Ogród zoologiczny

Generated by Doxygen 1.8.14

Contents

Chapter 1

Ogród zoologiczny

Disclaimer: THIS PROJECT DOESN'T MAKE ANY SENSE AT ALL

A simple GTK app for an university project

Warning: lots of duplicated code in order to meet the 1000-line minimum

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

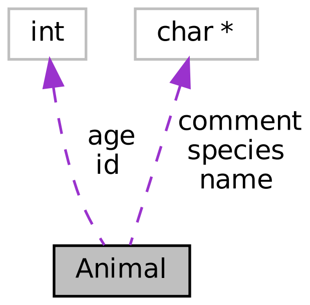
Chapter 4

Data Structure Documentation

4.1 Animal Struct Reference

```
#include <animal.h>
```

Collaboration diagram for Animal:



Data Fields

- int id
- char ∗ name
- char ∗ species
- int age
- char ∗ comment

4.1.1 Detailed Description

Definition at line 20 of file animal.h.

4.1.2 Field Documentation

4.1.2.1 age

```
int Animal::age
```

Definition at line 24 of file animal.h.

Referenced by add_animal_to_table(), animal_new(), and cmp().

4.1.2.2 comment

```
char* Animal::comment
```

Definition at line 25 of file animal.h.

Referenced by add_animal_to_table(), animal_new(), and cmp().

4.1.2.3 id

```
int Animal::id
```

Definition at line 21 of file animal.h.

Referenced by add_animal_to_table(), animal_new(), and cmp().

4.1.2.4 name

```
char* Animal::name
```

Definition at line 22 of file animal.h.

Referenced by add_animal_to_table(), animal_new(), and cmp().

4.1.2.5 species

```
char* Animal::species
```

Definition at line 23 of file animal.h.

Referenced by add_animal_to_table(), animal_new(), and cmp().

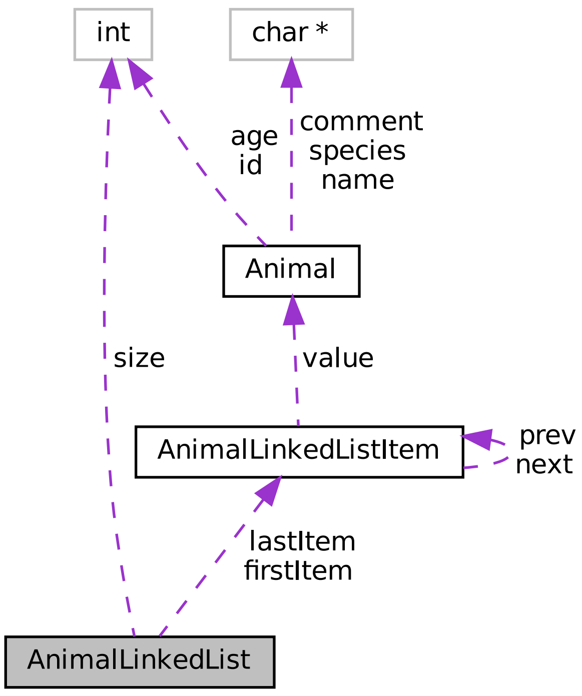
The documentation for this struct was generated from the following file:

- include/data/animal.h

4.2 AnimalLinkedList Struct Reference

```
#include <animal.h>
```

Collaboration diagram for AnimalLinkedList:



Data Fields

- int size
- struct AnimalLinkedListItem ∗ firstItem
- struct AnimalLinkedListItem ∗ lastItem

4.2.1 Detailed Description

Definition at line 31 of file animal.h.

4.2.2 Field Documentation

4.2.2.1 firstItem

`struct AnimalLinkedListItem* AnimalLinkedList::firstItem`

Definition at line 33 of file animal.h.

Referenced by animal_linked_list_add_item(), animal_linked_list_sort(), and fill_table().

4.2.2.2 lastItem

`struct AnimalLinkedListItem* AnimalLinkedList::lastItem`

Definition at line 34 of file animal.h.

Referenced by animal_linked_list_add_item(), and animal_linked_list_sort().

4.2.2.3 size

`int AnimalLinkedList::size`

Definition at line 32 of file animal.h.

Referenced by animal_linked_list_add_item(), animal_linked_list_new(), animal_linked_list_sort(), and fill_table().

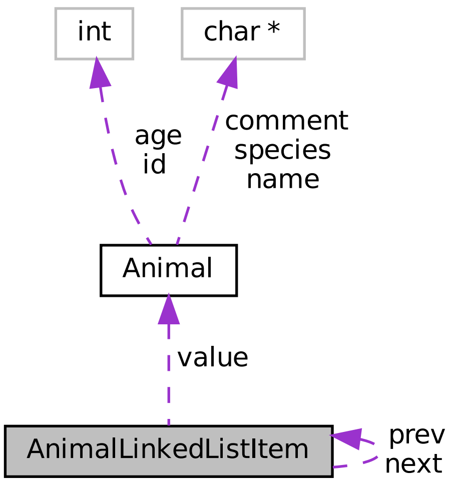
The documentation for this struct was generated from the following file:

- include/data/animal.h

4.3 AnimalLinkedListItem Struct Reference

```
#include <animal.h>
```

Collaboration diagram for AnimalLinkedListItem:



Data Fields

- Animal ∗ value
- struct AnimalLinkedListItem ∗ prev
- struct AnimalLinkedListItem ∗ next

4.3.1 Detailed Description

Definition at line 40 of file animal.h.

4.3.2 Field Documentation

4.3.2.1 next

```
struct AnimalLinkedListItem* AnimalLinkedListItem::next
```

Definition at line 43 of file animal.h.

Referenced by animal_linked_list_add_item(), animal_linked_list_sort(), and fill_table().

4.3.2.2 prev

```
struct AnimalLinkedListItem* AnimalLinkedListItem::prev
```

Definition at line 42 of file animal.h.

Referenced by animal_linked_list_add_item(), and animal_linked_list_sort().

4.3.2.3 value

```
Animal* AnimalLinkedListItem::value
```

Definition at line 41 of file animal.h.

Referenced by animal_linked_list_item_new(), animal_linked_list_sort(), and fill_table().

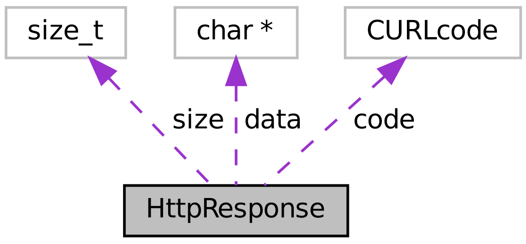
The documentation for this struct was generated from the following file:

- include/data/animal.h

4.4 HttpResponse Struct Reference

```
#include <http.h>
```

Collaboration diagram for HttpResponse:



Data Fields

- char ∗ data
- size_t size
- CURLcode code

4.4.1 Detailed Description

Definition at line 8 of file http.h.

4.4.2 Field Documentation

4.4.2.1 code

```
CURLcode HttpResponse::code
```

Definition at line 11 of file http.h.

Referenced by http_get().

4.4.2.2 data

```
char* HttpResponse::data
```

Definition at line 9 of file http.h.

Referenced by http_get(), http_response_new(), and write_function().

4.4.2.3 size

```
size_t HttpResponse::size
```

Definition at line 10 of file http.h.

Referenced by http_response_new(), and write_function().

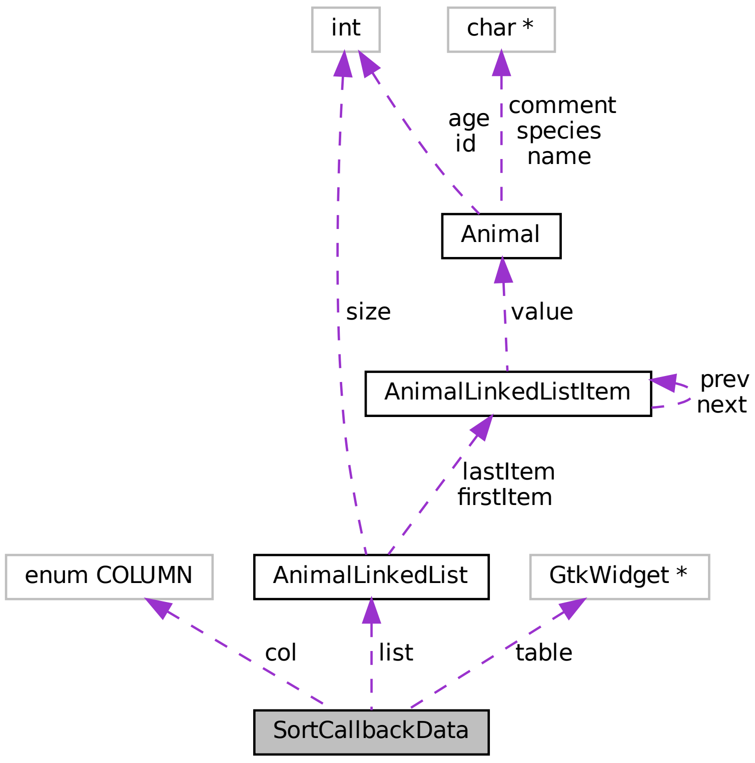
The documentation for this struct was generated from the following file:

- include/services/http.h

4.5 SortCallbackData Struct Reference

```
#include <main_window.h>
```

Collaboration diagram for SortCallbackData:



Data Fields

- enum COLUMN col
- AnimalLinkedList ∗ list
- GtkWidget ∗ table

4.5.1 Detailed Description

Definition at line 31 of file main_window.h.

4.5.2 Field Documentation

4.5.2.1 col

```
enum COLUMN SortCallbackData::col
```

Definition at line 32 of file main_window.h.

Referenced by callback_sort_click(), and sort_callback_data_new().

4.5.2.2 list

```
AnimalLinkedList* SortCallbackData::list
```

Definition at line 33 of file main_window.h.

Referenced by callback_sort_click(), and sort_callback_data_new().

4.5.2.3 table

```
GtkWidget* SortCallbackData::table
```

Definition at line 34 of file main_window.h.

Referenced by callback_sort_click(), and sort_callback_data_new().

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

- include/main_window.h

Chapter 5

File Documentation

5.1 CMakeFiles/3.13.2/CompilerIdC/CMakeCCompilerId.c File Reference

Macros

- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X) #X
- #define STRINGIFY(X) STRINGIFY_HELPER(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define C_DIALECT

Functions

- int main (int argc, char ∗argv[ ])

Variables

- char const ∗ info_compiler = "INFO" ":" "compiler[" "" "]"
- char const ∗ info_platform = "INFO" ":" "platform[" "]"
- char const ∗ info_arch = "INFO" ":" "arch[" "]"
- const char ∗ info_language_dialect_default

5.1.1 Macro Definition Documentation

5.1.1.1 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

Definition at line 492 of file CMakeCCompilerId.c.

5.1.1.2 C_DIALECT

```
#define C_DIALECT
```

Definition at line 577 of file CMakeCCompilerId.c.

5.1.1.3 COMPILER_ID

```
#define COMPILER_ID ""
```

Definition at line 312 of file CMakeCCompilerId.c.

5.1.1.4 DEC

```
#define DEC(
              n )
```

Value:

```
('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

Definition at line 496 of file CMakeCCompilerId.c.

5.1.1.5 HEX

```
#define HEX(
              n )
```

Value:

```
('0' + ((n)>>28 & 0xF)), \
  ('0' + ((n)>>24 & 0xF)), \
  ('0' + ((n)>>20 & 0xF)), \
  ('0' + ((n)>>16 & 0xF)), \
  ('0' + ((n)>>12 & 0xF)), \
  ('0' + ((n)>>8  & 0xF)), \
  ('0' + ((n)>>4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

Definition at line 507 of file CMakeCCompilerId.c.

5.1.1.6 PLATFORM_ID

```
#define PLATFORM_ID
```

Definition at line 429 of file CMakeCCompilerId.c.

5.1.1.7 STRINGIFY

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

Definition at line 333 of file CMakeCCompilerId.c.

5.1.1.8 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
              X ) #X
```

Definition at line 332 of file CMakeCCompilerId.c.

5.1.2 Function Documentation

5.1.2.1 main()

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

Definition at line 597 of file CMakeCCompilerId.c.

References info_arch, info_compiler, info_language_dialect_default, and info_platform.

```
{
  int require = 0;
  require += info_compiler[argc];
  require += info_platform[argc];
  require += info_arch[argc];
#ifdef COMPILER_VERSION_MAJOR
  require += info_version[argc];
#endif
#ifdef COMPILER_VERSION_INTERNAL
  require += info_version_internal[argc];
#endif
#ifdef SIMULATE_ID
  require += info_simulate[argc];
#endif
#ifdef SIMULATE_VERSION_MAJOR
  require += info_simulate_version[argc];
#endif
#if defined(__CRAYXE) || defined(__CRAYXC)
  require += info_cray[argc];
#endif
  require += info_language_dialect_default[argc];
  (void)argv;
  return require;
}
```

5.1.3 Variable Documentation

5.1.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" "]"
```

Definition at line 567 of file CMakeCCompilerId.c.

Referenced by main().

5.1.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" "" "]"
```

Definition at line 319 of file CMakeCCompilerId.c.

Referenced by main().

5.1.3.3 info_language_dialect_default

```
const char* info_language_dialect_default
```

Initial value:

```
=
  "INFO" ":" "dialect_default["    "]"
```

Definition at line 586 of file CMakeCCompilerId.c.

Referenced by main().

5.1.3.4 info_platform

```
char const* info_platform = "INFO" ":" "platform[" "]"
```

Definition at line 566 of file CMakeCCompilerId.c.

Referenced by main().

5.2 CMakeFiles/feature_tests.c File Reference

Functions

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

Variables

- const char features [ ]

5.2.1 Function Documentation

5.2.1.1 main()

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

Definition at line 34 of file feature_tests.c.

References features.

```
34 { (void)argv; return features[argc]; }
```

5.2.2 Variable Documentation

5.2.2.1 features

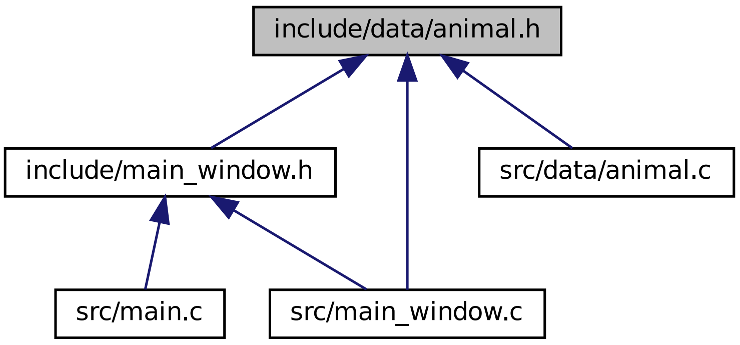
```
const char features[ ]
```

Definition at line 2 of file feature_tests.c.

Referenced by main().

5.3 include/data/animal.h File Reference

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



Data Structures

- struct Animal
- struct AnimalLinkedList
- struct AnimalLinkedListItem

Typedefs

- typedef struct Animal Animal
- typedef struct AnimalLinkedList AnimalLinkedList
- typedef struct AnimalLinkedListItem AnimalLinkedListItem

Functions

- Animal ∗ animal_new (int id, char ∗name, char ∗species, int age, char ∗comment)
- void animal_linked_list_sort (AnimalLinkedList ∗list, int(∗cmp)(Animal ∗a, Animal ∗b))
- AnimalLinkedList ∗ animal_linked_list_new ()
- void animal_linked_list_add_item (AnimalLinkedList ∗list, Animal ∗value)
- Animal animal_linked_list_get_item (int index)

5.3.1 Typedef Documentation

5.3.1.1 Animal

```
typedef struct Animal Animal
```

5.3.1.2 AnimalLinkedList

```
typedef struct AnimalLinkedList AnimalLinkedList
```

5.3.1.3 AnimalLinkedListItem

```
typedef struct AnimalLinkedListItem AnimalLinkedListItem
```

5.3.2 Function Documentation

5.3.2.1 animal_linked_list_add_item()

```
void animal_linked_list_add_item (
             AnimalLinkedList * list,
             Animal * value )
```

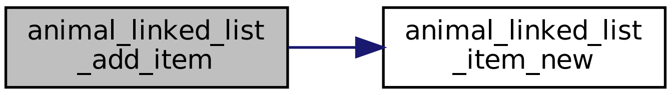
Definition at line 93 of file animal.c.

References animal_linked_list_item_new(), AnimalLinkedList::firstItem, AnimalLinkedList::lastItem, AnimalLinked↩ ListItem::next, AnimalLinkedListItem::prev, and AnimalLinkedList::size.

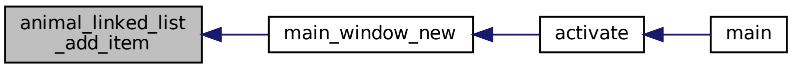
Referenced by main_window_new().

```
                                                                 {

    AnimalLinkedListItem* cur = list->firstItem;
    AnimalLinkedListItem* toAdd =
   animal_linked_list_item_new(value);

    if(list->size == 0){
        list->firstItem = toAdd;
         list->lastItem = toAdd;
         list->size ++;
         return;
     }
     int i;
     for(i=0; i<list->size-1; ++i){
         cur = cur->next;
     }

     cur->next = toAdd;
     list->lastItem = toAdd;
     toAdd->prev = cur;

     list->size ++;
 };
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.2 animal_linked_list_get_item()

```
Animal animal_linked_list_get_item (
            int index )
```

5.3.2.3 animal_linked_list_new()

```
AnimalLinkedList* animal_linked_list_new ( )
```

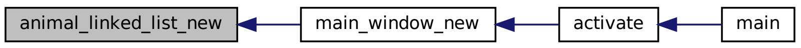
Definition at line 48 of file animal.c.

References AnimalLinkedList::size.

Referenced by main_window_new().

```
48                    {
49     AnimalLinkedList* list = malloc(sizeof(AnimalLinkedList));
50     list->size = 0;
51     return list;
52 }
```

Here is the caller graph for this function:



5.3.2.4 animal_linked_list_sort()

```
void animal_linked_list_sort (
            AnimalLinkedList * list,
            int(*)(Animal *a, Animal *b) cmp )
```

Definition at line 55 of file animal.c.

References cmp(), AnimalLinkedList::firstItem, AnimalLinkedList::lastItem, AnimalLinkedListItem::next, Animal↩
LinkedListItem::prev, AnimalLinkedList::size, and AnimalLinkedListItem::value.

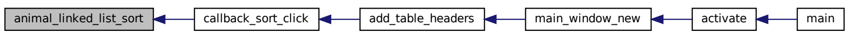
Referenced by callback_sort_click().

```
{
    int changed = 1;
    while(changed){
        changed = 0;
        AnimalLinkedListItem* cur = list->firstItem;
        for(int i=0; i<list->size-1; ++i){
            if((*cmp)(cur->value, cur->next->value)){
                changed = 1;
                int has_prev = list->firstItem != cur;
                int has_next = list->lastItem != cur->next;
                AnimalLinkedListItem* node_0 = has_prev ? cur->
    prev : NULL;
                AnimalLinkedListItem* node_1 = cur;
                AnimalLinkedListItem* node_2 = cur->next;
                AnimalLinkedListItem* node_3 = has_next ? cur->
    next->next : NULL;
                node_1->next = node_2->next;
                node_1->prev = node_2;
                node_2->next = node_1;
                node_2->prev = node_0;
                if(has_prev){
                    node_0->next = node_2;
                } else {
                    list->firstItem = node_2;
                }
                if(has_next){
                    node_3->prev = node_1;
                } else {
                    list->lastItem = node_1;
                }
                cur = node_2;
            }
            cur = cur->next;
        }
    }
}
```

Here is the call graph for this function:

animal_linked_list_sort
cmp

Here is the caller graph for this function:



5.3.2.5 animal_new()

```
Animal* animal_new (
            int id,
            char * name,
            char * species,
            int age,
            char * comment )
```

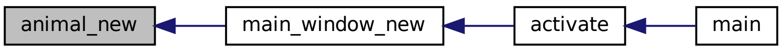
Definition at line 33 of file animal.c.

References Animal::age, Animal::comment, Animal::id, Animal::name, and Animal::species.

Referenced by main_window_new().

```
                                                                     {
    Animal* animal = malloc(sizeof(Animal));
    animal->id = id;
    animal->name = malloc(strlen(name)+1);
    strcpy(animal->name, name);
    animal->species = malloc(strlen(species)+1);
    strcpy(animal->species, species);

    animal->age = age;
    animal->comment = malloc(strlen(comment)+1);
    strcpy(animal->comment, comment);
    return animal;
}
```

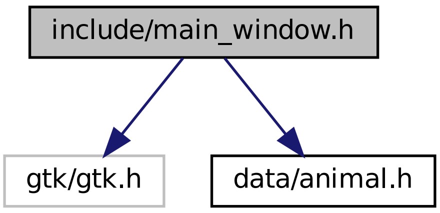
Here is the caller graph for this function:



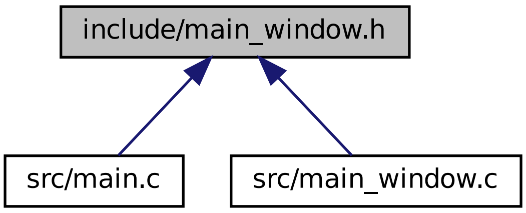
5.4 include/main_window.h File Reference

```
#include <gtk/gtk.h>
#include "data/animal.h"
```

Include dependency graph for main_window.h:



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



Data Structures

- struct SortCallbackData

Typedefs

- typedef enum COLUMN COLUMN
- typedef struct SortCallbackData SortCallbackData

Enumerations

- enum COLUMN {
  ID, NAME, SPECIES, AGE,
  COMMENT }

Functions

- GtkWidget ∗ main_window_new (GtkApplication ∗app)

  Initialize the main window.
- void callback_sort_click (GtkWidget ∗widget, gpointer callback_data)
- void add_table_headers (GtkWidget ∗mainContainer, GtkWidget ∗table, AnimalLinkedList ∗animals)

  Add table header buttons.
- void add_control_buttons (GtkApplication ∗app, GtkWidget ∗mainContainer)
- void fill_table (GtkWidget ∗table, AnimalLinkedList ∗animals)

5.4.1 Typedef Documentation

5.4.1.1 COLUMN

```
typedef enum COLUMN COLUMN
```

5.4.1.2 SortCallbackData

```
typedef struct SortCallbackData SortCallbackData
```

5.4.2 Enumeration Type Documentation

5.4.2.1 COLUMN

```
enum COLUMN
```

Enumerator

ID	
NAME	
SPECIES	
AGE	
COMMENT	

Definition at line 24 of file main_window.h.

```
                 {ID,
      NAME,
      SPECIES,
      AGE,
      COMMENT}
```

5.4.3 Function Documentation

5.4.3.1 add_control_buttons()

```
void add_control_buttons (
             GtkApplication * app,
             GtkWidget * mainContainer )
```

Add edit and delete buttons below the table

Parameters

mainContainer	

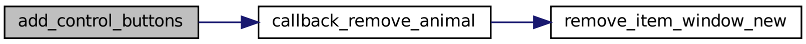
Definition at line 97 of file main_window.c.

References callback_remove_animal().

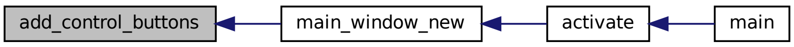
Referenced by main_window_new().

```
                        {
    GtkWidget* buttonRemoveAnimal = gtk_button_new_with_label("Usuń element");
    gtk_widget_set_hexpand(buttonRemoveAnimal, 1);
    gtk_grid_attach(GTK_GRID(mainContainer), buttonRemoveAnimal, 0, 2, 2, 1);
    g_signal_connect(G_OBJECT(buttonRemoveAnimal), "clicked", G_CALLBACK(
  callback_remove_animal), (gpointer) app);


    GtkWidget* buttonAddAnimal = gtk_button_new_with_label("Dodaj element");
    gtk_widget_set_hexpand(buttonAddAnimal, 1);
    gtk_widget_set_margin_start(buttonAddAnimal, 5);
    gtk_grid_attach(GTK_GRID(mainContainer), buttonAddAnimal, 2, 2, 3, 1);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.3.2 add_table_headers()

```
void add_table_headers (
             GtkWidget * mainContainer,
             GtkWidget * table,
             AnimalLinkedList * animals )
```

Add table header buttons.

Parameters

mainContainer	the main GtkGrid of the window
table	table
animals	list of animals

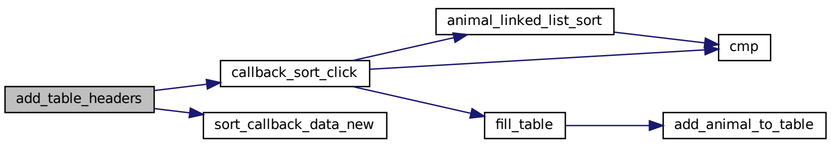
Definition at line 147 of file main_window.c.

References AGE, callback_sort_click(), COMMENT, ID, NAME, sort_callback_data_new(), and SPECIES.

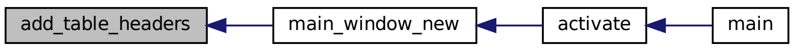
Referenced by main_window_new().

```
{
    GtkWidget *buttonId = gtk_button_new();
    gtk_button_set_label(GTK_BUTTON(buttonId), "Id");
    gtk_widget_set_hexpand(buttonId, TRUE);

    g_signal_connect(G_OBJECT(buttonId), "clicked", G_CALLBACK(
  callback_sort_click), (gpointer) sort_callback_data_new(
  ID, animals, table));
    gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonId), 0, 0, 1, 1);

    GtkWidget *buttonName = gtk_button_new();
    gtk_button_set_label(GTK_BUTTON(buttonName), "Imię");
    gtk_widget_set_hexpand(buttonName, TRUE);
    g_signal_connect(G_OBJECT(buttonName), "clicked", G_CALLBACK(
  callback_sort_click), (gpointer) sort_callback_data_new(
  NAME, animals, table));
    gtk_widget_set_margin_start(buttonName, 5);
    gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonName), 1, 0, 1, 1);

    GtkWidget *buttonSpecies = gtk_button_new();
    gtk_button_set_label(GTK_BUTTON(buttonSpecies), "Gatunek");
    gtk_widget_set_hexpand(buttonSpecies, TRUE);
    g_signal_connect(G_OBJECT(buttonSpecies), "clicked", G_CALLBACK(
  callback_sort_click),
                     (gpointer) sort_callback_data_new(
  SPECIES, animals, table));
    gtk_widget_set_margin_start(buttonSpecies, 5);
    gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonSpecies), 2, 0, 1, 1);

    GtkWidget* buttonAge = gtk_button_new();
    gtk_button_set_label(GTK_BUTTON(buttonAge), "Wiek");
    gtk_widget_set_hexpand(buttonAge, TRUE);
    g_signal_connect(G_OBJECT(buttonAge), "clicked", G_CALLBACK(
  callback_sort_click),
                     (gpointer) sort_callback_data_new(AGE, animals, table));
    gtk_widget_set_margin_start(buttonAge, 5);
    gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonAge), 3, 0, 1, 1);

    GtkWidget* buttonComment = gtk_button_new();
    gtk_button_set_label(GTK_BUTTON(buttonComment), "Komentarz");
    gtk_widget_set_hexpand(buttonComment, TRUE);
    g_signal_connect(G_OBJECT(buttonComment), "clicked", G_CALLBACK(
  callback_sort_click),
                     (gpointer) sort_callback_data_new(
  COMMENT, animals, table));
    gtk_widget_set_margin_start(buttonComment, 5);
    gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonComment), 4, 0, 1, 1);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.3.3 callback_sort_click()

```
void callback_sort_click (
             GtkWidget * widget,
             gpointer callback_data )
```

On click on any of the header buttons

Parameters

widget	
callback_data	an instance of SortCallbackData

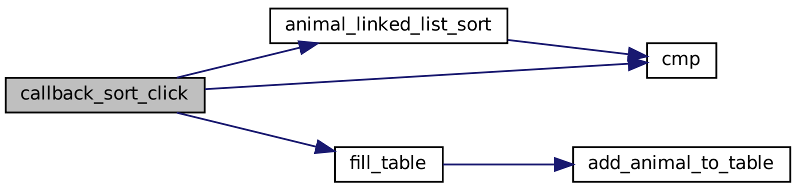
Definition at line 197 of file main_window.c.

References animal_linked_list_sort(), cmp(), SortCallbackData::col, fill_table(), SortCallbackData::list, sort_asc, sort_by, and SortCallbackData::table.

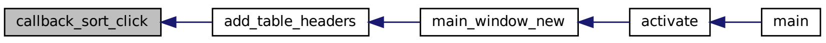
Referenced by add_table_headers().

```
{
    SortCallbackData* cd = (SortCallbackData*) callback_data;
    if(sort_by == cd->col) sort_asc = !sort_asc;
    sort_by = cd->col;
    animal_linked_list_sort(cd->list, cmp);
    fill_table(cd->table, cd->list);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.3.4 fill_table()

```
void fill_table (
             GtkWidget * table,
             AnimalLinkedList * animals )
```

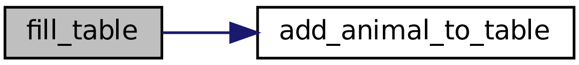
Definition at line 264 of file main_window.c.

References add_animal_to_table(), AnimalLinkedList::firstItem, AnimalLinkedListItem::next, AnimalLinkedList::size, and AnimalLinkedListItem::value.

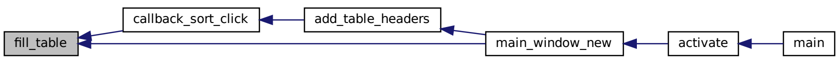
Referenced by callback_sort_click(), and main_window_new().

```
{
    GList *children, *iter;

    children = gtk_container_get_children(GTK_CONTAINER(table));
    for(iter = children; iter != NULL; iter = g_list_next(iter))
        gtk_widget_destroy(GTK_WIDGET(iter->data));
    g_list_free(children);

    AnimalLinkedListItem* cur = animals->firstItem;
    for(int i=0; i<animals->size; ++i){
        add_animal_to_table(table, cur->value, i+1);
        cur = cur->next;
    }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.3.5 main_window_new()

```
GtkWidget* main_window_new (
             GtkApplication * app )
```

Initialize the main window.

Parameters

app	Application

Returns

main window, not shown yet

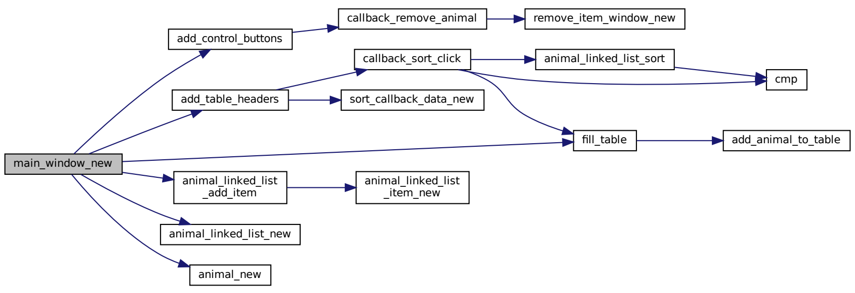
Definition at line 49 of file main_window.c.

References add_control_buttons(), add_table_headers(), animal_linked_list_add_item(), animal_linked_list_new(), animal_new(), and fill_table().

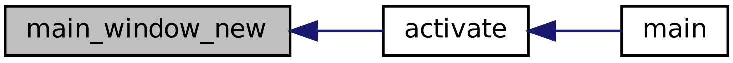
Referenced by activate().

```
{

    GtkWidget* window;
    GtkWidget* mainContainer;

    window = gtk_application_window_new (app);
    gtk_window_set_title (GTK_WINDOW (window), "Ogród zoologiczny");
    gtk_window_set_default_size (GTK_WINDOW (window), 500, 500);

    mainContainer = gtk_grid_new();
    gtk_container_add(GTK_CONTAINER (window), GTK_WIDGET (mainContainer));

    GtkWidget* containerTable;
    containerTable = gtk_scrolled_window_new(NULL, NULL);
    gtk_widget_set_hexpand(containerTable, 1);
    gtk_widget_set_vexpand(containerTable, 1);
    GtkWidget* table;
    table = gtk_grid_new();
    AnimalLinkedList* animals = animal_linked_list_new();

    for(int i=0; i<200; ++i){
        Animal* a = animal_new(i, "Blazej", "Wielblad", i/4+3, "Je orzeszki");
        animal_linked_list_add_item(animals, a);
    }
    gtk_container_add(GTK_CONTAINER(containerTable), GTK_WIDGET(table));
    gtk_grid_set_column_homogeneous(GTK_GRID(table), gtk_true());
    gtk_grid_set_column_homogeneous(GTK_GRID(mainContainer), gtk_true());
    add_table_headers(mainContainer, table, animals);
    fill_table(table, animals);
    gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (containerTable), 0, 1, 5, 1);
    add_control_buttons(app, mainContainer);

    return window;
}
```

Here is the call graph for this function:



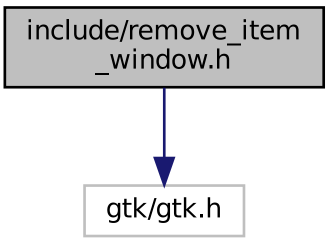
Here is the caller graph for this function:



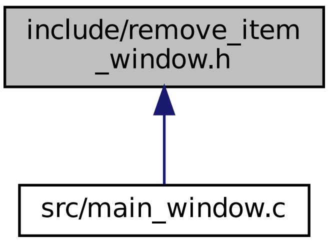
5.5 include/remove_item_window.h File Reference

```
#include <gtk/gtk.h>
```

Include dependency graph for remove_item_window.h:



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



Functions

- GtkWidget ∗ remove_item_window_new (GtkApplication ∗app)

5.5.1 Function Documentation

5.5.1.1 remove_item_window_new()

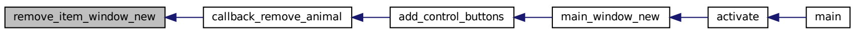
```
GtkWidget* remove_item_window_new (
             GtkApplication * app )
```

Definition at line 22 of file remove_item_window.c.

Referenced by callback_remove_animal().

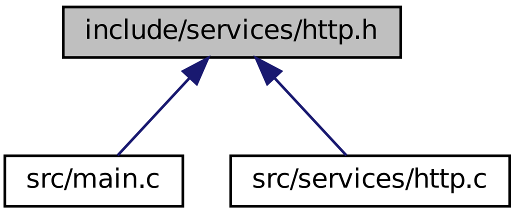
```
                                       {
    GtkWidget* window;
    GtkWidget* mainContainer;
    mainContainer = gtk_grid_new();
    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    gtk_container_add(GTK_CONTAINER(window), mainContainer);
    GtkEntry* entryId = gtk_entry_new();
    gtk_entry_set_placeholder_text(entryId, "Id elementu");
    gtk_grid_attach(GTK_GRID(mainContainer), entryId, 0, 0, 1, 1);
    gtk_widget_set_hexpand(entryId, 1);
    gtk_widget_set_vexpand(entryId, 1);
    gtk_window_set_title (GTK_WINDOW (window), "Usuwanie elementu");

    return window;
}
```

Here is the caller graph for this function:



5.6 include/services/http.h File Reference

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



Data Structures

- struct HttpResponse

Typedefs

- typedef struct HttpResponse HttpResponse

Functions

- size_t write_function (void ∗ptr, size_t size, size_t nmemb, HttpResponse ∗r)
- HttpResponse ∗ http_get (char ∗url)

    Gets HTTP response for url.

5.6.1 Typedef Documentation

5.6.1.1 HttpResponse

```
typedef struct HttpResponse HttpResponse
```

5.6.2 Function Documentation

5.6.2.1 http_get()

```
HttpResponse* http_get (
              char * url )
```

Gets HTTP response for url.

Parameters

url	Site address

Returns

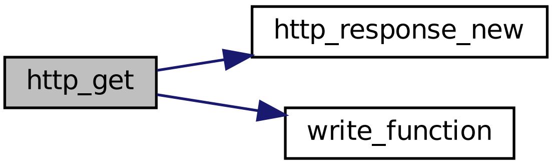
Filled HttpResponse

Definition at line 46 of file http.c.

References HttpResponse::code, HttpResponse::data, http_response_new(), and write_function().

```
                                        {
    CURL *curl;
    HttpResponse * res = http_response_new();

    curl = curl_easy_init();
    if(curl) {
        curl_easy_setopt(curl, CURLOPT_URL, url);
        /* example.com is redirected, so we tell libcurl to follow redirection */
        curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);
        curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, write_function);
        curl_easy_setopt(curl, CURLOPT_WRITEDATA, res);

        /* Perform the request, res->code will get the return code */
        res->code = curl_easy_perform(curl);
        /* Check for errors */
        if(res->code != CURLE_OK)
            fprintf(stderr, "curl_easy_perform() failed: %s\n",
                    curl_easy_strerror(res->code));
        else {
            /* Print data for debug */
            printf("Data: %s\n", res->data);
        }

        /* Cleanup */
        curl_easy_cleanup(curl);
    }
    return res;
}
```

Here is the call graph for this function:



5.6.2.2 write_function()

```
size_t write_function (
              void * ptr,
              size_t size,
              size_t nmemb,
              HttpResponse * r )
```

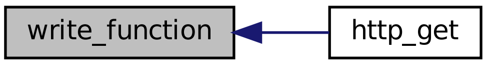
Definition at line 25 of file http.c.

References HttpResponse::data, and HttpResponse::size.

Referenced by http_get().

```
                                                                               {
    size_t new_len = r->size + size*nmemb;
    r->data= realloc(r->data, new_len+1);
    memcpy(r->data+r->size, ptr, size*nmemb);
    r->data[new_len] = '\0';
    return size*nmemb;
}
```

Here is the caller graph for this function:

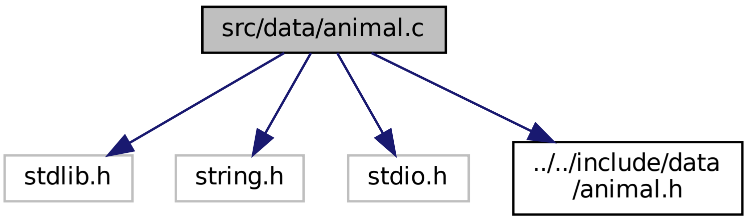


5.7 README.md File Reference

5.8 src/data/animal.c File Reference

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "../../include/data/animal.h"
```

Include dependency graph for animal.c:



Functions

- AnimalLinkedListItem ∗ animal_linked_list_item_new (Animal ∗value)
- Animal ∗ animal_new (int id, char ∗name, char ∗species, int age, char ∗comment)
- AnimalLinkedList ∗ animal_linked_list_new ()
- void animal_linked_list_sort (AnimalLinkedList ∗list, int(∗cmp)(Animal ∗a, Animal ∗b))
- void animal_linked_list_add_item (AnimalLinkedList ∗list, Animal ∗value)

5.8.1 Function Documentation

5.8.1.1 animal_linked_list_add_item()

```
void animal_linked_list_add_item (
             AnimalLinkedList * list,
             Animal * value )
```

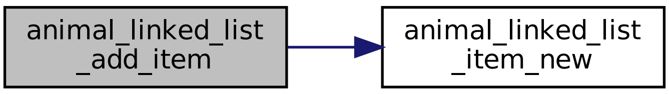
Definition at line 93 of file animal.c.

References animal_linked_list_item_new(), AnimalLinkedList::firstItem, AnimalLinkedList::lastItem, AnimalLinkedListItem::next, AnimalLinkedListItem::prev, and AnimalLinkedList::size.

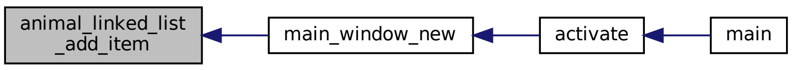
Referenced by main_window_new().

```
                                                                  {

   AnimalLinkedListItem* cur = list->firstItem;
   AnimalLinkedListItem* toAdd =
  animal_linked_list_item_new(value);

   if(list->size == 0){
       list->firstItem = toAdd;
        list->lastItem = toAdd;
        list->size ++;
        return;
    }
    int i;
    for(i=0; i<list->size-1; ++i){
        cur = cur->next;
    }

    cur->next = toAdd;
    list->lastItem = toAdd;
    toAdd->prev = cur;

    list->size ++;
};
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.1.2 animal_linked_list_item_new()

```
AnimalLinkedListItem* animal_linked_list_item_new (
              Animal * value )
```

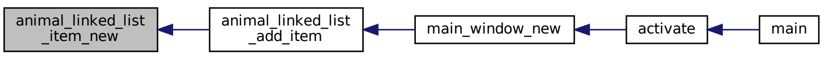
Definition at line 25 of file animal.c.

References AnimalLinkedListItem::value.

Referenced by animal_linked_list_add_item().

```
                                         {
     AnimalLinkedListItem* item = malloc(sizeof(
    AnimalLinkedListItem));
     item->value = value;
     return item;
 }
```

Here is the caller graph for this function:



5.8.1.3 animal_linked_list_new()

```
AnimalLinkedList* animal_linked_list_new ( )
```

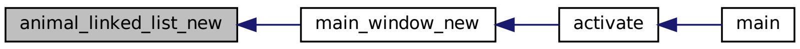
Definition at line 48 of file animal.c.

References AnimalLinkedList::size.

Referenced by main_window_new().

```
                           {
     AnimalLinkedList* list = malloc(sizeof(AnimalLinkedList));
     list->size = 0;
     return list;
 }
```

Here is the caller graph for this function:



5.8.1.4 animal_linked_list_sort()

```
void animal_linked_list_sort (
            AnimalLinkedList * list,
            int(*)(Animal *a, Animal *b) cmp )
```

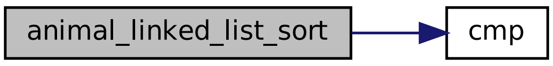
Definition at line 55 of file animal.c.

References cmp(), AnimalLinkedList::firstItem, AnimalLinkedList::lastItem, AnimalLinkedListItem::next, Animal↩
LinkedListItem::prev, AnimalLinkedList::size, and AnimalLinkedListItem::value.

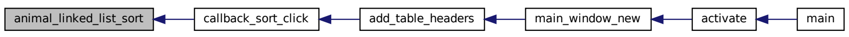
Referenced by callback_sort_click().

```
{
    int changed = 1;
    while(changed){
        changed = 0;
        AnimalLinkedListItem* cur = list->firstItem;
        for(int i=0; i<list->size-1; ++i){
            if((*cmp)(cur->value, cur->next->value)){
                changed = 1;
                int has_prev = list->firstItem != cur;
                int has_next = list->lastItem != cur->next;
                AnimalLinkedListItem* node_0 = has_prev ? cur->
      prev : NULL;
                AnimalLinkedListItem* node_1 = cur;
                AnimalLinkedListItem* node_2 = cur->next;
                AnimalLinkedListItem* node_3 = has_next ? cur->
      next->next : NULL;
                node_1->next = node_2->next;
                node_1->prev = node_2;
                node_2->next = node_1;
                node_2->prev = node_0;
                if(has_prev){
                    node_0->next = node_2;
                } else {
                    list->firstItem = node_2;
                }
                if(has_next){
                    node_3->prev = node_1;
                } else {
                    list->lastItem = node_1;
                }
                cur = node_2;
            }
            cur = cur->next;
        }
    }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.1.5 animal_new()

```
Animal* animal_new (
            int id,
            char * name,
            char * species,
            int age,
            char * comment )
```

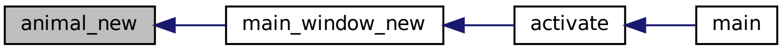
Definition at line 33 of file animal.c.

References Animal::age, Animal::comment, Animal::id, Animal::name, and Animal::species.

Referenced by main_window_new().

```
33                                                                          {
34     Animal* animal = malloc(sizeof(Animal));
35     animal->id = id;
36     animal->name = malloc(strlen(name)+1);
37     strcpy(animal->name, name);
38     animal->species = malloc(strlen(species)+1);
39     strcpy(animal->species, species);
40
41     animal->age = age;
42     animal->comment = malloc(strlen(comment)+1);
43     strcpy(animal->comment, comment);
44     return animal;
45 }
```

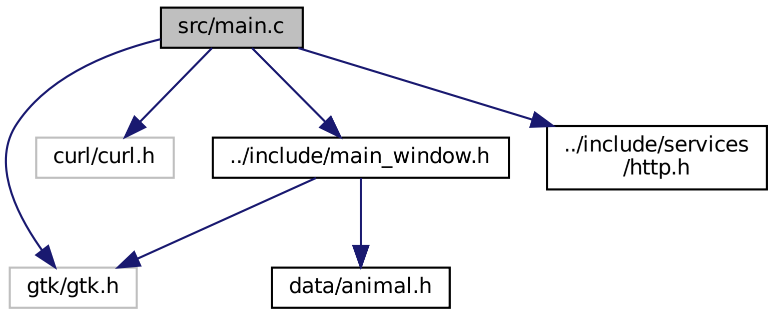
Here is the caller graph for this function:



5.9 src/main.c File Reference

```
#include <gtk/gtk.h>
#include <curl/curl.h>
#include "../include/main_window.h"
#include "../include/services/http.h"
```

Include dependency graph for main.c:



Functions

- static void activate (GtkApplication ∗app, gpointer user_data)

  Initialize UI.
- int main (int argc, char ∗∗argv)

5.9.1 Function Documentation

5.9.1.1 activate()

```
static void activate (
            GtkApplication * app,
            gpointer user_data )  [static]
```

Initialize UI.

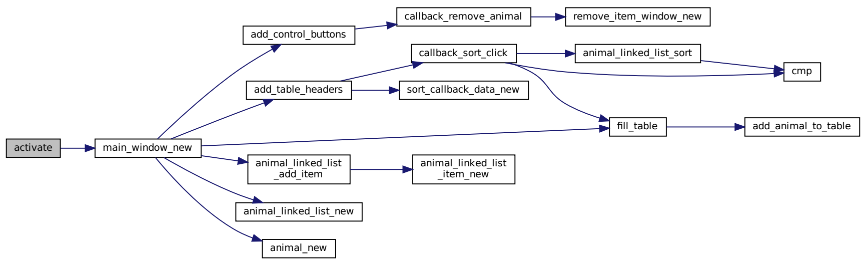
Definition at line 27 of file main.c.

References main_window_new().

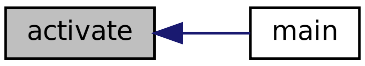
Referenced by main().

```
{
    gtk_widget_show_all (main_window_new(app));
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.1.2 main()

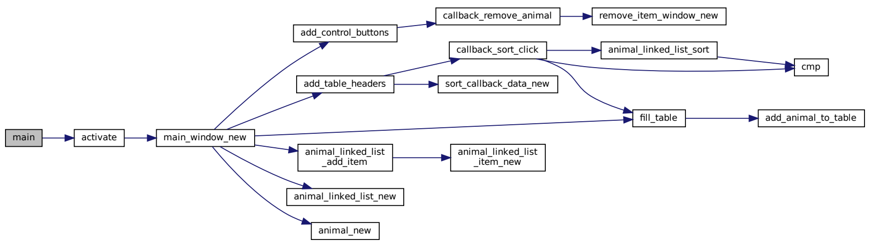
```
int main (
            int argc,
            char ** argv )
```

Definition at line 33 of file main.c.

References activate().

```
                              {
    GtkApplication *app;

    int status;

    app = gtk_application_new ("pl.mrokita.pri-proj-3", G_APPLICATION_FLAGS_NONE);
    g_signal_connect (app, "activate", G_CALLBACK (activate), NULL);
    status = g_application_run (G_APPLICATION (app), argc, argv);
    g_object_unref (app);

    return status;
}
```

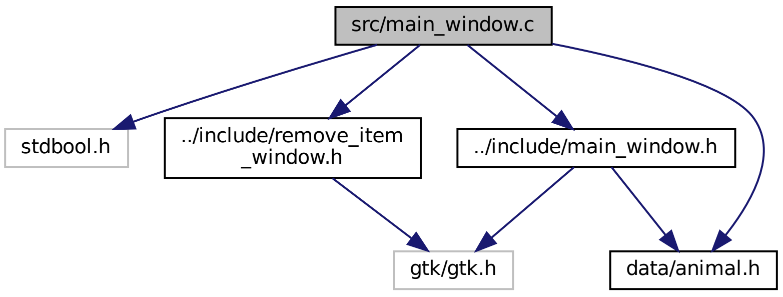
Here is the call graph for this function:



5.10 src/main_window.c File Reference

```
#include <stdbool.h>
#include "../include/main_window.h"
#include "../include/data/animal.h"
#include "../include/remove_item_window.h"
```

Include dependency graph for main_window.c:



Functions

- SortCallbackData ∗ sort_callback_data_new (COLUMN col, AnimalLinkedList ∗list, GtkWidget ∗table)
- GtkWidget ∗ main_window_new (GtkApplication ∗app)

  Initialize the main window.
- void callback_remove_animal (GtkWidget ∗widget, gpointer callback_data)
- void add_control_buttons (GtkApplication ∗app, GtkWidget ∗mainContainer)
- int cmp (Animal ∗a, Animal ∗b)

  Animal comparator Used to sort the table.
- void add_table_headers (GtkWidget ∗mainContainer, GtkWidget ∗table, AnimalLinkedList ∗animals)

  Add table header buttons.
- void callback_sort_click (GtkWidget ∗widget, gpointer callback_data)
- void add_animal_to_table (GtkWidget ∗table, Animal ∗animal, int row)
- void fill_table (GtkWidget ∗table, AnimalLinkedList ∗animals)

Variables

- int sort_by = -1
- int sort_asc = 1

5.10.1 Function Documentation

5.10.1.1 add_animal_to_table()

```
void add_animal_to_table (
            GtkWidget * table,
            Animal * animal,
            int row )
```

Add animal to the table

Parameters

table	
animal	
row	row number, should be empty.

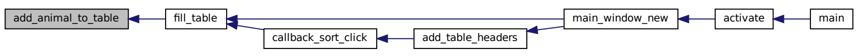
Definition at line 215 of file main_window.c.

References Animal::age, Animal::comment, Animal::id, Animal::name, and Animal::species.

Referenced by fill_table().

```
{
    char* idString = malloc(12);
    sprintf(idString, "%d", animal->id);
    GtkWidget* labelId = gtk_label_new(idString);
    gtk_label_set_xalign(GTK_LABEL (labelId), 0.0);
    gtk_widget_set_margin_start(labelId, 5);
    gtk_widget_set_margin_top(labelId, 5);
    gtk_widget_set_hexpand(labelId, TRUE);
    gtk_grid_attach(GTK_GRID (table), labelId, 0, row, 1, 1);

    GtkWidget* labelName = gtk_label_new(animal->name);
    gtk_label_set_xalign(GTK_LABEL (labelName), 0.0);
    gtk_widget_set_margin_start(labelName, 5);
    gtk_widget_set_margin_top(labelName, 5);
    gtk_widget_set_hexpand(labelName, TRUE);
    gtk_grid_attach(GTK_GRID (table), labelName, 1, row, 1, 1);

    GtkWidget* labelSpecies = gtk_label_new(animal->species);
    gtk_label_set_xalign(GTK_LABEL (labelSpecies), 0.0);
    gtk_widget_set_margin_start(labelSpecies, 5);
    gtk_widget_set_margin_top(labelSpecies, 5);
    gtk_widget_set_hexpand(labelSpecies, TRUE);
    gtk_grid_attach(GTK_GRID (table), labelSpecies, 2, row, 1, 1);

    char* ageString = malloc(12);
    sprintf(ageString, "%d", animal->age);

    GtkWidget* labelAge = gtk_label_new(ageString);
    gtk_label_set_xalign(GTK_LABEL (labelAge), 0.0);
    gtk_widget_set_margin_start(labelAge, 5);
    gtk_widget_set_margin_top(labelAge, 5);
    gtk_widget_set_hexpand(labelAge, TRUE);
    gtk_grid_attach(GTK_GRID (table), labelAge, 3, row, 1, 1);

    GtkWidget* labelComment = gtk_label_new(animal->comment);
    gtk_label_set_xalign(GTK_LABEL (labelComment), 0.0);
    gtk_widget_set_margin_start(labelComment, 5);
    gtk_widget_set_margin_top(labelComment, 5);
    gtk_widget_set_hexpand(labelComment, TRUE);
    gtk_grid_attach(GTK_GRID (table), labelComment, 4, row, 1, 1);
    gtk_widget_show_all(table);
}
```

Here is the caller graph for this function:



5.10.1.2 add_control_buttons()

```
void add_control_buttons (
            GtkApplication * app,
            GtkWidget * mainContainer )
```

Add edit and delete buttons below the table

Parameters

mainContainer	

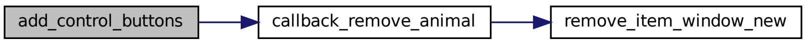
Definition at line 97 of file main_window.c.

References callback_remove_animal().

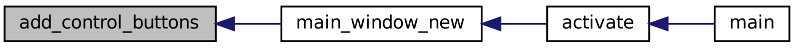
Referenced by main_window_new().

```
                         {
       GtkWidget* buttonRemoveAnimal = gtk_button_new_with_label("Usuń element");
       gtk_widget_set_hexpand(buttonRemoveAnimal, 1);
       gtk_grid_attach(GTK_GRID(mainContainer), buttonRemoveAnimal, 0, 2, 2, 1);
       g_signal_connect(G_OBJECT(buttonRemoveAnimal), "clicked", G_CALLBACK(
     callback_remove_animal), (gpointer) app);


       GtkWidget* buttonAddAnimal = gtk_button_new_with_label("Dodaj element");
       gtk_widget_set_hexpand(buttonAddAnimal, 1);
       gtk_widget_set_margin_start(buttonAddAnimal, 5);
       gtk_grid_attach(GTK_GRID(mainContainer), buttonAddAnimal, 2, 2, 3, 1);
   }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.1.3 add_table_headers()

```
void add_table_headers (
             GtkWidget * mainContainer,
             GtkWidget * table,
             AnimalLinkedList * animals )
```

Add table header buttons.

Parameters

mainContainer	the main GtkGrid of the window
table	table
animals	list of animals

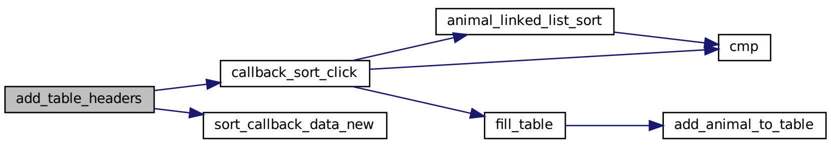
Definition at line 147 of file main_window.c.

References AGE, callback_sort_click(), COMMENT, ID, NAME, sort_callback_data_new(), and SPECIES.

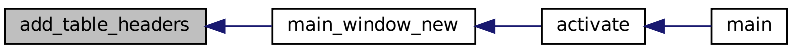
Referenced by main_window_new().

```
{
    GtkWidget *buttonId = gtk_button_new();
    gtk_button_set_label(GTK_BUTTON(buttonId), "Id");
    gtk_widget_set_hexpand(buttonId, TRUE);

    g_signal_connect(G_OBJECT(buttonId), "clicked", G_CALLBACK(
  callback_sort_click), (gpointer) sort_callback_data_new(
  ID, animals, table));
    gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonId), 0, 0, 1, 1);

    GtkWidget *buttonName = gtk_button_new();
    gtk_button_set_label(GTK_BUTTON(buttonName), "Imię");
    gtk_widget_set_hexpand(buttonName, TRUE);
    g_signal_connect(G_OBJECT(buttonName), "clicked", G_CALLBACK(
  callback_sort_click), (gpointer) sort_callback_data_new(
  NAME, animals, table));
    gtk_widget_set_margin_start(buttonName, 5);
    gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonName), 1, 0, 1, 1);

    GtkWidget *buttonSpecies = gtk_button_new();
    gtk_button_set_label(GTK_BUTTON(buttonSpecies), "Gatunek");
    gtk_widget_set_hexpand(buttonSpecies, TRUE);
    g_signal_connect(G_OBJECT(buttonSpecies), "clicked", G_CALLBACK(
  callback_sort_click),
                     (gpointer) sort_callback_data_new(
  SPECIES, animals, table));
    gtk_widget_set_margin_start(buttonSpecies, 5);
    gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonSpecies), 2, 0, 1, 1);

    GtkWidget* buttonAge = gtk_button_new();
    gtk_button_set_label(GTK_BUTTON(buttonAge), "Wiek");
    gtk_widget_set_hexpand(buttonAge, TRUE);
    g_signal_connect(G_OBJECT(buttonAge), "clicked", G_CALLBACK(
  callback_sort_click),
                     (gpointer) sort_callback_data_new(AGE, animals, table));
    gtk_widget_set_margin_start(buttonAge, 5);
    gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonAge), 3, 0, 1, 1);

    GtkWidget* buttonComment = gtk_button_new();
    gtk_button_set_label(GTK_BUTTON(buttonComment), "Komentarz");
    gtk_widget_set_hexpand(buttonComment, TRUE);
    g_signal_connect(G_OBJECT(buttonComment), "clicked", G_CALLBACK(
  callback_sort_click),
                     (gpointer) sort_callback_data_new(
  COMMENT, animals, table));
    gtk_widget_set_margin_start(buttonComment, 5);
    gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonComment), 4, 0, 1, 1);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.1.4 callback_remove_animal()

```
void callback_remove_animal (
             GtkWidget * widget,
             gpointer callback_data )
```

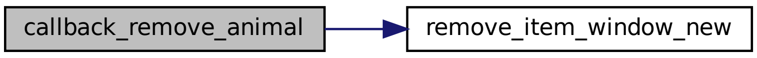
Definition at line 85 of file main_window.c.

References remove_item_window_new().

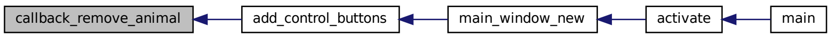
Referenced by add_control_buttons().

```
         {
      gtk_widget_show_all(remove_item_window_new((GtkApplication*)callback_data));
  }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.1.5 callback_sort_click()

```
void callback_sort_click (
             GtkWidget * widget,
             gpointer callback_data )
```

On click on any of the header buttons

Parameters

widget	
callback_data	an instance of SortCallbackData

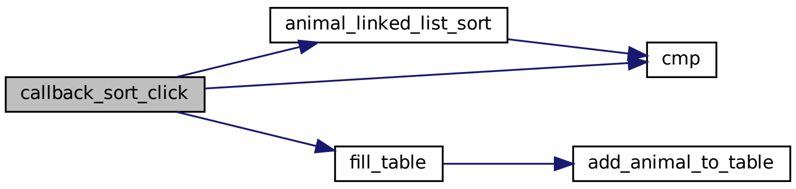
Definition at line 197 of file main_window.c.

References animal_linked_list_sort(), cmp(), SortCallbackData::col, fill_table(), SortCallbackData::list, sort_asc, sort_by, and SortCallbackData::table.

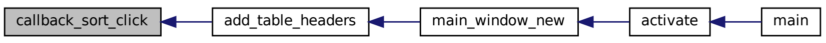
Referenced by add_table_headers().

```
{
    SortCallbackData* cd = (SortCallbackData*) callback_data;
    if(sort_by == cd->col) sort_asc = !sort_asc;
    sort_by = cd->col;
    animal_linked_list_sort(cd->list, cmp);
    fill_table(cd->table, cd->list);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.1.6 cmp()

```
int cmp (
            Animal * a,
            Animal * b )
```

Animal comparator Used to sort the table.

Parameters

a	
b	

Returns

boolean, true if a and b should be swapped

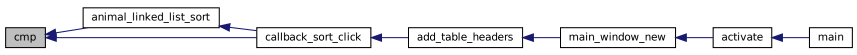
Definition at line 124 of file main_window.c.

References Animal::age, AGE, Animal::comment, COMMENT, Animal::id, ID, Animal::name, NAME, sort_asc, sort_by, Animal::species, and SPECIES.

Referenced by animal_linked_list_sort(), and callback_sort_click().

```
124                          {
125     int res = 0;
126     if(!sort_asc) {
127         Animal* t = a;
128         a = b;
129         b = t;
130     }
131     if(sort_by == ID) res = a->id > b->id;
132     if(sort_by == AGE) res = a->age > b->age;
133     if(sort_by == NAME) res = strcmp(a->name, b->name) > 0;
134     if(sort_by == SPECIES) res = strcmp(a->species, b->
      species) > 0;
135     if(sort_by == COMMENT) res = strcmp(a->comment, b->
      comment) > 0;
136     return res;
137 }
```

Here is the caller graph for this function:



5.10.1.7 fill_table()

```
void fill_table (
            GtkWidget * table,
            AnimalLinkedList * animals )
```

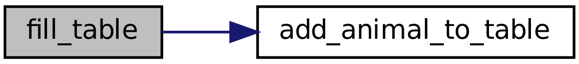
Definition at line 264 of file main_window.c.

References add_animal_to_table(), AnimalLinkedList::firstItem, AnimalLinkedListItem::next, AnimalLinkedList::size, and AnimalLinkedListItem::value.

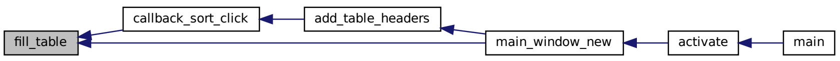
Referenced by callback_sort_click(), and main_window_new().

```
266 {
267     GList *children, *iter;
268 
269     children = gtk_container_get_children(GTK_CONTAINER(table));
270     for(iter = children; iter != NULL; iter = g_list_next(iter))
271         gtk_widget_destroy(GTK_WIDGET(iter->data));
272     g_list_free(children);
273 
274     AnimalLinkedListItem* cur = animals->firstItem;
275     for(int i=0; i<animals->size; ++i){
276         add_animal_to_table(table, cur->value, i+1);
277         cur = cur->next;
278     }
279 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.1.8 main_window_new()

```
GtkWidget* main_window_new (
              GtkApplication * app )
```

Initialize the main window.

Parameters

app	Application

Returns

main window, not shown yet

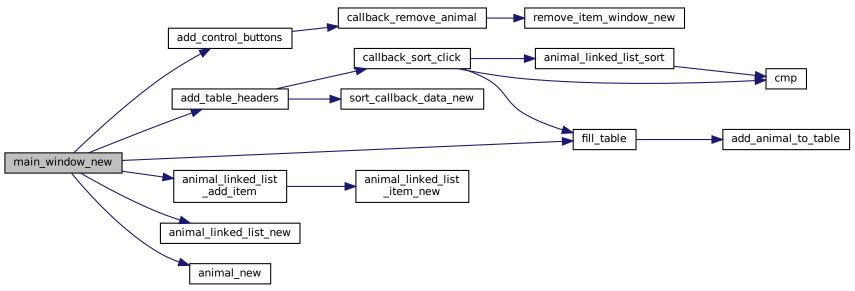
Definition at line 49 of file main_window.c.

References add_control_buttons(), add_table_headers(), animal_linked_list_add_item(), animal_linked_list_new(), animal_new(), and fill_table().

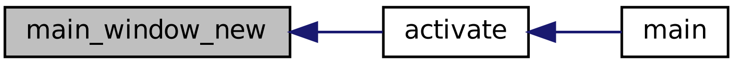
Referenced by activate().

```
{

    GtkWidget* window;
    GtkWidget* mainContainer;

    window = gtk_application_window_new (app);
    gtk_window_set_title (GTK_WINDOW (window), "Ogród zoologiczny");
    gtk_window_set_default_size (GTK_WINDOW (window), 500, 500);

    mainContainer = gtk_grid_new();
    gtk_container_add(GTK_CONTAINER (window), GTK_WIDGET (mainContainer));

    GtkWidget* containerTable;
    containerTable = gtk_scrolled_window_new(NULL, NULL);
    gtk_widget_set_hexpand(containerTable, 1);
    gtk_widget_set_vexpand(containerTable, 1);
    GtkWidget* table;
    table = gtk_grid_new();
    AnimalLinkedList* animals = animal_linked_list_new();

    for(int i=0; i<200; ++i){
        Animal* a = animal_new(i, "Blazej", "Wielblad", i/4+3, "Je orzeszki");
        animal_linked_list_add_item(animals, a);
    }
    gtk_container_add(GTK_CONTAINER(containerTable), GTK_WIDGET(table));
    gtk_grid_set_column_homogeneous(GTK_GRID(table), gtk_true());
    gtk_grid_set_column_homogeneous(GTK_GRID(mainContainer), gtk_true());
    add_table_headers(mainContainer, table, animals);
    fill_table(table, animals);
    gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (containerTable), 0, 1, 5, 1);
    add_control_buttons(app, mainContainer);

    return window;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.1.9 sort_callback_data_new()

```
SortCallbackData* sort_callback_data_new (
            COLUMN col,
            AnimalLinkedList * list,
            GtkWidget * table )
```

A container for data required to sort the table

Parameters

col	Column
list	List of animals
table	GtkGrid of the table

Returns

All arguments neatly packed in an SortCallbackData instance.

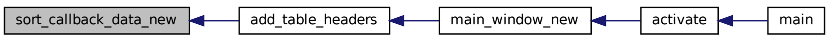
Definition at line 32 of file main_window.c.

References SortCallbackData::col, SortCallbackData::list, and SortCallbackData::table.

Referenced by add_table_headers().

```
                                        {
     SortCallbackData* sortCallbackData = malloc(sizeof(
    SortCallbackData));
     sortCallbackData->col = col;
     sortCallbackData->list = list;
     sortCallbackData->table = table;
     return sortCallbackData;
 }
```

Here is the caller graph for this function:



5.10.2 Variable Documentation

5.10.2.1 sort_asc

```
int sort_asc = 1
```

Definition at line 113 of file main_window.c.

Referenced by callback_sort_click(), and cmp().

5.10.2.2 sort_by

```
int sort_by = -1
```

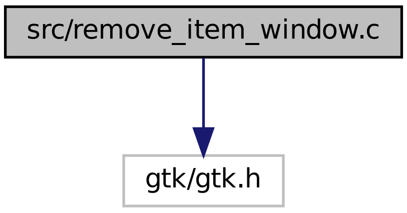
Definition at line 112 of file main_window.c.

Referenced by callback_sort_click(), and cmp().

5.11 src/remove_item_window.c File Reference

```
#include <gtk/gtk.h>
```

Include dependency graph for remove_item_window.c:



Functions

- GtkWidget ∗ remove_item_window_new (GtkApplication ∗app)

5.11.1 Function Documentation

5.11.1.1 remove_item_window_new()

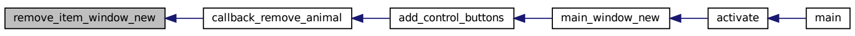
```
GtkWidget* remove_item_window_new (
            GtkApplication * app )
```

Definition at line 22 of file remove_item_window.c.

Referenced by callback_remove_animal().

```
22                                          {
23     GtkWidget* window;
24     GtkWidget* mainContainer;
25     mainContainer = gtk_grid_new();
26     window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
27     gtk_container_add(GTK_CONTAINER(window), mainContainer);
28     GtkEntry* entryId = gtk_entry_new();
29     gtk_entry_set_placeholder_text(entryId, "Id elementu");
30     gtk_grid_attach(GTK_GRID(mainContainer), entryId, 0, 0, 1, 1);
31     gtk_widget_set_hexpand(entryId, 1);
32     gtk_widget_set_vexpand(entryId, 1);
33     gtk_window_set_title (GTK_WINDOW (window), "Usuwanie elementu");
34 
35     return window;
36 }
```

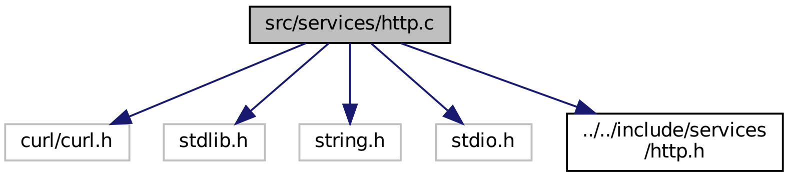
Here is the caller graph for this function:



5.12 src/services/http.c File Reference

```
#include <curl/curl.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "../../include/services/http.h"
```

Include dependency graph for http.c:



Functions

- size_t write_function (void ∗ptr, size_t size, size_t nmemb, HttpResponse ∗r)
- HttpResponse ∗ http_response_new ()
- HttpResponse ∗ http_get (char ∗url)

  Gets HTTP response for url.

5.12.1 Function Documentation

5.12.1.1 http_get()

```
HttpResponse* http_get (
          char * url )
```

Gets HTTP response for url.

Parameters

url	Site address

Returns

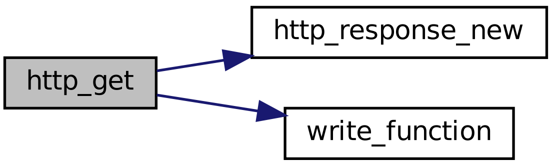
Filled HttpResponse

Definition at line 46 of file http.c.

References HttpResponse::code, HttpResponse::data, http_response_new(), and write_function().

```
                                    {
    CURL *curl;
    HttpResponse * res = http_response_new();

    curl = curl_easy_init();
    if(curl) {
        curl_easy_setopt(curl, CURLOPT_URL, url);
        /* example.com is redirected, so we tell libcurl to follow redirection */
        curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);
        curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, write_function);
        curl_easy_setopt(curl, CURLOPT_WRITEDATA, res);

        /* Perform the request, res->code will get the return code */
        res->code = curl_easy_perform(curl);
        /* Check for errors */
        if(res->code != CURLE_OK)
            fprintf(stderr, "curl_easy_perform() failed: %s\n",
                    curl_easy_strerror(res->code));
        else {
            /* Print data for debug */
            printf("Data: %s\n", res->data);
        }

        /* Cleanup */
        curl_easy_cleanup(curl);
    }
    return res;
}
```

Here is the call graph for this function:



5.12.1.2 http_response_new()

```
HttpResponse* http_response_new ( )
```

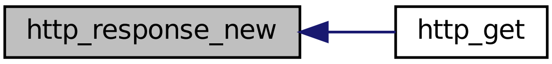
Definition at line 33 of file http.c.

References HttpResponse::data, and HttpResponse::size.

Referenced by http_get().

```
                                   {
    HttpResponse * res = malloc(sizeof(HttpResponse));
    res->size = 0;
    res->data = malloc(res->size+1);
    res->data[res->size] = '\0';
    return res;
};
```

Here is the caller graph for this function:



5.12.1.3 write_function()

```
size_t write_function (
            void * ptr,
            size_t size,
            size_t nmemb,
            HttpResponse * r )
```

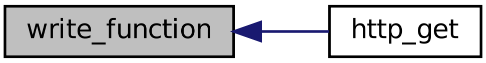
Definition at line 25 of file http.c.

References HttpResponse::data, and HttpResponse::size.

Referenced by http_get().

```
                                                                      {
    size_t new_len = r->size + size*nmemb;
    r->data= realloc(r->data, new_len+1);
    memcpy(r->data+r->size, ptr, size*nmemb);
    r->data[new_len] = '\0';
    return size*nmemb;
}
```

Here is the caller graph for this function:



Index