2 Function Documentation

4.1.2.1 bool ask_user_cancel ( )

Method for getting a true/false user input.

Returns

True, if the user decided to cancel, false if the user wants to continue

4.1.2.2 void client_loop (client_config ∗ *c*)

Loop function with handles the client connection and user input stuff.

Parameters

client_config∗	A pointer to the client configuration

4.1.2.3 food∗ get_food_from_user ( )

Method for creating a food object with help of user input.

Returns

A pointer to a food structure if the user entered all data correctly, NULL if the user decided to cancel the operation.

4.1.2.4 int main (int *argc*, char ∗∗ *argv*)

Main entry point of diet-client.

Parameters

int	Number of arguments
char∗∗	Pointer to array of arguments

Returns

Exit code of diet-client

4.1.2.5 void usage (char ∗ *pname*)

Prints the help for diet-client to the console.

Parameters

char∗	Program name

4.2 diet-server.c File Reference

Main program file with main() entry point.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include "sockethandler.h"
```

Functions

- void usage (char ∗pname)

  Prints the help for diet-server to the console.
- void signal_callback_handler (int signum)

  Define the function to be called when ctrl-c (SIGINT) signal is sent to process.
- int main (int argc, char ∗∗argv)

  Main entry point of airport-sim.

Variables

- foodlist ∗ fl

  Representation of the food list.
- sockethandler ∗ s

  Representation of the socket handler.

4.2.1 Detailed Description

Main program file with main() entry point.

Author

Lukas Elsner

Date

01-09-2014 The calory-counter server application.

4.2.2 Function Documentation

4.2.2.1 int main (int *argc*, char ∗∗ *argv*)

Main entry point of airport-sim.

Parameters

int	Number of arguments
char∗∗	Pointer to array of arguments

Returns

Exit code of airport-sim

4.2.2.2 void usage (char ∗ *pname*)

Prints the help for diet-server to the console.

Parameters

char∗	Program name

4.3 food.c File Reference

File containing the food structure and its member methods.

```
#include <time.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "food.h"
```

Data Structures

- struct food

  food structure for representing a food item

Functions

- food ∗ food_init ()

  Constructor for food.
- void food_set_name (food ∗f, const char ∗name)

  Method for setting the name of a food structure.
- char ∗ food_get_name (food ∗f)

  Method for getting the name of a food structure.
- void food_set_measure (food ∗f, const char ∗measure)

  Method for setting the measure of a food structure.
- char ∗ food_get_measure (food ∗f)

  Method for getting the measure of a food structure.
- char ∗ food_to_string (food ∗f)

  Method for getting a string representation of a food structure.
- void food_set_weight (food ∗f, const int weight)

  Method for setting the weight of a food structure.
- int food_get_weight (food ∗f)

  Method for getting the weight of a food structure.
- void food_set_kcal (food ∗f, const int kcal)

  Method for setting the kCal of a food structure.
- int food_get_kcal (food ∗f)

  Method for getting the kCal of a food structure.
- void food_set_fat (food ∗f, const int fat)

  Method for setting the fat of a food structure.
- int food_get_fat (food ∗f)

  Method for getting the fat of a food structure.
- void food_set_carbo (food ∗f, const int carbo)

  Method for setting the carbo of a food structure.
- int food_get_carbo (food ∗f)

  Method for getting the carbo of a food structure.
- void food_set_protein (food ∗f, const int protein)

  Method for setting the protein of a food structure.
- int food_get_protein (food ∗f)

  Method for getting the protein of a food structure.
- char ∗ food_serialize (food ∗f)

  Method for serializing a food structure into a character array.
- food ∗ food_deserialize (char ∗c)

  Method for deserializing a char array into a food structure, this method is being used for loading foods from the csv file, as well as for the network communication.
- size_t food_get_size ()

  Method for getting the size of a food structure.
- void food_destroy (food ∗f)

  Destructor for food.

4.3.1 Detailed Description

File containing the food structure and its member methods.

Author

Lukas Elsner

Date

01-09-2014

4.3.2 Function Documentation

4.3.2.1 food∗ food_deserialize (char ∗)

Method for deserializing a char array into a food structure, this method is being used for loading foods from the csv file, as well as for the network communication.

Parameters

char∗	The serialized structure

Returns

A pointer to the deserialized structure. Must be freed with food_destroy(food ∗)

4.3.2.2 void food_destroy (food ∗)

Destructor for food.

Parameters

food∗	Pointer to structure to be freed

4.3.2.3 int food_get_carbo (food ∗)

Method for getting the carbo of a food structure.

Parameters

food∗	Pointer to structure to work on

Returns

Carbo value in g of the food

4.3.2.4 int food_get_fat (food ∗)

Method for getting the fat of a food structure.

Parameters

food∗	Pointer to structure to work on

Returns

Fat value in g of the food

4.3.2.5 int food_get_kcal (food ∗)

Method for getting the kCal of a food structure.

Parameters

food∗	Pointer to structure to work on

Returns

kCal value of the food

4.3.2.6 char∗ food_get_measure (food ∗)

Method for getting the measure of a food structure.

Parameters

food∗	Pointer to structure to work on

Returns

A pointer to a string containing the measure of the food. Must be freed by caller.

4.3.2.7 char∗ food_get_name (food ∗)

Method for getting the name of a food structure.

Parameters

food∗	Pointer to structure to work on

Returns

A pointer to a string containing the name of the food. Must be freed by caller.

4.3.2.8 int food_get_protein (food ∗)

Method for getting the protein of a food structure.

Parameters

food∗	Pointer to structure to work on

Returns

Protein value in g of the food

4.3.2.9 size_t food_get_size ( )

Method for getting the size of a food structure.

Returns

Size of a food structure

4.3.2.10 int food_get_weight (food ∗)

Method for getting the weight of a food structure.

Parameters

food∗	Pointer to structure to work on

Returns

Weight value in g of the food

4.3.2.11 food∗ food_init ( )

Constructor for food.

Returns

A pointer to the food structure, representing the created object

After using this structure, it must be freed with food_destroy(food ∗)

4.3.2.12 char* food_serialize (food *)

Method for serializing a food structure into a character array.

Parameters

food∗	Pointer to structure to work on

Returns

A pointer to the serialized structure. Must be freed by caller

4.3.2.13 void food_set_carbo (food ∗ , const int)

Method for setting the carbo of a food structure.

Parameters

food∗	Pointer to structure to work on
int	Carbo value in g to be set

4.3.2.14 void food_set_fat (food ∗ , const int)

Method for setting the fat of a food structure.

Parameters

food∗	Pointer to structure to work on
int	Fat value in g to be set

4.3.2.15 void food_set_kcal (food ∗ , const int)

Method for setting the kCal of a food structure.

Parameters

food∗	Pointer to structure to work on
int	kCal value to be set

4.3.2.16 void food_set_measure (food ∗ , const char ∗)

Method for setting the measure of a food structure.

Parameters

food∗	Pointer to structure to work on
char∗	Pointer to character array containing the measure to be set

4.3.2.17 void food_set_name (food ∗ , const char ∗)

Method for setting the name of a food structure.

Parameters

food∗	Pointer to structure to work on
char∗	Pointer to character array containing the name to be set

4.3.2.18 void food_set_protein (food ∗ , const int)

Method for setting the protein of a food structure.

Parameters

food∗	Pointer to structure to work on
int	Protein value in g to be set

4.3.2.19 void food_set_weight (**food** ∗ , const int)

Method for setting the weight of a food structure.

Parameters

food∗	Pointer to structure to work on
int	Weight value in g to be set

4.3.2.20 char∗ food_to_string (**food** ∗)

Method for getting a string representation of a food structure.

Parameters

food∗	Pointer to structure to work on

Returns

String representation of food structure. Must be freed by caller.

4.4 food.h File Reference

Header containing the public accessible food methods.

Macros

- #define **MAX_NAME_LEN** 1024
- #define **MAX_MEASURE_LEN** 256

Typedefs

- typedef struct food food

  Forward declaration for food.

Functions

- food ∗ food_init ()

  Constructor for food.
- char ∗ food_serialize (food ∗)

  Method for serializing a food structure into a character array.
- food ∗ food_deserialize (char ∗)

  Method for deserializing a char array into a food structure, this method is being used for loading foods from the csv file, as well as for the network communication.
- void food_set_name (food ∗, const char ∗)

  Method for setting the name of a food structure.
- char ∗ food_get_name (food ∗)

  Method for getting the name of a food structure.
- void food_set_measure (food ∗, const char ∗)

  Method for setting the measure of a food structure.

- char ∗ food_get_measure (food ∗)

    Method for getting the measure of a food structure.
- void food_set_weight (food ∗, const int)

    Method for setting the weight of a food structure.
- int food_get_weight (food ∗)

    Method for getting the weight of a food structure.
- void food_set_kcal (food ∗, const int)

    Method for setting the kCal of a food structure.
- int food_get_kcal (food ∗)

    Method for getting the kCal of a food structure.
- void food_set_fat (food ∗, const int)

    Method for setting the fat of a food structure.
- int food_get_fat (food ∗)

    Method for getting the fat of a food structure.
- void food_set_carbo (food ∗, const int)

    Method for setting the carbo of a food structure.
- int food_get_carbo (food ∗)

    Method for getting the carbo of a food structure.
- void food_set_protein (food ∗, const int)

    Method for setting the protein of a food structure.
- int food_get_protein (food ∗)

    Method for getting the protein of a food structure.
- char ∗ food_to_string (food ∗)

    Method for getting a string representation of a food structure.
- size_t food_get_size ()

    Method for getting the size of a food structure.
- void food_destroy (food ∗)

    Destructor for food.

4.4.1 Detailed Description

Header containing the public accessible food methods.

Author

Lukas Elsner

Date

25-09-2014

4.4.2 Function Documentation

4.4.2.1 food∗ food_deserialize (char ∗)

Method for deserializing a char array into a food structure, this method is being used for loading foods from the csv file, as well as for the network communication.

Parameters

char∗	The serialized structure

Returns

A pointer to the deserialized structure. Must be freed with food_destroy(food ∗)

4.4.2.2 void food_destroy (food ∗)

Destructor for food.

Parameters

food∗	Pointer to structure to be freed

4.4.2.3 int food_get_carbo (food ∗)

Method for getting the carbo of a food structure.

Parameters

food∗	Pointer to structure to work on

Returns

Carbo value in g of the food

4.4.2.4 int food_get_fat (food ∗)

Method for getting the fat of a food structure.

Parameters

food∗	Pointer to structure to work on

Returns

Fat value in g of the food

4.4.2.5 int food_get_kcal (food ∗)

Method for getting the kCal of a food structure.

Parameters

food∗	Pointer to structure to work on

Returns

kCal value of the food

4.4.2.6 char∗ food_get_measure (food ∗)

Method for getting the measure of a food structure.

Parameters

food∗	Pointer to structure to work on

Returns

A pointer to a string containing the measure of the food. Must be freed by caller.

4.4.2.7 char∗ food_get_name (food ∗)

Method for getting the name of a food structure.

Parameters

food∗	Pointer to structure to work on

Returns

A pointer to a string containing the name of the food. Must be freed by caller.

4.4.2.8 int food_get_protein (food ∗)

Method for getting the protein of a food structure.

Parameters

food∗	Pointer to structure to work on

Returns

Protein value in g of the food

4.4.2.9 size_t food_get_size ( )

Method for getting the size of a food structure.

Returns

Size of a food structure

4.4.2.10 int food_get_weight (food ∗)

Method for getting the weight of a food structure.

Parameters

food∗	Pointer to structure to work on

Returns

Weight value in g of the food

4.4.2.11 food∗ food_init ( )

Constructor for food.

Returns

A pointer to the food structure, representing the created object

After using this structure, it must be freed with food_destroy(food ∗)

4.4.2.12 char∗ food_serialize (food ∗)

Method for serializing a food structure into a character array.

Parameters

food∗	Pointer to structure to work on

Returns

A pointer to the serialized structure. Must be freed by caller

4.4.2.13 void food_set_carbo (food ∗ , const int)

Method for setting the carbo of a food structure.

Parameters

food∗	Pointer to structure to work on
int	Carbo value in g to be set

4.4.2.14 void food_set_fat (food ∗ , const int)

Method for setting the fat of a food structure.

Parameters

food∗	Pointer to structure to work on
int	Fat value in g to be set

4.4.2.15 void food_set_kcal (food ∗ , const int)

Method for setting the kCal of a food structure.

Parameters

food∗	Pointer to structure to work on
int	kCal value to be set

4.4.2.16 void food_set_measure (food ∗ , const char ∗)

Method for setting the measure of a food structure.

Parameters

food∗	Pointer to structure to work on
char∗	Pointer to character array containing the measure to be set

4.4.2.17 void food_set_name (food ∗ , const char ∗)

Method for setting the name of a food structure.

Parameters

food∗	Pointer to structure to work on
char∗	Pointer to character array containing the name to be set

4.4.2.18 void food_set_protein (food ∗ , const int)

Method for setting the protein of a food structure.

Parameters

food∗	Pointer to structure to work on
int	Protein value in g to be set

4.4.2.19 void food_set_weight (**food** ∗ , const int)

Method for setting the weight of a food structure.

Parameters

food∗	Pointer to structure to work on
int	Weight value in g to be set

4.4.2.20 char∗ food_to_string (**food** ∗)

Method for getting a string representation of a food structure.

Parameters

food∗	Pointer to structure to work on

Returns

String representation of food structure. Must be freed by caller.

4.5 foodlist.c File Reference

File containing the foodlist structure and its member methods.

```
#include <stdlib.h>
#include <stdbool.h>
#include <stdio.h>
#include <string.h>
#include <strings.h>
#include <pthread.h>
#include "food.h"
#include "foodlistnode.h"
#include "foodlist.h"
```

Data Structures

- struct foodlist

  foodlist structure for representing a food item

Functions

- void start_read (foodlist ∗fl)

  Helper function to enter a critical section for reading.
- void end_read (foodlist ∗fl)

  Helper function to exit a critical section for reading.
- void start_write (foodlist ∗fl)

  Helper function to enter a critical section for writing.
- void end_write (foodlist ∗fl)

  Helper function to exit a critical section for writing.

- int cmpfunc (const void ∗a, const void ∗b)

  Compare function for using qsort() with food objects.
- foodlist ∗ foodlist_init ()

  Constructor for foodlist.
- foodlist ∗ foodlist_init_csv (char ∗file)

  Constructor for foodlist, loads a csv file which is passed as argument, throws a warning if file cannot be read and starts with empty dataset then.
- int foodlist_count (foodlist ∗fl)

  Method for getting the length of the list.
- bool foodlist_is_empty (foodlist ∗fl)

  Method for checking if the foodlist is empty.
- void foodlist_append (foodlist ∗fl, food ∗∗f)

  Method for appending a food structure to the list.
- foodlistnode ∗ foodlist_get_data (foodlist ∗fl)

  Method for getting the data of the list.
- food ∗∗ foodlist_find (foodlist ∗fl, char ∗str, size_t ∗num)

  Method for finding food within the food list.
- void foodlist_save (foodlist ∗fl)

  Method for saving the food structure to a file.
- void foodlist_destroy (foodlist ∗fl)

  Destructor for foodlist.

4.5.1 Detailed Description

File containing the foodlist structure and its member methods.

Author

Lukas Elsner

Date

25-09-2014

4.5.2 Function Documentation

4.5.2.1 int cmpfunc (const void ∗ *a*, const void ∗ *b*)

Compare function for using qsort() with food objects.

Parameters

void∗	Pointer to first food object
void∗	Pointer to second food object

Returns

An integer less than, equal to, or greater than zero if a (or the first n bytes thereof) is found, respectively, to be less than, to match, or be greater than b

4.5.2.2 void end_read (foodlist ∗ *fl*)

Helper function to exit a critical section for reading.

Parameters

foodlist∗	The foodlist structure to unlock

4.5.2.3 void end_write (**foodlist** ∗ *fl*)

Helper function to exit a critical section for writing.

Parameters

foodlist∗	The foodlist structure to unlock

4.5.2.4 void foodlist_append (**foodlist** ∗ , **food** ∗∗)

Method for appending a food structure to the list.

Parameters

foodlist∗	Pointer to structure to work on
food∗∗	pointer to pointer to food structure to add

4.5.2.5 int foodlist_count (**foodlist** ∗)

Method for getting the length of the list.

Parameters

foodlist∗	Pointer to structure to work on

Returns

Length of the list

4.5.2.6 void foodlist_destroy (**foodlist** ∗)

Destructor for foodlist.

Parameters

foodlist∗	Pointer to structure to be freed

4.5.2.7 **food**∗∗ foodlist_find (**foodlist** ∗ , char ∗ , size_t ∗)

Method for finding food within the food list.

Parameters

foodlist∗	Pointer to structure to work on
char∗	A pointer to the string which should be found
size_t∗	Pointer to a size_t instance. The method updates its value to the length of the returned list.

Returns

food∗∗ A pointer to an array of food pointers, which are satisfying the search criteria.

4.5.2.8 **foodlistnode**∗ foodlist_get_data (**foodlist** ∗)

Method for getting the data of the list.

Parameters

foodlist∗	Pointer to structure to work on

Returns

First node of the list.

4.5.2.9 **foodlist**∗ foodlist_init ( )

Constructor for foodlist.

Returns

A pointer to the foodlist structure, representing the created object

After using this structure, it must be freed with foodlist_destroy(foodlist ∗)

4.5.2.10 **foodlist**∗ foodlist_init_csv (char ∗)

Constructor for foodlist, loads a csv file which is passed as argument, throws a warning if file cannot be read and starts with empty dataset then.

Parameters

char∗	Filename to the csv-file to be loaded.

Returns

A pointer to the foodlist structure, representing the created object

After using this structure, it must be freed with foodlist_destroy(foodlist ∗)

4.5.2.11 bool foodlist_is_empty (**foodlist** ∗)

Method for checking if the foodlist is empty.

Parameters

foodlist∗	Pointer to structure to work on

Returns

True, if the foodlist is empty, false otherwise

4.5.2.12 void foodlist_save (**foodlist** ∗)

Method for saving the food structure to a file.

Parameters

foodlist∗	Pointer to structure to work on

Before saving, the foodlist is being sorted by the name of the foods. If the file does not exist, it will be created.

4.5.2.13 void start_read (**foodlist** ∗ *fl*)

Helper function to enter a critical section for reading.

Parameters

foodlist∗	The foodlist structure to lock

4.5.2.14 void start_write (foodlist ∗ *fl*)

Helper function to enter a critical section for writing.

Parameters

foodlist∗	The foodlist structure to lock

4.6 foodlist.h File Reference

Header containing the public accessible foodlist methods.

```
#include "food.h"
#include "foodlistnode.h"
```

Typedefs

- typedef struct foodlist foodlist

  Forward declaration for foodlist.

Functions

- foodlist ∗ foodlist_init ()

  Constructor for foodlist.
- foodlist ∗ foodlist_init_csv (char ∗)

  Constructor for foodlist, loads a csv file which is passed as argument, throws a warning if file cannot be read and starts with empty dataset then.
- void foodlist_append (foodlist ∗, food ∗∗)

  Method for appending a food structure to the list.
- food ∗∗ foodlist_find (foodlist ∗, char ∗, size_t ∗)

  Method for finding food within the food list.
- void foodlist_save (foodlist ∗)

  Method for saving the food structure to a file.
- int foodlist_count (foodlist ∗)

  Method for getting the length of the list.
- foodlistnode ∗ foodlist_get_data (foodlist ∗)

  Method for getting the data of the list.
- bool foodlist_is_empty (foodlist ∗)

  Method for checking if the foodlist is empty.
- void foodlist_destroy (foodlist ∗)

  Destructor for foodlist.

4.6.1 Detailed Description

Header containing the public accessible foodlist methods.

Author

Lukas Elsner

Date

25-09-2014

4.6.2 Function Documentation

4.6.2.1 void foodlist_append (foodlist ∗ , food ∗∗)

Method for appending a food structure to the list.

Parameters

foodlist∗	Pointer to structure to work on
food∗∗	pointer to pointer to food structure to add

4.6.2.2 int foodlist_count (foodlist ∗)

Method for getting the length of the list.

Parameters

foodlist∗	Pointer to structure to work on

Returns

Length of the list

4.6.2.3 void foodlist_destroy (foodlist ∗)

Destructor for foodlist.

Parameters

foodlist∗	Pointer to structure to be freed

4.6.2.4 food∗∗ foodlist_find (foodlist ∗ , char ∗ , size_t ∗)

Method for finding food within the food list.

Parameters

foodlist∗	Pointer to structure to work on
char∗	A pointer to the string which should be found
size_t∗	Pointer to a size_t instance. The method updates its value to the length of the returned list.

Returns

food∗∗ A pointer to an array of food pointers, which are satisfying the search criteria.

4.6.2.5 foodlistnode∗ foodlist_get_data (foodlist ∗)

Method for getting the data of the list.

Parameters

foodlist∗	Pointer to structure to work on

Returns

First node of the list.

4.6.2.6 **foodlist**∗ foodlist_init ( )

Constructor for foodlist.

Returns

A pointer to the foodlist structure, representing the created object

After using this structure, it must be freed with foodlist_destroy(foodlist ∗)

4.6.2.7 **foodlist**∗ foodlist_init_csv (char ∗)

Constructor for foodlist, loads a csv file which is passed as argument, throws a warning if file cannot be read and starts with empty dataset then.

Parameters

char∗	Filename to the csv-file to be loaded.

Returns

A pointer to the foodlist structure, representing the created object

After using this structure, it must be freed with foodlist_destroy(foodlist ∗)

4.6.2.8 bool foodlist_is_empty (**foodlist** ∗)

Method for checking if the foodlist is empty.

Parameters

foodlist∗	Pointer to structure to work on

Returns

True, if the foodlist is empty, false otherwise

4.6.2.9 void foodlist_save (**foodlist** ∗)

Method for saving the food structure to a file.

Parameters

foodlist∗	Pointer to structure to work on

Before saving, the foodlist is being sorted by the name of the foods. If the file does not exist, it will be created.

4.7 foodlistnode.c File Reference

File containing the foodlistnode structure and its member methods.

```
#include <time.h>
#include <stdlib.h>
#include <assert.h>
#include "foodlistnode.h"
```

Data Structures

- struct foodlistnode

  foodlistnode structure for representing a foodlistnode item

Functions

- foodlistnode ∗ foodlistnode_init ()

  constructor for foodlistnode
- foodlistnode ∗ foodlistnode_get_next (foodlistnode ∗fln)

  Method for getting the next foodlistnode of a foodlistnode structure.
- food ∗ foodlistnode_get_item (foodlistnode ∗fln)

  Method for getting the item of a foodlistnode structure.
- void foodlistnode_set_next (foodlistnode ∗fln, foodlistnode ∗∗f)

  Method for setting the next node of a node.
- void foodlistnode_set_item (foodlistnode ∗fln, food ∗∗f)

  Method for setting the item of a node.
- bool foodlistnode_has_next (foodlistnode ∗fln)

  Method for checking if a foodlistnode has a next element.
- int foodlistnode_count (foodlistnode ∗fln)

  Recursive method for getting the number of items in this foodlistnode structure.
- void foodlistnode_destroy (foodlistnode ∗fln)

  Destructor for foodlistnode.

4.7.1 Detailed Description

File containing the foodlistnode structure and its member methods.

Author

Lukas Elsner

Date

01-09-2014

4.7.2 Function Documentation

4.7.2.1 int foodlistnode_count (**foodlistnode** ∗)

Recursive method for getting the number of items in this foodlistnode structure.

Parameters

foodlistnode∗	Pointer to structure to work on

Returns

The number of items in this foodlistnode structure

4.7.2.2 void foodlistnode_destroy (**foodlistnode** ∗)

Destructor for foodlistnode.

Parameters

foodlistnode∗	Pointer to structure to be freed

4.7.2.3 food∗ foodlistnode_get_item (foodlistnode ∗ *fln*)

Method for getting the item of a foodlistnode structure.

Parameters

foodlistnode∗	Pointer to structure to work on

Returns

Item of the passed foodlistnode

4.7.2.4 foodlistnode∗ foodlistnode_get_next (foodlistnode ∗ *fln*)

Method for getting the next foodlistnode of a foodlistnode structure.

Parameters

foodlistnode∗	Pointer to structure to work on

Returns

Next foodlistnode of the passed foodlistnode

4.7.2.5 bool foodlistnode_has_next (foodlistnode ∗)

Method for checking if a foodlistnode has a next element.

Parameters

foodlistnode∗	Pointer to structure to work on

Returns

True, if passed foodlistnode has a next element, false otherwise

4.7.2.6 foodlistnode∗ foodlistnode_init ( )

constructor for foodlistnode

Returns

A pointer to the foodlistnode structure, representing the created object

After using this structure, it must be freed with foodlistnode_destroy(foodlistnode ∗)

4.7.2.7 void foodlistnode_set_item (foodlistnode ∗ *fln,* food ∗∗ *f*)

Method for setting the item of a node.

Parameters

foodlistnode∗	Pointer to structure to work on

foodlistnode∗∗	Pointer to pointer to food to set

4.7.2.8 void foodlistnode_set_next (**foodlistnode** ∗ *fln,* **foodlistnode** ∗∗ *f*)

Method for setting the next node of a node.

Parameters

foodlistnode∗	Pointer to structure to work on
foodlistnode∗∗	Pointer to pointer to foodlistnode to add

4.8 foodlistnode.h File Reference

Header containing the public accessible foodlistnode methods.

```
#include <stdbool.h>
#include "food.h"
```

Typedefs

- typedef struct foodlistnode foodlistnode

  Forward declaration for foodlistnode.

Functions

- foodlistnode ∗ foodlistnode_init ()

  constructor for foodlistnode
- void foodlistnode_set_next (foodlistnode ∗fln, foodlistnode ∗∗f)

  Method for setting the next node of a node.
- foodlistnode ∗ foodlistnode_get_next (foodlistnode ∗fln)

  Method for getting the next foodlistnode of a foodlistnode structure.
- void foodlistnode_set_item (foodlistnode ∗fln, food ∗∗f)

  Method for setting the item of a node.
- food ∗ foodlistnode_get_item (foodlistnode ∗fln)

  Method for getting the item of a foodlistnode structure.
- int foodlistnode_count (foodlistnode ∗)

  Recursive method for getting the number of items in this foodlistnode structure.
- bool foodlistnode_has_next (foodlistnode ∗)

  Method for checking if a foodlistnode has a next element.
- void foodlistnode_destroy (foodlistnode ∗)

  Destructor for foodlistnode.

4.8.1 Detailed Description

Header containing the public accessible foodlistnode methods.

Author

Lukas Elsner

Date

25-09-2014

4.8.2 Function Documentation

4.8.2.1 int foodlistnode_count (**foodlistnode** ∗)

Recursive method for getting the number of items in this foodlistnode structure.

Parameters

foodlistnode∗	Pointer to structure to work on

Returns

The number of items in this foodlistnode structure

4.8.2.2 void foodlistnode_destroy (**foodlistnode** ∗)

Destructor for foodlistnode.

Parameters

foodlistnode∗	Pointer to structure to be freed

4.8.2.3 **food**∗ foodlistnode_get_item (**foodlistnode** ∗ *fln*)

Method for getting the item of a foodlistnode structure.

Parameters

foodlistnode∗	Pointer to structure to work on

Returns

Item of the passed foodlistnode

4.8.2.4 **foodlistnode**∗ foodlistnode_get_next (**foodlistnode** ∗ *fln*)

Method for getting the next foodlistnode of a foodlistnode structure.

Parameters

foodlistnode∗	Pointer to structure to work on

Returns

Next foodlistnode of the passed foodlistnode

4.8.2.5 bool foodlistnode_has_next (**foodlistnode** ∗)

Method for checking if a foodlistnode has a next element.

Parameters

foodlistnode∗	Pointer to structure to work on

Returns

True, if passed foodlistnode has a next element, false otherwise

4.8.2.6 **foodlistnode**∗ foodlistnode_init ( )

constructor for foodlistnode

Returns

A pointer to the foodlistnode structure, representing the created object

After using this structure, it must be freed with foodlistnode_destroy(foodlistnode ∗)

4.8.2.7 void foodlistnode_set_item (**foodlistnode** ∗ *fln,* **food** ∗∗ *f*)

Method for setting the item of a node.

Parameters

foodlistnode∗	Pointer to structure to work on
foodlistnode∗∗	Pointer to pointer to food to set

4.8.2.8 void foodlistnode_set_next (**foodlistnode** ∗ *fln,* **foodlistnode** ∗∗ *f*)

Method for setting the next node of a node.

Parameters

foodlistnode∗	Pointer to structure to work on
foodlistnode∗∗	Pointer to pointer to foodlistnode to add

4.9 sock.c File Reference

File containing read and write functions for calory socket protocol.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <assert.h>
#include <string.h>
#include "sock.h"
```

Functions

- bool sock_write (int socket, char ∗data)

  Lower level function to send data to the other endpoint.
- size_t sock_read (int socket, char ∗data)

  Function to read data from the other endpoint.
- bool sock_send_food (int socket, char ∗data)

  Higher level function to send serialized food to the other endpoint.
- bool sock_send_search (int socket, char ∗data)

  Higher level function to send a search request to the other endpoint.
- bool sock_send_count (int socket, char ∗data)

  Higher level function to send the number of found items to the other endpoint.

4.9.1 Detailed Description

File containing read and write functions for calory socket protocol.

Author

Lukas Elsner

Date

02-09-2014

4.9.2 Function Documentation

4.9.2.1 size_t sock_read (int *socket,* char ∗ *data*)

Function to read data from the other endpoint.

Parameters

int	The socket to communicate with
char∗	A pointer to a buffer which for the read data. Must be at least 4096 bytes long.

Returns

The size of read data

4.9.2.2 bool sock_send_count (int *socket,* char ∗ *data*)

Higher level function to send the number of found items to the other endpoint.

Parameters

int	The socket to communicate with
char∗	The number of found items as string value

Returns

True, if the communication was successful, false otherwise

4.9.2.3 bool sock_send_food (int *socket,* char ∗ *data*)

Higher level function to send serialized food to the other endpoint.

Parameters

int	The socket to communicate with
char∗	The serialized food to send

Returns

True, if the communication was successful, false otherwise

4.9.2.4 bool sock_send_search (int *socket,* char ∗ *data*)

Higher level function to send a search request to the other endpoint.

Parameters

int	The socket to communicate with
char∗	The search term to send

Returns

True, if the communication was successful, false otherwise

4.9.2.5 bool sock_write (int *socket,* char ∗ *data*)

Lower level function to send data to the other endpoint.

Parameters

int	The socket to communicate with
char∗	The data to send

Returns

True, if the communication was successful, false otherwise

4.10 sock.h File Reference

Header file containing read and write functions for calory socket protocol.

```
#include <stdbool.h>
```

Macros

- #define **BUF_LEN** 4096
- #define **RE_LEN** 32

Functions

- bool sock_write (int socket, char ∗data)

  Lower level function to send data to the other endpoint.
- size_t sock_read (int socket, char ∗data)

  Function to read data from the other endpoint.
- bool sock_send_food (int socket, char ∗data)

  Higher level function to send serialized food to the other endpoint.
- bool sock_send_search (int socket, char ∗data)

  Higher level function to send a search request to the other endpoint.
- bool sock_send_count (int socket, char ∗data)

  Higher level function to send the number of found items to the other endpoint.

4.10.1 Detailed Description

Header file containing read and write functions for calory socket protocol.

Author

Lukas Elsner

Date

02-09-2014 Every write is BUF_LEN bytes long and has to be acknowledged with a RE_LEN bytes long answer containing ACK or NACK.

4.10.2 Function Documentation

4.10.2.1 size_t sock_read (int *socket,* char ∗ *data*)

Function to read data from the other endpoint.

Parameters

int	The socket to communicate with
char∗	A pointer to a buffer which for the read data. Must be at least 4096 bytes long.

Returns

The size of read data

4.10.2.2 bool sock_send_count (int *socket,* char ∗ *data*)

Higher level function to send the number of found items to the other endpoint.

Parameters

int	The socket to communicate with
char∗	The number of found items as string value

Returns

True, if the communication was successful, false otherwise

4.10.2.3 bool sock_send_food (int *socket,* char ∗ *data*)

Higher level function to send serialized food to the other endpoint.

Parameters

int	The socket to communicate with
char∗	The serialized food to send

Returns

True, if the communication was successful, false otherwise

4.10.2.4 bool sock_send_search (int *socket,* char ∗ *data*)

Higher level function to send a search request to the other endpoint.

Parameters

int	The socket to communicate with
char∗	The search term to send

Returns

True, if the communication was successful, false otherwise

4.10.2.5 bool sock_write (int *socket,* char ∗ *data*)

Lower level function to send data to the other endpoint.

Parameters

int	The socket to communicate with
char∗	The data to send

Returns

True, if the communication was successful, false otherwise

4.11 sockethandler.c File Reference

File containing the sockethandler structure and its member methods.

```
#include <time.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <stdio.h>
#include <stdbool.h>
#include <sys/select.h>
#include <semaphore.h>
#include <pthread.h>
#include <assert.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include "../lib/sock.h"
#include "../lib/food.h"
#include "../lib/foodlist.h"
#include "sockethandler.h"
```

Data Structures

- struct sockethandler

  sockethandler structure for representing a sockethandler item

Macros

- #define MAX_THREADS 10
- #define MAX_SOCKETS 5

Functions

- void sockethandler_client_thread_func (sockethandler ∗s)

  Method for client connection handling.
- sockethandler ∗ sockethandler_init (foodlist ∗fl)

  Constructor for sockethandler.
- void sockethandler_server_thread_func (sockethandler ∗s)

  Main loop function for the sockethandling procedure.
- void sockethandler_set_port (sockethandler ∗s, int port)

  Method for setting the listening port of a sockethandler structure.
- void sockethandler_shutdown (sockethandler ∗s)

  Function to notify main loop thread, that it should shut down.
- void sockethandler_destroy (sockethandler ∗s)

  Destructor for sockethandler.

4.11.1 Detailed Description

File containing the sockethandler structure and its member methods.

Author

Lukas Elsner

Date

01-09-2014

4.11.2 Macro Definition Documentation

4.11.2.1 #define MAX_SOCKETS 5

Maximum number of waiting clients

4.11.2.2 #define MAX_THREADS 10

Size of the Threadpool

4.11.3 Function Documentation

4.11.3.1 void sockethandler_client_thread_func (**sockethandler** ∗ *s*)

Method for client connection handling.

Parameters

sockethandler∗	A pointer to a valid sockethandler structure

Every Thread is a consumer for the client_sockets[] array. If a socket is available, it is popped out by one of the threads and served in a loop until the connection closes. After that, the thread waits for its next client socket.

4.11.3.2 void sockethandler_destroy (**sockethandler** ∗)

Destructor for sockethandler.

Parameters

sockethandler∗	Pointer to structure to be freed

4.11.3.3 **sockethandler**∗ sockethandler_init (**foodlist** ∗)

Constructor for sockethandler.

Returns

A pointer to the sockethandler structure, representing the created object

After using this structure, it must be freed with sockethandler_destroy(food ∗)

4.11.3.4 void sockethandler_server_thread_func (**sockethandler** ∗ *s*)

Main loop function for the sockethandling procedure.

Parameters

sockethandler∗	A pointer to a valid initialized sockethandler structure

This method starts a listening socket and produces client sockets for the spawned threads which are responsible for client connection handling. The method returns after sockethandler_shutdown() was called and all threads ended gracefully.

4.11.3.5 void sockethandler_set_port (**sockethandler** ∗ *s*, int *port*)

Method for setting the listening port of a sockethandler structure.

Parameters

sockethandler∗	Pointer to structure to work on
int	Port to listen on

4.11.3.6 void sockethandler_shutdown (sockethandler ∗ *s*)

Function to notify main loop thread, that it should shut down.

Parameters

sockethandler∗	A pointer to a valid initialized sockethandler structure

This method sets the shutdown flag for the sockethandler structure. After that it joins all running threads before it returns.

4.12 sockethandler.h File Reference

Header containing the public accessible sockethandler methods.

```
#include "../lib/foodlist.h"
```

Typedefs

- typedef struct sockethandler sockethandler

  Forward declaration for food.

Functions

- sockethandler ∗ sockethandler_init (foodlist ∗)

  Constructor for sockethandler.
- void sockethandler_server_thread_func (sockethandler ∗s)

  Main loop function for the sockethandling procedure.
- void sockethandler_shutdown (sockethandler ∗s)

  Function to notify main loop thread, that it should shut down.
- void sockethandler_set_port (sockethandler ∗s, int port)

  Method for setting the listening port of a sockethandler structure.
- void sockethandler_destroy (sockethandler ∗)

  Destructor for sockethandler.

4.12.1 Detailed Description

Header containing the public accessible sockethandler methods.

Author

Lukas Elsner

Date

25-09-2014

4.12.2 Function Documentation

4.12.2.1 void sockethandler_destroy (sockethandler ∗)

Destructor for sockethandler.

Parameters

sockethandler∗	Pointer to structure to be freed

4.12.2.2 sockethandler∗ sockethandler_init (foodlist ∗)

Constructor for sockethandler.

Returns

A pointer to the sockethandler structure, representing the created object

After using this structure, it must be freed with sockethandler_destroy(food ∗)

4.12.2.3 void sockethandler_server_thread_func (sockethandler ∗ *s*)

Main loop function for the sockethandling procedure.

Parameters

sockethandler∗	A pointer to a valid initialized sockethandler structure

This method starts a listening socket and produces client sockets for the spawned threads which are responsible for client connection handling. The method returns after sockethandler_shutdown() was called and all threads ended gracefully.

4.12.2.4 void sockethandler_set_port (sockethandler ∗ *s,* int *port*)

Method for setting the listening port of a sockethandler structure.

Parameters

sockethandler∗	Pointer to structure to work on
int	Port to listen on

4.12.2.5 void sockethandler_shutdown (sockethandler ∗ *s*)

Function to notify main loop thread, that it should shut down.

Parameters

sockethandler∗	A pointer to a valid initialized sockethandler structure

This method sets the shutdown flag for the sockethandler structure. After that it joins all running threads before it returns.

Index