

MC833 - Projeto 1

Generated by Doxygen 1.9.3

1 Data Structure Index	1
1.1 Data Structures	1
2 File Index	3
2.1 File List	3
3 Data Structure Documentation	5
3.1 Catalog Struct Reference	5
3.1.1 Detailed Description	5
3.1.2 Field Documentation	5
3.1.2.1 movie_list	5
3.1.2.2 size	6
3.2 CatalogMovie Union Reference	6
3.2.1 Detailed Description	6
3.2.2 Field Documentation	6
3.2.2.1 catalog	6
3.2.2.2 movie	6
3.3 Movie Struct Reference	7
3.3.1 Detailed Description	7
3.3.2 Field Documentation	7
3.3.2.1 director_name	7
3.3.2.2 genre_list	7
3.3.2.3 id	8
3.3.2.4 num_genres	8
3.3.2.5 title	8
3.3.2.6 year	8
3.4 Payload Struct Reference	8
3.4.1 Detailed Description	9
3.4.2 Field Documentation	9
3.4.2.1 movie	9
3.4.2.2 op	9
3.5 Response Struct Reference	9
3.5.1 Detailed Description	10
3.5.2 Field Documentation	10
3.5.2.1 data	10
4 File Documentation	11
4.1 src/data/catalog.c File Reference	11
4.1.1 Function Documentation	11
4.1.1.1 add_movie()	11
4.1.1.2 delete_movie()	12
4.1.1.3 update_movie()	12
4.2 catalog.c	12

4.3 src/data/catalog.h File Reference	13
4.3.1 Macro Definition Documentation	13
4.3.1.1 MAX_MOVIES	13
4.3.2 Function Documentation	14
4.3.2.1 add_movie()	14
4.3.2.2 delete_movie()	14
4.3.2.3 update_movie()	14
4.4 catalog.h	15
4.5 src/data/movie.c File Reference	15
4.5.1 Function Documentation	15
4.5.1.1 add_genre()	15
4.5.1.2 contains_genre()	16
4.5.1.3 create_movie()	16
4.6 movie.c	17
4.7 src/data/movie.h File Reference	17
4.7.1 Macro Definition Documentation	18
4.7.1.1 MAX_MOVIE_GENRES	18
4.7.2 Function Documentation	18
4.7.2.1 add_genre()	18
4.7.2.2 contains_genre()	18
4.7.2.3 create_movie()	19
4.8 movie.h	19
4.9 src/tcp/client.c File Reference	20
4.9.1 Macro Definition Documentation	21
4.9.1.1 PORT	21
4.9.2 Function Documentation	21
4.9.2.1 get_movies()	21
4.9.2.2 handle_get()	21
4.9.2.3 handle_user()	21
4.9.2.4 list_all_info()	21
4.9.2.5 list_info_by_genre()	22
4.9.2.6 list_info_by_id()	22
4.9.2.7 list_titles()	22
4.9.2.8 main()	23
4.9.2.9 post_movie()	23
4.9.2.10 print_all_info()	23
4.9.2.11 print_menu()	24
4.9.2.12 put_genre()	24
4.9.2.13 remove_movie()	24
4.9.2.14 send_exit()	24
4.9.2.15 sigint_handler()	25
4.9.2.16 wait_for_enter()	25

4.9.3 Variable Documentation	25
4.9.3.1 get_handlers	25
4.9.3.2 handlers	25
4.9.3.3 SOCKFD	25
4.10 client.c	26
4.11 src/tcp/server.c File Reference	29
4.11.1 Macro Definition Documentation	30
4.11.1.1 BACKLOG	30
4.11.1.2 PORT	30
4.11.2 Function Documentation	30
4.11.2.1 backup()	31
4.11.2.2 del_movie()	31
4.11.2.3 get_movie()	31
4.11.2.4 handle_client()	31
4.11.2.5 load_backup()	32
4.11.2.6 main()	32
4.11.2.7 post_movie()	32
4.11.2.8 put_movie()	32
4.11.2.9 sigchld_handler()	33
4.11.2.10 sigint_handler()	33
4.11.3 Variable Documentation	33
4.11.3.1 CATALOG	33
4.11.3.2 handlers	33
4.11.3.3 SOCKFD	34
4.12 server.c	34
4.13 src/utls/net_utls.c File Reference	36
4.13.1 Function Documentation	36
4.13.1.1 get_in_addr()	36
4.14 net_utls.c	37
4.15 src/utls/net_utls.h File Reference	37
4.15.1 Macro Definition Documentation	38
4.15.1.1 ALL	38
4.15.2 Enumeration Type Documentation	38
4.15.2.1 Operation	38
4.15.3 Function Documentation	38
4.15.3.1 get_in_addr()	38
4.16 net_utls.h	39
4.17 src/utls/utls.h File Reference	39
4.17.1 Macro Definition Documentation	39
4.17.1.1 MAX_STR_LEN	39
4.18 utls.h	40

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Catalog	5
CatalogMovie	6
Movie	7
Payload	8
Response	9

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

src/data/ catalog.c	11
src/data/ catalog.h	13
src/data/ movie.c	15
src/data/ movie.h	17
src/tcp/ client.c	20
src/tcp/ server.c	29
src/utls/ net_utls.c	36
src/utls/ net_utls.h	37
src/utls/ utls.h	39

Chapter 3

Data Structure Documentation

3.1 Catalog Struct Reference

```
#include <catalog.h>
```

Data Fields

- char [size](#)
- [Movie](#) [movie_list](#) [[MAX_MOVIES](#)]

3.1.1 Detailed Description

Lista de filmes (movies).

Definition at line [12](#) of file [catalog.h](#).

3.1.2 Field Documentation

3.1.2.1 [movie_list](#)

```
Movie movie\_list [MAX\_MOVIES]
```

Lista que armazena os filmes

Definition at line [14](#) of file [catalog.h](#).

3.1.2.2 size

`char size`

Quantidade de filmes na lista

Definition at line 13 of file [catalog.h](#).

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

- [src/data/catalog.h](#)

3.2 CatalogMovie Union Reference

```
#include <net_utils.h>
```

Data Fields

- [Catalog catalog](#)
- [Movie movie](#)

3.2.1 Detailed Description

Definition at line 22 of file [net_utils.h](#).

3.2.2 Field Documentation

3.2.2.1 catalog

[Catalog](#) catalog

Definition at line 23 of file [net_utils.h](#).

3.2.2.2 movie

[Movie](#) movie

Definition at line 24 of file [net_utils.h](#).

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

- [src/utlis/net_utils.h](#)

3.3 Movie Struct Reference

```
#include <movie.h>
```

Data Fields

- int [id](#)
- char [title](#) [[MAX_STR_LEN](#)]
- int [num_genres](#)
- char [genre_list](#) [[MAX_MOVIE_GENRES](#)][[MAX_STR_LEN](#)]
- char [director_name](#) [[MAX_STR_LEN](#)]
- int [year](#)

3.3.1 Detailed Description

Struct para armazenar as informações referentes a um filme.

Definition at line [12](#) of file [movie.h](#).

3.3.2 Field Documentation

3.3.2.1 [director_name](#)

```
char director_name[MAX\_STR\_LEN]
```

Nome do diretor

Definition at line [17](#) of file [movie.h](#).

3.3.2.2 [genre_list](#)

```
char genre_list[MAX\_MOVIE\_GENRES][MAX\_STR\_LEN]
```

Lista de gêneros do filme

Definition at line [16](#) of file [movie.h](#).

3.3.2.3 id

```
int id
```

Identificador único

Definition at line 13 of file [movie.h](#).

3.3.2.4 num_genres

```
int num_genres
```

Quantidade de gêneros na sua lista de gêneros

Definition at line 15 of file [movie.h](#).

3.3.2.5 title

```
char title[MAX_STR_LEN]
```

Título do filme

Definition at line 14 of file [movie.h](#).

3.3.2.6 year

```
int year
```

Ano de publicação

Definition at line 18 of file [movie.h](#).

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

- [src/data/movie.h](#)

3.4 Payload Struct Reference

```
#include <net_utils.h>
```

Data Fields

- [Operation op](#)
- [Movie movie](#)

3.4.1 Detailed Description

Struct utilizada para enviar as informações para o servidor.

Definition at line 17 of file [net_utils.h](#).

3.4.2 Field Documentation

3.4.2.1 movie

[Movie](#) movie

Informações de filme a serem utilizadas pela operação

Definition at line 19 of file [net_utils.h](#).

3.4.2.2 op

[Operation](#) op

Tipo de operação a ser feito.

Definition at line 18 of file [net_utils.h](#).

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

- [src/utls/net_utils.h](#)

3.5 Response Struct Reference

```
#include <net_utils.h>
```

Data Fields

- [CatalogMovie data](#)

3.5.1 Detailed Description

Struct utilizada como resposta do servidor.

Definition at line 31 of file [net_utils.h](#).

3.5.2 Field Documentation

3.5.2.1 data

[CatalogMovie](#) data

Tipo de data a ser retornada.

Definition at line 32 of file [net_utils.h](#).

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

- [src/utlis/net_utils.h](#)

Chapter 4

File Documentation

4.1 src/data/catalog.c File Reference

```
#include "catalog.h"
#include <string.h>
```

Functions

- void `add_movie` (`Catalog` *catalog, `Movie` movie)
Função para adicionar um filme a um catálogo.
- void `update_movie` (`Catalog` *catalog, `Movie` movie)
Modifica um filme no catálogo com as informações passadas.
- void `delete_movie` (`Catalog` *catalog, `Movie` movie)
Deleta um movie baseado no seu ID.

4.1.1 Function Documentation

4.1.1.1 add_movie()

```
void add_movie (
    Catalog * catalog,
    Movie movie )
```

Função para adicionar um filme a um catálogo.

Caso não haja espaço para adicioná-lo, nada é feito.

Parameters

in, out	<code>catalog</code>	Catálogo a ser modificado.
in	<code>movie</code>	Filme a ser adicionado.

Definition at line 10 of file [catalog.c](#).

4.1.1.2 delete_movie()

```
void delete_movie (
    Catalog * catalog,
    Movie movie )
```

Deleta um movie baseado no seu ID.

Para deleção, apenas sobrecrevemo-o.

Parameters

in, out	<i>catalog</i>	Catálogo da onde o movie será tirado.
in	<i>movie</i>	Movie com o ID do filme a ser removido.

Definition at line 37 of file [catalog.c](#).

4.1.1.3 update_movie()

```
void update_movie (
    Catalog * catalog,
    Movie movie )
```

Modifica um filme no catálogo com as informações passadas.

Baseado no movie->id, modifica os dados deste no catálogo. Por enquanto, apenas adição de gênero é suportada.

Parameters

in, out	<i>catalog</i>	Lista de filmes contendo o filme a ser alterado.
in	<i>movie</i>	As informações a serem atualizadas estarão preenchidas. O resto deve estar vazio enquanto que o ID é obrigatório.

Definition at line 23 of file [catalog.c](#).

4.2 catalog.c

[Go to the documentation of this file.](#)

```
00001 #include "catalog.h"
00002 #include <string.h>
00003
00010 void add_movie(Catalog *catalog, Movie movie) {
00011     if (catalog->size < MAX_MOVIES)
00012         catalog->movie_list[catalog->size++] = movie;
```

```

00013 }
00014
00023 void update_movie(Catalog *catalog, Movie movie) {
00024     for (int i = 0; i < catalog->size; i++)
00025         if (movie.id == catalog->movie_list[i].id) {
00026             add_genre(&catalog->movie_list[i], movie.genre_list[0]);
00027             return;
00028         }
00029 }
00030
00037 void delete_movie(Catalog *catalog, Movie movie) {
00038     for (int i = 0; i < catalog->size; i++)
00039         if (movie.id == catalog->movie_list[i].id) {
00040             memcpy(&catalog->movie_list[i], &catalog->movie_list[i + 1],
00041                 (catalog->size - i - 1) * sizeof(Movie));
00042             catalog->size--;
00043             return;
00044         }
00045 }

```

4.3 src/data/catalog.h File Reference

```
#include "movie.h"
```

Data Structures

- struct [Catalog](#)

Macros

- #define [MAX_MOVIES](#) 50

Functions

- void [add_movie](#) ([Catalog](#) *catalog, [Movie](#) movie)
Função para adicionar um filme a um catálogo.
- void [update_movie](#) ([Catalog](#) *catalog, [Movie](#) movie)
Modifica um filme no catálogo com as informações passadas.
- void [delete_movie](#) ([Catalog](#) *catalog, [Movie](#) movie)
Deleta um movie baseado no seu ID.

4.3.1 Macro Definition Documentation

4.3.1.1 MAX_MOVIES

```
#define MAX_MOVIES 50
```

Definition at line 6 of file [catalog.h](#).

4.3.2 Function Documentation

4.3.2.1 add_movie()

```
void add_movie (
    Catalog * catalog,
    Movie movie )
```

Função para adicionar um filme a um catálogo.

Caso não haja espaço para adicioná-lo, nada é feito.

Parameters

in, out	<i>catalog</i>	Catálogo a ser modificado.
in	<i>movie</i>	Filme a ser adicionado.

Definition at line 10 of file [catalog.c](#).

4.3.2.2 delete_movie()

```
void delete_movie (
    Catalog * catalog,
    Movie movie )
```

Deleta um movie baseado no seu ID.

Para deleção, apenas sobrecrevemo-o.

Parameters

in, out	<i>catalog</i>	Catálogo da onde o movie será tirado.
in	<i>movie</i>	Movie com o ID do filme a ser removido.

Definition at line 37 of file [catalog.c](#).

4.3.2.3 update_movie()

```
void update_movie (
    Catalog * catalog,
    Movie movie )
```

Modifica um filme no catálogo com as informações passadas.

Baseado no movie->id, modifica os dados deste no catálogo. Por enquanto, apenas adição de gênero é suportada.

Parameters

in, out	<i>catalog</i>	Lista de filmes contendo o filme a ser alterado.
in	<i>movie</i>	As informações a serem atualizadas estarão preenchidas. O resto deve estar vazio enquanto que o ID é obrigatório.

Definition at line 23 of file [catalog.c](#).

4.4 catalog.h

[Go to the documentation of this file.](#)

```

00001 #ifndef MC833_PROJETO_CATALOG_H
00002 #define MC833_PROJETO_CATALOG_H
00003
00004 #include "movie.h"
00005
00006 #define MAX_MOVIES 50
00007
00012 typedef struct {
00013     char size;
00014     Movie movie_list[MAX_MOVIES];
00015 } Catalog;
00016
00017 void add_movie(Catalog *catalog, Movie movie);
00018 void update_movie(Catalog *catalog, Movie movie);
00019 void delete_movie(Catalog *catalog, Movie movie);
00020
00021 #endif // MC833_PROJETO_CATALOG_H

```

4.5 src/data/movie.c File Reference

```

#include "movie.h"
#include <stdlib.h>
#include <string.h>

```

Functions

- [Movie * create_movie](#) (int id, char *title, char *genre, char *director, int year)
Função para criar uma struct movie.
- void [add_genre](#) (Movie *movie, char *genre)
Adiciona um gênero a lista de gêneros do filme.
- char [contains_genre](#) (Movie *movie, char *genre)
Confere se o gênero está na lista de gêneros.

4.5.1 Function Documentation

4.5.1.1 add_genre()

```

void add_genre (
    Movie * movie,
    char * genre )

```

Adiciona um gênero a lista de gêneros do filme.

Parameters

in, out	<i>movie</i>	Filme a ter um gênero inserido.
out	<i>genre</i>	Gênero a ser inserido.

Definition at line 32 of file [movie.c](#).

4.5.1.2 contains_genre()

```
char contains_genre (
    Movie * movie,
    char * genre )
```

Confere se o gênero está na lista de gêneros.

Usa comparação de strings para determinar se o gênero fornecido está ou não presente no filme.

Parameters

in, out	<i>movie</i>	Filme a ser testado.
out	<i>genre</i>	Genero a ser procurado

Returns

1 caso o filme seja do gênero passado, 0 caso contrário.

Definition at line 44 of file [movie.c](#).

4.5.1.3 create_movie()

```
Movie * create_movie (
    int id,
    char * title,
    char * genre,
    char * director,
    int year )
```

Função para criar uma struct movie.

Aloca um espaço em memória para a struct e preenche com os dados passados.

Parameters

in	<i>id</i>	
out	<i>title</i>	Título do filme a ser criado.
out	<i>genre</i>	Gêneros iniciais para o filme.
out	<i>director</i>	Nome do diretor.
in	<i>year</i>	Ano de publicação

Returns

Ponteiro para a nova struct movie.

Definition at line 16 of file [movie.c](#).

4.6 movie.c

[Go to the documentation of this file.](#)

```

00001 #include "movie.h"
00002 #include <stdlib.h>
00003 #include <string.h>
00004
00016 Movie *create_movie(int id, char *title, char *genre, char *director,
00017                     int year) {
00018     Movie *new = calloc(1, sizeof(Movie));
00019     new->id = id; // TODO acho que temos que mudar esse compostamento
00020     memcpy(new->title, title, MAX_STR_LEN);
00021     memcpy(new->genre_list[new->num_genres++], genre, MAX_STR_LEN);
00022     memcpy(new->director_name, director, MAX_STR_LEN);
00023     new->year = year;
00024     return new;
00025 }
00026
00032 void add_genre(Movie *movie, char *genre) {
00033     memcpy(movie->genre_list[movie->num_genres++], genre, MAX_STR_LEN);
00034 }
00035
00044 char contains_genre(Movie *movie, char *genre) {
00045     for (int i = 0; i < movie->num_genres; i++)
00046         if (strstr(movie->genre_list[i], genre))
00047             return 1;
00048     return 0;
00049 }

```

4.7 src/data/movie.h File Reference

```
#include "../utils/utils.h"
```

Data Structures

- struct [Movie](#)

Macros

- #define [MAX_MOVIE_GENRES](#) 10

Functions

- [Movie * create_movie](#) (int id, char *title, char *genre, char *director, int year)
Função para criar uma struct movie.
- void [add_genre](#) ([Movie](#) *movie, char *genre)
Adiciona um gênero a lista de gêneros do filme.
- char [contains_genre](#) ([Movie](#) *movie, char *genre)
Confere se o gênero está na lista de gêneros.

4.7.1 Macro Definition Documentation

4.7.1.1 MAX_MOVIE_GENRES

```
#define MAX_MOVIE_GENRES 10
```

Definition at line 6 of file [movie.h](#).

4.7.2 Function Documentation

4.7.2.1 add_genre()

```
void add_genre (  
    Movie * movie,  
    char * genre )
```

Adiciona um gênero a lista de gêneros do filme.

Parameters

in, out	<i>movie</i>	Filme a ter um gênero inserido.
out	<i>genre</i>	Gênero a ser inserido.

Definition at line 32 of file [movie.c](#).

4.7.2.2 contains_genre()

```
char contains_genre (  
    Movie * movie,  
    char * genre )
```

Confere se o gênero está na lista de gêneros.

Usa comparação de strings para determinar se o gênero fornecido está ou não presente no filme.

Parameters

in, out	<i>movie</i>	Filme a ser testado.
out	<i>genre</i>	Genero a ser procurado

Returns

1 caso o filme seja do gênero passado, 0 caso contrário.

Definition at line 44 of file [movie.c](#).

4.7.2.3 create_movie()

```
Movie * create_movie (
    int id,
    char * title,
    char * genre,
    char * director,
    int year )
```

Função para criar uma struct movie.

Aloca um espaço em memória para a struct e preenche com os dados passados.

Parameters

in	<i>id</i>	
out	<i>title</i>	Título do filme a ser criado.
out	<i>genre</i>	Gêneros iniciais para o filme.
out	<i>director</i>	Nome do diretor.
in	<i>year</i>	Ano de publicação

Returns

Ponteiro para a nova struct movie.

Definition at line 16 of file [movie.c](#).

4.8 movie.h

[Go to the documentation of this file.](#)

```
00001 #ifndef MC833_PROJETO_MOVIE_H
00002 #define MC833_PROJETO_MOVIE_H
00003
00004 #include "../utils/utils.h"
00005
00006 #define MAX_MOVIE_GENRES 10
00007
00012 typedef struct {
00013     int id;
00014     char title[MAX_STR_LEN];
00015     int num_genres;
00016     char genre_list[MAX_MOVIE_GENRES][MAX_STR_LEN];
00017     char director_name[MAX_STR_LEN];
00018     int year;
00019     // imagem capa;
00020 } Movie;
00021
00022 Movie *create_movie(int id, char *title, char *genre, char *director, int year);
00023 void add_genre(Movie *movie, char *genre);
00024 char contains_genre(Movie *movie, char *genre);
00025
00026 #endif // MC833_PROJETO_MOVIE_H
```

4.9 src/tcp/client.c File Reference

```
#include "../utils/net_utils.h"
#include <arpa/inet.h>
#include <netdb.h>
#include <netinet/in.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>
```

Macros

- `#define PORT "3490"`

Functions

- void `wait_for_enter` ()
- void `print_menu` ()
- void `send_exit` ()
- void `handle_get` (char cmd)
- void `handle_user` ()
- void `sigint_handler` (int sig_num)
- int `main` (int argc, char *argv[])

Operações do servidor

=====

Funções para montar o payload compatível com as operações do servidor.

- `Payload get_movies` ()
Listar todos os títulos, junto aos seus respectivos identificadores.
- `Payload post_movie` ()
Cadastrar um novo filme.
- `Payload put_genre` ()
Acrescentar um novo gênero a um filme.
- `Payload remove_movie` ()
Remover um filme do catálogo a partir de seu ID.

Funções de impressão

=====

Funções auxiliares para imprimir os resultados obtidos.

- void `list_titles` (`Response` response)
Imprimir os filmes obtidos.
- void `print_all_info` (`Movie` movie)
Imprime todas as informações de um filme.
- void `list_all_info` (`Response` response)
Summary.
- void `list_info_by_genre` (`Response` response)
Função para imprimir apenas os filmes de um gênero.
- void `list_info_by_id` (`Response` response)
Função para imprimir apenas o filme com um id.

Variables

- int [SOCKFD](#)
- [Payload](#)(* [handlers](#) [])()
- void(* [get_handlers](#) [])([Response](#))

4.9.1 Macro Definition Documentation

4.9.1.1 PORT

```
#define PORT "3490"
```

Definition at line 16 of file [client.c](#).

4.9.2 Function Documentation

4.9.2.1 get_movies()

```
Payload get_movies ( )
```

Listar todos os títulos, junto aos seus respectivos identificadores.

Returns

Struct [Payload](#) com as informações a serem colocadas

Definition at line 37 of file [client.c](#).

4.9.2.2 handle_get()

```
void handle_get (
    char cmd )
```

Definition at line 269 of file [client.c](#).

4.9.2.3 handle_user()

```
void handle_user ( )
```

Definition at line 276 of file [client.c](#).

4.9.2.4 list_all_info()

```
void list_all_info (
    Response response )
```

Summary.

Description

Parameters

in	<i>response</i>	Description
----	-----------------	-------------

Definition at line 178 of file [client.c](#).

4.9.2.5 list_info_by_genre()

```
void list_info_by_genre (
    Response response )
```

Função para imprimir apenas os filmes de um gênero.

A partir do catálogo solicitado ao servidor, imprimimos apenas aqueles que possuem em sua lista de gêneros, o genero fornecido.

Parameters

in	<i>response</i>	Resposta do servidor com o catálogo.
----	-----------------	--------------------------------------

Definition at line 193 of file [client.c](#).

4.9.2.6 list_info_by_id()

```
void list_info_by_id (
    Response response )
```

Função para imprimir apenas o filme com um id.

A partir do catálogo solicitado ao servidor, imprimimos apenas aquele que possui id igual ao fornecido.

Parameters

in	<i>response</i>	Resposta do servidor com o catálogo.
----	-----------------	--------------------------------------

Definition at line 221 of file [client.c](#).

4.9.2.7 list_titles()

```
void list_titles (
    Response response )
```

Imprimir os filmes obtidos.

Imprime no formato "-> ID - TÌTULO\n".

Parameters

in	<i>response</i>	Resposta com o catálogo a ser impresso.
----	-----------------	---

Definition at line 150 of file [client.c](#).

4.9.2.8 main()

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

Definition at line 311 of file [client.c](#).

4.9.2.9 post_movie()

```
Payload post_movie ( )
```

Cadastrar um novo filme.

O identificador numérico será definido pelo sistema

Returns

Struct [Payload](#) com as informações a serem colocadas.

Definition at line 50 of file [client.c](#).

4.9.2.10 print_all_info()

```
void print_all_info (
    Movie movie )
```

Imprime todas as informações de um filme.

Fotmato " -> ID - ANO TÍTULO DIRETOR | GÊNEROS".

Parameters

in	<i>movie</i>	Filme cujas informações devemos imprimir.
----	--------------	---

Definition at line 165 of file [client.c](#).

4.9.2.11 print_menu()

```
void print_menu ( )
```

Definition at line 249 of file [client.c](#).

4.9.2.12 put_genre()

```
Payload put_genre ( )
```

Acrescentar um novo gênero a um filme.

Returns

Struct [Payload](#) com as informações a serem colocadas.

Definition at line 92 of file [client.c](#).

4.9.2.13 remove_movie()

```
Payload remove_movie ( )
```

Remover um filme do catálogo a partir de seu ID.

Description

Returns

Description

Definition at line 122 of file [client.c](#).

4.9.2.14 send_exit()

```
void send_exit ( )
```

Definition at line 262 of file [client.c](#).

4.9.2.15 sigint_handler()

```
void sigint_handler (
    int sig_num )
```

Definition at line 302 of file [client.c](#).

4.9.2.16 wait_for_enter()

```
void wait_for_enter ( )
```

Definition at line 20 of file [client.c](#).

4.9.3 Variable Documentation

4.9.3.1 get_handlers

```
void(* get_handlers[ ]) (Response) (
    Response )
```

Initial value:

```
= {NULL, NULL,
    list_titles, list_info_by_genre,
    list_all_info, list_info_by_id}
```

Definition at line 245 of file [client.c](#).

4.9.3.2 handlers

```
Payload(* handlers[ ]) () ( )
```

Initial value:

```
= {post_movie, put_genre, get_movies, get_movies,
    get_movies, get_movies, remove_movie}
```

=====

Definition at line 243 of file [client.c](#).

4.9.3.3 SOCKFD

```
int SOCKFD
```

Definition at line 18 of file [client.c](#).

4.10 client.c

[Go to the documentation of this file.](#)

```

00001 /*
00002 ** client.c -- a stream socket client demo
00003 */
00004
00005 #include "../utils/net_utils.h"
00006 #include <arpa/inet.h>
00007 #include <arpa/inet.h>
00008 #include <netinet/in.h>
00009 #include <signal.h>
00010 #include <stdio.h>
00011 #include <stdlib.h>
00012 #include <string.h>
00013 #include <sys/socket.h>
00014 #include <unistd.h>
00015
00016 #define PORT "3490" // the port client will be connecting to
00017
00018 int SOCKFD;
00019
00020 void wait_for_enter() {
00021     static char aux[MAX_STR_LEN];
00022     printf("Aperte enter para continuar...");
00023     fgets(aux, MAX_STR_LEN, stdin);
00024 }
00025
00037 Payload get_movies() {
00038     Payload ret;
00039     memset(&ret, 0, sizeof(Payload));
00040     ret.op = GET;
00041     ret.movie.id = ALL;
00042     return ret;
00043 }
00044
00050 Payload post_movie() {
00051     Payload ret;
00052     memset(&ret, 0, sizeof(Payload));
00053     ret.op = POST;
00054
00055     system("clear");
00056     printf("Cadastro de filme");
00057
00058     printf("\nDigite o id do filme: ");
00059     scanf("%d", &ret.movie.id);
00060     getchar(); // ignores the leading \n
00061
00062     printf("Digite o título do filme: ");
00063     fgets(ret.movie.title, MAX_STR_LEN, stdin);
00064     ret.movie.title[strcspn(ret.movie.title, "\n")] = '\0';
00065
00066     printf("Digite o número de gêneros do filme: ");
00067     scanf("%d", &ret.movie.num_genres);
00068     getchar(); // ignores the leading \n
00069
00070     for (int i = 0; i < ret.movie.num_genres; i++) {
00071         printf("Digite o %dº gênero do filme: ", i + 1);
00072         fgets(ret.movie.genre_list[i], MAX_STR_LEN, stdin);
00073         ret.movie.genre_list[i][strcspn(ret.movie.genre_list[i], "\n")] = '\0';
00074     }
00075
00076     printf("Digite o nome do diretor do filme: ");
00077     fgets(ret.movie.director_name, MAX_STR_LEN, stdin);
00078     ret.movie.director_name[strcspn(ret.movie.director_name, "\n")] = '\0';
00079
00080     printf("Digite o ano do filme: ");
00081     scanf("%d", &ret.movie.year);
00082     getchar(); // ignores the leading \n
00083
00084     return ret;
00085 }
00086
00092 Payload put_genre() {
00093     Payload ret;
00094     memset(&ret, 0, sizeof(Payload));
00095     ret.op = PUT;
00096
00097     system("clear");
00098     printf("Adição de gênero em filme");
00099
00100     printf("\nDigite o id do filme: ");
00101     scanf("%d", &ret.movie.id);
00102
00103     // BUG Não precisa ser mais de um, caso dê problema sugiro simplificarmos.

```



```

00104     printf("Digite o número de gêneros que deseja adicionar a esse filme: ");
00105     scanf("%d", &ret.movie.num_genres);
00106     getchar(); // ignores the leading \n
00107
00108     for (int i = 0; i < ret.movie.num_genres; i++) {
00109         printf("Digite o %dº novo gênero desse filme: ", i + 1);
00110         fgets(ret.movie.genre_list[i], MAX_STR_LEN, stdin);
00111         ret.movie.genre_list[i][strcspn(ret.movie.genre_list[i], "\n")] = '\0';
00112     }
00113
00114     return ret;
00115 }
00116
00122 Payload remove_movie() {
00123     Payload ret;
00124     memset(&ret, 0, sizeof(Payload));
00125     ret.op = DEL;
00126
00127     system("clear");
00128     printf("Remover filme");
00129
00130     printf("\nDigite o id do filme a ser removido: ");
00131     scanf("%d", &ret.movie.id);
00132     getchar(); // ignores the leading \n
00133
00134     return ret;
00135 }
00150 void list_titles(Response response) {
00151     system("clear");
00152     printf("Lista de filmes:\n");
00153     printf(" -> id - título\n");
00154     for (int i = 0; i < response.data.catalog.size; i++)
00155         printf(" -> %d - %s\n", response.data.catalog.movie_list[i].id,
00156             response.data.catalog.movie_list[i].title);
00157     wait_for_enter();
00158 }
00159
00165 void print_all_info(Movie movie) {
00166     printf(" -> %d - (%d) %s by %s |", movie.id, movie.year, movie.title,
00167         movie.director_name);
00168     for (int i = 0; i < movie.num_genres; i++)
00169         printf(" %s", movie.genre_list[i]);
00170     printf("\n");
00171 }
00172
00178 void list_all_info(Response response) {
00179     system("clear");
00180     printf("Informações dos filmes:\n");
00181     printf(" -> id - (ano) título by diretor | Gêneros\n");
00182     for (int i = 0; i < response.data.catalog.size; i++)
00183         print_all_info(response.data.catalog.movie_list[i]);
00184     wait_for_enter();
00185 }
00186
00193 void list_info_by_genre(Response response) {
00194     char genre[MAX_STR_LEN];
00195     system("clear");
00196     printf("Digite o gênero: ");
00197     fgets(genre, MAX_STR_LEN, stdin);
00198     genre[strcspn(genre, "\n")] = '\0';
00199
00200     // REVIEW Do jeito que ele colocou na descrição, parecia que ele queria
00201     // que essa filtragem fosse feita pelo servidor.
00202     system("clear");
00203     printf("Lista de filmes:\n");
00204     printf(" -> título - nome do diretor - ano\n");
00205     Movie aux;
00206     for (int i = 0; i < response.data.catalog.size; i++) {
00207         aux = response.data.catalog.movie_list[i];
00208         if (contains_genre(&aux, genre))
00209             printf(" -> %s - %s - %d\n", aux.title, aux.director_name,
00210                 aux.year);
00211     }
00212     wait_for_enter();
00213 }
00214
00221 void list_info_by_id(Response response) {
00222     int id;
00223     system("clear");
00224     printf("Digite o id: ");
00225     scanf("%d", &id);
00226     getchar(); // ignores the leading \n
00227
00228     system("clear");
00229     printf(" -> id - (ano) título by diretor | Gêneros\n");
00230     for (int i = 0; i < response.data.catalog.size; i++)
00231         if (response.data.catalog.movie_list[i].id == id) {

```

```

00232         print_all_info(response.data.catalog.movie_list[i]);
00233         break;
00234     }
00235
00236     wait_for_enter();
00237 }
00243 Payload (*handlers[])() = {post_movie, put_genre, get_movies, get_movies,
00244                             get_movies, get_movies, remove_movie};
00245 void (*get_handlers[])(Response) = {NULL, NULL,
00246                                     list_titles, list_info_by_genre,
00247                                     list_all_info, list_info_by_id};
00248
00249 void print_menu() {
00250     system("clear");
00251     printf("0 - Cadastrar um novo filme");
00252     printf("\n1 - Acrescentar um novo gênero em um filme");
00253     printf("\n2 - Listar títulos");
00254     printf("\n3 - Listar informações por gênero");
00255     printf("\n4 - Listar todas as informações de todos os filmes");
00256     printf("\n5 - Listar todas as informações de um filme");
00257     printf("\n6 - Remover filme");
00258     printf("\ne - exit");
00259     printf("\nDigite um comando: ");
00260 }
00261
00262 void send_exit() {
00263     Payload payload;
00264     payload.op = EXIT;
00265     if (send(SOCKFD, &payload, sizeof(Payload), 0) == -1)
00266         perror("send");
00267 }
00268
00269 void handle_get(char cmd) {
00270     Response response;
00271     if (recv(SOCKFD, &response, sizeof(Response), 0) == -1)
00272         perror("recv");
00273     get_handlers[cmd](response);
00274 }
00275
00276 void handle_user() {
00277     char cmd;
00278     print_menu();
00279     while (scanf("%c", &cmd) == 1) {
00280         getchar(); // ignores the leading \n
00281         if (cmd == 'e')
00282             break;
00283
00284         cmd -= '0'; // converts to number
00285         // Checks if is a valid command:
00286         if (cmd >= 0 && cmd < sizeof(handlers) / sizeof(void *)) {
00287             Payload payload = handlers[cmd]();
00288             if (send(SOCKFD, &payload, sizeof(Payload), 0) == -1)
00289                 perror("send");
00290             if (payload.op == GET)
00291                 handle_get(cmd);
00292         } else {
00293             printf("\nInvalid command\n");
00294             sleep(1);
00295         }
00296         print_menu();
00297     }
00298
00299     send_exit();
00300 }
00301
00302 void sigint_handler(int sig_num) { // Signal Handler for SIGINT
00303     system("clear");
00304     printf("client: exiting...\n");
00305     send_exit();
00306     close(SOCKFD);
00307     sleep(1);
00308     exit(0);
00309 }
00310
00311 int main(int argc, char *argv[]) {
00312     struct addrinfo hints, *servinfo, *p;
00313     int rv;
00314     char s[INET6_ADDRSTRLEN];
00315
00316     if (argc != 2) {
00317         fprintf(stderr, "usage: client hostname\n");
00318         exit(1);
00319     }
00320
00321     memset(&hints, 0, sizeof hints);
00322     hints.ai_family = AF_UNSPEC;
00323     hints.ai_socktype = SOCK_STREAM;

```

```

00324
00325     if ((rv = getaddrinfo(argv[1], PORT, &hints, &servinfo)) != 0) {
00326         fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(rv));
00327         return 1;
00328     }
00329
00330     // Loop through all the results and connect to the first we can:
00331     for (p = servinfo; p != NULL; p = p->ai_next) {
00332         SOCKFD = socket(p->ai_family, p->ai_socktype, p->ai_protocol);
00333         if (SOCKFD == -1) {
00334             perror("client: socket");
00335             continue;
00336         }
00337
00338         if (connect(SOCKFD, p->ai_addr, p->ai_addrlen) == -1) {
00339             perror("client: connect");
00340             close(SOCKFD);
00341             continue;
00342         }
00343
00344         break;
00345     }
00346
00347     if (p == NULL) {
00348         fprintf(stderr, "client: failed to connect\n");
00349         return 2;
00350     }
00351
00352     inet_ntop(p->ai_family, get_in_addr((struct sockaddr *)p->ai_addr), s,
00353             sizeof s);
00354     printf("client: connecting to %s\n", s);
00355     sleep(1);
00356
00357     freeaddrinfo(servinfo); // all done with this structure
00358
00359     // Make sure the socket will be cleaned and the user will send an EXIT:
00360     signal(SIGINT, sigint_handler);
00361
00362     handle_user();
00363     close(SOCKFD);
00364
00365     return 0;
00366 }

```

4.11 src/tcp/server.c File Reference

```

#include "../data/catalog.h"
#include "../utils/net_utils.h"
#include <arpa/inet.h>
#include <errno.h>
#include <netdb.h>
#include <netinet/in.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <unistd.h>

```

Macros

- #define PORT "3490"
- #define BACKLOG 10

Functions

- void [sigchld_handler](#) (int s)
- void [handle_client](#) (int socket)
- void [backup](#) ()
- void [sigint_handler](#) (int sig_num)
- void [load_backup](#) ()
- int [main](#) (int argc, char *argv[])

Interface com o banco de dados.

=====

Funções responsáveis por interagir com as informações, alheia de qualquer protocolo de rede.

- void [post_movie](#) ([Movie](#) movie, int socket)
Summary.
- void [put_movie](#) ([Movie](#) movie, int socket)
Função responsável por atualizar as informações de um filme.
- void [get_movie](#) ([Movie](#) movie, int socket)
Função responsável por recuperar informações.
- void [del_movie](#) ([Movie](#) movie, int socket)
Summary.

Variables

- int [SOCKFD](#)
- [Catalog](#) [CATALOG](#)
- void(* [handlers](#) [])([Movie](#), int) = {[post_movie](#), [get_movie](#), [put_movie](#), [del_movie](#)}

4.11.1 Macro Definition Documentation

4.11.1.1 BACKLOG

```
#define BACKLOG 10
```

Definition at line 21 of file [server.c](#).

4.11.1.2 PORT

```
#define PORT "3490"
```

Definition at line 19 of file [server.c](#).

4.11.2 Function Documentation

4.11.2.1 backup()

```
void backup ( )
```

Definition at line 103 of file [server.c](#).

4.11.2.2 del_movie()

```
void del_movie (
    Movie movie,
    int socket )
```

Summary.

Description

Parameters

in	<i>movie</i>	Description
in	<i>socket</i>	Description

Definition at line 79 of file [server.c](#).

4.11.2.3 get_movie()

```
void get_movie (
    Movie movie,
    int socket )
```

Função responsável por recuperar informações.

Retorna todo o catálogo e é responsabilidade do client filtrar.

Returns

Struct Resposne com todo o catálogo.

Definition at line 65 of file [server.c](#).

4.11.2.4 handle_client()

```
void handle_client (
    int socket )
```

Definition at line 86 of file [server.c](#).

4.11.2.5 load_backup()

```
void load_backup ( )
```

Definition at line 127 of file [server.c](#).

4.11.2.6 main()

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

Definition at line 138 of file [server.c](#).

4.11.2.7 post_movie()

```
void post_movie (
    Movie movie,
    int socket )
```

Summary.

Description

Parameters

in	<i>movie</i>	Description
in	<i>socket</i>	Description

Definition at line 49 of file [server.c](#).

4.11.2.8 put_movie()

```
void put_movie (
    Movie movie,
    int socket )
```

Função responsável por atualizar as informações de um filme.

As informações que vierem preenchidas serão aquelas a serem atualizadas. O filme será determinado pelo ID, único campo obrigatório.

Parameters

<code>in</code>	<code>movie</code>	Struct com as informações a serem atualizadas preenchidas e o resto vazio.
-----------------	--------------------	--

Definition at line 58 of file [server.c](#).

4.11.2.9 sigchld_handler()

```
void sigchld_handler (
    int s )
```

Definition at line 26 of file [server.c](#).

4.11.2.10 sigint_handler()

```
void sigint_handler (
    int sig_num )
```

Definition at line 118 of file [server.c](#).

4.11.3 Variable Documentation

4.11.3.1 CATALOG

[Catalog](#) CATALOG

Definition at line 24 of file [server.c](#).

4.11.3.2 handlers

```
void(* handlers[])(Movie, int) (
    Movie ,
    int ) = {post_movie, get_movie, put_movie, del_movie}
```

=====

Definition at line 84 of file [server.c](#).

4.11.3.3 SOCKFD

int SOCKFD

Definition at line 23 of file [server.c](#).

4.12 server.c

[Go to the documentation of this file.](#)

```

00001 /*
00002 ** server.c -- a stream socket server demo
00003 */
00004
00005 #include "../data/catalog.h"
00006 #include "../utils/net_utils.h"
00007 #include <arpa/inet.h>
00008 #include <errno.h>
00009 #include <netdb.h>
00010 #include <netinet/in.h>
00011 #include <signal.h>
00012 #include <stdio.h>
00013 #include <stdlib.h>
00014 #include <string.h>
00015 #include <sys/socket.h>
00016 #include <sys/wait.h>
00017 #include <unistd.h>
00018
00019 #define PORT "3490" // the port users will be connecting to
00020
00021 #define BACKLOG 10 // how many pending connections queue will hold
00022
00023 int SOCKFD; // global to be closed on exit
00024 Catalog CATALOG; // global to all handlers task
00025
00026 void sigchld_handler(int s) {
00027     (void)s; // quiet unused variable warning
00028
00029     // waitpid() might overwrite errno, so we save and restore it:
00030     int saved_errno = errno;
00031     while (waitpid(-1, NULL, WNOHANG) > 0);
00032     errno = saved_errno;
00033 }
00034
00049 void post_movie(Movie movie, int socket) { add_movie(&CATALOG, movie); }
00050
00058 void put_movie(Movie movie, int socket) { update_movie(&CATALOG, movie); }
00059
00065 void get_movie(Movie movie, int socket) {
00066     Response response;
00067     memset(&response, 0, sizeof(Payload));
00068     response.data.catalog = CATALOG;
00069     if (send(socket, &response, sizeof(Response), 0) == -1)
00070         perror("send");
00071 }
00072
00079 void del_movie(Movie movie, int socket) { delete_movie(&CATALOG, movie); }
00084 void (*handlers[])(Movie, int) = {post_movie, get_movie, put_movie, del_movie};
00085
00086 void handle_client(int socket) {
00087     Payload payload;
00088     while (1) {
00089         if (recv(socket, &payload, sizeof(Payload), 0) == -1)
00090             perror("recv");
00091
00092         if (payload.op == EXIT)
00093             break;
00094         if (payload.op < 0 || payload.op > EXIT) {
00095             printf("\nInvalid operation\n");
00096             continue;
00097         }
00098         #pragma omp critical(Catalog)
00099         handlers[payload.op](payload.movie, socket); // execute the action
00100     }
00101 }
00102
00103 void backup() {
00104     FILE *f = fopen("catalog_database.data", "wb");
00105     if (f == NULL) {

```



```

00106         printf("\nError opening database file\n");
00107         return;
00108     }
00109
00110     char catalog_str[sizeof(Catalog)];
00111     // Coverts the Catalog to a byte stream:
00112     #pragma omp critical(Catalog)
00113     memcpy(catalog_str, &CATALOG, sizeof(Catalog));
00114     fwrite(catalog_str, sizeof(Catalog), 1, f);
00115     fclose(f);
00116 }
00117
00118 void sigint_handler(int sig_num) { // Signal Handler for SIGINT
00119     close(SOCKFD);
00120     system("clear");
00121     printf("server: exiting...\n");
00122     backup();
00123     sleep(1);
00124     exit(0);
00125 }
00126
00127 void load_backup() {
00128     FILE *f = fopen("catalog_database.data", "rb");
00129     if (f == NULL) {
00130         printf("\nError opening database file\n");
00131         return;
00132     }
00133
00134     fread(&CATALOG, sizeof(Catalog), 1, f);
00135     fclose(f);
00136 }
00137
00138 int main(int argc, char *argv[]) {
00139     system("clear");
00140     CATALOG.size = 0;
00141     int new_fd, code; // listen on sock_fd, new connection on new_fd
00142     struct addrinfo hints, *servinfo, *p;
00143     struct sockaddr_storage their_addr; // connector's address information
00144     socklen_t sin_size;
00145     struct sigaction sa;
00146     int yes = 1;
00147     char s[INET6_ADDRSTRLEN];
00148     int rv;
00149
00150     if (argc == 2 && strcmp(argv[1], "load") == 0)
00151         load_backup();
00152
00153     memset(&hints, 0, sizeof hints);
00154     hints.ai_family = AF_UNSPEC; // either ipv4 or ipv6
00155     hints.ai_socktype = SOCK_STREAM;
00156     hints.ai_flags = AI_PASSIVE; // use my IP
00157
00158     if ((rv = getaddrinfo(NULL, PORT, &hints, &servinfo)) != 0) {
00159         fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(rv));
00160         return 1;
00161     }
00162
00163     // Loop through all the results and bind to the first we can:
00164     for (p = servinfo; p != NULL; p = p->ai_next) {
00165         SOCKFD = socket(p->ai_family, p->ai_socktype, p->ai_protocol);
00166         if (SOCKFD == -1) {
00167             perror("server: socket");
00168             continue;
00169         }
00170
00171         // Check if the socket is ALL clear to be used.
00172         code = setsockopt(SOCKFD, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(int));
00173         if (code == -1) {
00174             perror("setsockopt");
00175             exit(1);
00176         }
00177
00178         if (bind(SOCKFD, p->ai_addr, p->ai_addrlen) == -1) {
00179             close(SOCKFD);
00180             perror("server: bind");
00181             continue;
00182         }
00183
00184         break;
00185     }
00186
00187     freeaddrinfo(servinfo); // all done with this structure
00188
00189     if (p == NULL) {
00190         fprintf(stderr, "server: failed to bind\n");
00191         exit(1);
00192     }

```

```

00193
00194     if (listen(SOCKFD, BACKLOG) == -1) {
00195         perror("listen");
00196         exit(1);
00197     }
00198
00199     sa.sa_handler = sigchld_handler; // reap all dead processes
00200     sigemptyset(&sa.sa_mask);
00201     sa.sa_flags = SA_RESTART;
00202     if (sigaction(SIGCHLD, &sa, NULL) == -1) {
00203         perror("sigaction");
00204         exit(1);
00205     }
00206
00207     // Make sure the socket will be cleaned:
00208     signal(SIGINT, sigint_handler);
00209
00210     printf("server: waiting for connections...\n");
00211
00212     #pragma omp parallel
00213     #pragma omp single nowait
00214     while (1) { // main accept() loop
00215         sin_size = sizeof(their_addr);
00216         new_fd = accept(SOCKFD, (struct sockaddr *)&their_addr, &sin_size);
00217         if (new_fd == -1) {
00218             perror("accept");
00219             continue;
00220         }
00221
00222         inet_ntop(their_addr.ss_family,
00223                 get_in_addr((struct sockaddr *)&their_addr), s, sizeof s);
00224         printf("server: got connection from %s\n", s);
00225
00226         #pragma omp task firstprivate(new_fd) // this is the child task
00227         {
00228             handle_client(new_fd);
00229             close(new_fd);
00230         }
00231         #pragma omp critical(Backup)
00232         backup();
00233     }
00234 }

```

4.13 src/utls/net_utils.c File Reference

```
#include "net_utils.h"
```

Functions

- void * [get_in_addr](#) (struct sockaddr *sa)
Retorna o endereço do socket, IPv4 ou IPv6.

4.13.1 Function Documentation

4.13.1.1 get_in_addr()

```
void * get_in_addr (
    struct sockaddr * sa )
```

Retorna o endereço do socket, IPv4 ou IPv6.

Parameters

in, out	sa	FIXME Description
---------	----	-------------------

Returns

FIXME ?

Definition at line 10 of file [net_utils.c](#).

4.14 net_utils.c

[Go to the documentation of this file.](#)

```
00001 #include "net_utils.h"
00002
00003 // Get sockaddr, IPv4 or IPv6:
00004
00010 void *get_in_addr(struct sockaddr *sa) {
00011     if (sa->sa_family == AF_INET)
00012         return &((struct sockaddr_in *)sa)->sin_addr;
00013     return &((struct sockaddr_in6 *)sa)->sin6_addr;
00014 }
```

4.15 src/utils/net_utils.h File Reference

```
#include "../data/catalog.h"
#include "../data/movie.h"
#include <netinet/in.h>
#include <sys/socket.h>
```

Data Structures

- struct [Payload](#)
- union [CatalogMovie](#)
- struct [Response](#)

Macros

- #define [ALL](#) -1

Enumerations

- enum [Operation](#) {
[POST](#) , [GET](#) , [PUT](#) , [DEL](#) ,
[EXIT](#) }

Functions

- void * [get_in_addr](#) (struct sockaddr *sa)
Retorna o endereço do socket, IPv4 ou IPv6.

4.15.1 Macro Definition Documentation

4.15.1.1 ALL

```
#define ALL -1
```

Definition at line 9 of file [net_utils.h](#).

4.15.2 Enumeration Type Documentation

4.15.2.1 Operation

```
enum Operation
```

Enumerator

POST	
GET	
PUT	
DEL	
EXIT	

Definition at line 11 of file [net_utils.h](#).

4.15.3 Function Documentation

4.15.3.1 get_in_addr()

```
void * get_in_addr (  
    struct sockaddr * sa )
```

Retorna o endereço do socket, IPv4 ou IPv6.

Parameters

in, out	sa	FIXME Description
---------	----	-------------------

Returns

FIXME ?

Definition at line 10 of file [net_utils.c](#).

4.16 net_utils.h

[Go to the documentation of this file.](#)

```

00001 #ifndef MC833_PROJETO_NET_UTILS_H
00002 #define MC833_PROJETO_NET_UTILS_H
00003
00004 #include "../data/catalog.h"
00005 #include "../data/movie.h"
00006 #include <netinet/in.h>
00007 #include <sys/socket.h>
00008
00009 #define ALL -1
00010
00011 typedef enum { POST, GET, PUT, DEL, EXIT } Operation;
00012
00017 typedef struct {
00018     Operation op;
00019     Movie movie;
00020 } Payload;
00021
00022 typedef union {
00023     Catalog catalog;
00024     Movie movie;
00025 } CatalogMovie;
00026
00031 typedef struct {
00032     CatalogMovie data;
00033 } Response;
00034
00035 void *get_in_addr(struct sockaddr *sa);
00036
00037 #endif // MC833_PROJETO_NET_UTILS_H

```

4.17 src/utils/utils.h File Reference

Macros

- `#define MAX_STR_LEN 200`

4.17.1 Macro Definition Documentation

4.17.1.1 MAX_STR_LEN

```
#define MAX_STR_LEN 200
```

Definition at line 4 of file [utils.h](#).

4.18 utils.h

[Go to the documentation of this file.](#)

```
00001 #ifndef MC833_PROJETO_UTILS_H
00002 #define MC833_PROJETO_UTILS_H
00003
00004 #define MAX_STR_LEN 200
00005
00006 #endif // MC833_PROJETO_UTILS_H
```

Index

- add_genre
 - movie.c, [15](#)
 - movie.h, [18](#)
- add_movie
 - catalog.c, [11](#)
 - catalog.h, [14](#)
- ALL
 - net_utils.h, [38](#)
- BACKLOG
 - server.c, [30](#)
- backup
 - server.c, [30](#)
- CATALOG
 - server.c, [33](#)
- Catalog, [5](#)
 - movie_list, [5](#)
 - size, [5](#)
- catalog
 - CatalogMovie, [6](#)
- catalog.c
 - add_movie, [11](#)
 - delete_movie, [12](#)
 - update_movie, [12](#)
- catalog.h
 - add_movie, [14](#)
 - delete_movie, [14](#)
 - MAX_MOVIES, [13](#)
 - update_movie, [14](#)
- CatalogMovie, [6](#)
 - catalog, [6](#)
 - movie, [6](#)
- client.c
 - get_handlers, [25](#)
 - get_movies, [21](#)
 - handle_get, [21](#)
 - handle_user, [21](#)
 - handlers, [25](#)
 - list_all_info, [21](#)
 - list_info_by_genre, [22](#)
 - list_info_by_id, [22](#)
 - list_titles, [22](#)
 - main, [23](#)
 - PORT, [21](#)
 - post_movie, [23](#)
 - print_all_info, [23](#)
 - print_menu, [24](#)
 - put_genre, [24](#)
 - remove_movie, [24](#)
 - send_exit, [24](#)
 - sigint_handler, [24](#)
 - SOCKFD, [25](#)
 - wait_for_enter, [25](#)
- contains_genre
 - movie.c, [16](#)
 - movie.h, [18](#)
- create_movie
 - movie.c, [16](#)
 - movie.h, [19](#)
- data
 - Response, [10](#)
- DEL
 - net_utils.h, [38](#)
- del_movie
 - server.c, [31](#)
- delete_movie
 - catalog.c, [12](#)
 - catalog.h, [14](#)
- director_name
 - Movie, [7](#)
- EXIT
 - net_utils.h, [38](#)
- genre_list
 - Movie, [7](#)
- GET
 - net_utils.h, [38](#)
- get_handlers
 - client.c, [25](#)
- get_in_addr
 - net_utils.c, [36](#)
 - net_utils.h, [38](#)
- get_movie
 - server.c, [31](#)
- get_movies
 - client.c, [21](#)
- handle_client
 - server.c, [31](#)
- handle_get
 - client.c, [21](#)
- handle_user
 - client.c, [21](#)
- handlers
 - client.c, [25](#)
 - server.c, [33](#)
- id

- Movie, 7
- list_all_info
 - client.c, 21
- list_info_by_genre
 - client.c, 22
- list_info_by_id
 - client.c, 22
- list_titles
 - client.c, 22
- load_backup
 - server.c, 31
- main
 - client.c, 23
 - server.c, 32
- MAX_MOVIE_GENRES
 - movie.h, 18
- MAX_MOVIES
 - catalog.h, 13
- MAX_STR_LEN
 - utils.h, 39
- Movie, 7
 - director_name, 7
 - genre_list, 7
 - id, 7
 - num_genres, 8
 - title, 8
 - year, 8
- movie
 - CatalogMovie, 6
 - Payload, 9
- movie.c
 - add_genre, 15
 - contains_genre, 16
 - create_movie, 16
- movie.h
 - add_genre, 18
 - contains_genre, 18
 - create_movie, 19
 - MAX_MOVIE_GENRES, 18
- movie_list
 - Catalog, 5
- net_utils.c
 - get_in_addr, 36
- net_utils.h
 - ALL, 38
 - DEL, 38
 - EXIT, 38
 - GET, 38
 - get_in_addr, 38
 - Operation, 38
 - POST, 38
 - PUT, 38
- num_genres
 - Movie, 8
- op
 - Payload, 9
- Operation
 - net_utils.h, 38
- Payload, 8
 - movie, 9
 - op, 9
- PORT
 - client.c, 21
 - server.c, 30
- POST
 - net_utils.h, 38
- post_movie
 - client.c, 23
 - server.c, 32
- print_all_info
 - client.c, 23
- print_menu
 - client.c, 24
- PUT
 - net_utils.h, 38
- put_genre
 - client.c, 24
- put_movie
 - server.c, 32
- remove_movie
 - client.c, 24
- Response, 9
 - data, 10
- send_exit
 - client.c, 24
- server.c
 - BACKLOG, 30
 - backup, 30
 - CATALOG, 33
 - del_movie, 31
 - get_movie, 31
 - handle_client, 31
 - handlers, 33
 - load_backup, 31
 - main, 32
 - PORT, 30
 - post_movie, 32
 - put_movie, 32
 - sigchld_handler, 33
 - sigint_handler, 33
 - SOCKFD, 33
- sigchld_handler
 - server.c, 33
- sigint_handler
 - client.c, 24
 - server.c, 33
- size
 - Catalog, 5
- SOCKFD
 - client.c, 25
 - server.c, 33

src/data/catalog.c, [11](#), [12](#)
src/data/catalog.h, [13](#), [15](#)
src/data/movie.c, [15](#), [17](#)
src/data/movie.h, [17](#), [19](#)
src/tcp/client.c, [20](#), [26](#)
src/tcp/server.c, [29](#), [34](#)
src/utls/net_utls.c, [36](#), [37](#)
src/utls/net_utls.h, [37](#), [39](#)
src/utls/utls.h, [39](#), [40](#)

title

 Movie, [8](#)

update_movie

 catalog.c, [12](#)

 catalog.h, [14](#)

utls.h

 MAX_STR_LEN, [39](#)

wait_for_enter

 client.c, [25](#)

year

 Movie, [8](#)