MC833 - Projeto 1

Generated by Doxygen 1.9.3

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 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/utils/net_utils.h

3.3 Movie Struct Reference

`#include <movie.h>`

Data Fields

- short id
- char title [MAX_STR_LEN]
- short num_genres
- char genre_list [MAX_MOVIE_GENRES][MAX_STR_LEN]
- char director_name [MAX_STR_LEN]
- short 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

```
short id
```

Identificador único

Definition at line 13 of file movie.h.

3.3.2.4 num_genres

```
short 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

```
short 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/utils/net_utils.h

3.5 Response Struct Reference

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

Data Fields

- CatalogMovie data

3.5.1 Detailed Description

Struct utilizada como respota 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/utils/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	*catalog*	Catálogo a ser modificado.
in	*movie*	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	*catalog*	Catálogo da onde o movie será tirado.
in	*movie*	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	*catalog*	Lista de filmes contendo o filme a ser alterado.
in	*movie*	As informações a serem atualizadas estarão preenchidas. O resto deve estar vazio enquanto que o ID é obirgatório.

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 20

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 20
```

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	*catalog*	Catálogo a ser modificado.
in	*movie*	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	*catalog*	Catálogo da onde o movie será tirado.
in	*movie*	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	*catalog*	Lista de filmes contendo o filme a ser alterado.
in	*movie*	As informações a serem atualizadas estarão preenchidas. O resto deve estar vazio enquanto que o ID é obirgatório.

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 20
00007
00012 typedef struct {
00013     char size;
00014     Movie movie_list[MAX_MOVIES];
00015 } Catalog; // **SHOULD NOT EXCED 4096 BYTES!!!**
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 (short id, char *title, char *genre, char *director, short 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	*movie*	Filme a ter um gênero inserido.
out	*genre*	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	*movie*	Filme a ser testado.
out	*genre*	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 (
            short id,
            char * title,
            char * genre,
            char * director,
            short year )
```

Função para criar uma struct movie.

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

Parameters

in	*id*	
out	*title*	Título do filme a ser criado.
out	*genre*	Gêneros iniciais para o filme.
out	*director*	Nome do diretor.
in	*year*	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(short id, char *title, char *genre, char *director,
00017                     short 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 4

Functions

- Movie * create_movie (short id, char *title, char *genre, char *director, short 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 4
```

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	*movie*	Filme a ter um gênero inserido.
out	*genre*	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	*movie*	Filme a ser testado.
out	*genre*	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 (
            short id,
            char * title,
            char * genre,
            char * director,
            short year )
```

Função para criar uma struct movie.

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

Parameters

in	*id*	
out	*title*	Título do filme a ser criado.
out	*genre*	Gêneros iniciais para o filme.
out	*director*	Nome do diretor.
in	*year*	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 4
00007
00012 typedef struct {
00013     short id;
00014     char title[MAX_STR_LEN];
00015     short num_genres;
00016     char genre_list[MAX_MOVIE_GENRES][MAX_STR_LEN];
00017     char director_name[MAX_STR_LEN];
00018     short year;
00019     // imagem capa;                                  /**< Description */
00020 } Movie;
00021
00022 Movie *create_movie(short id, char *title, char *genre, char *director, short 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 ()

  Impressão do menu principal para o cliente.
- void send_exit ()

  Função para enviar o fechamento de conexão.
- void handle_get (char cmd)

  Função que aguarda o retorno de operações get.
- void handle_user ()

  Função de controle do menu.
- 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 )
```

Função que aguarda o retorno de operações get.

Uma vez enviada, a operação GET espera um retorno.

Parameters

in	*cmd*	Qual comando foi enviado.

Definition at line 280 of file client.c.

4.9.2.3 handle_user()

```
void handle_user ( )
```

Função de controle do menu.

A cada iteração do menu, lemos um caracter que indica qual o comando a ser realizado.

Definition at line 292 of file client.c.

4.9.2.4 list_all_info()

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

Summary.

Description

Parameters

in	*response*	Description

Definition at line 177 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	*response*	Resposta do servidor com o catálogo.

Definition at line 192 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 possue id igual ao fornecido.

Parameters

in	*response*	Resposta do servidor com o catálogo.

Definition at line 220 of file client.c.

4.9.2.7 list_titles()

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

Imprimir os filmes obtidos.

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

Parameters

in	*response*	Resposta com o catálogo a ser impresso.

Definition at line 149 of file client.c.

4.9.2.8 main()

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

Definition at line 327 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	*movie*	Filme cujas informações devemos imprimir.

Definition at line 164 of file client.c.

4.9.2.11 print_menu()

```
void print_menu ( )
```

Impressão do menu principal para o cliente.

Definition at line 251 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 121 of file client.c.

4.9.2.14 send_exit()

```
void send_exit ( )
```

Função para enviar o fechamento de conexão.

Envia um payload com a operação EXIT.

Definition at line 268 of file client.c.

4.9.2.15 sigint_handler()

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

Definition at line 318 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 244 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 242 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 <netdb.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("%hd", &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("%hd", &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("%hd", &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("%hd", &ret.movie.id);
00102
00103     printf("Digite o número de gêneros que deseja adicionar a esse filme: ");
00104     scanf("%hd", &ret.movie.num_genres);
00105     getchar(); // ignores the leading \n
00106
00107     for (int i = 0; i < ret.movie.num_genres; i++) {
00108         printf("Digite o %dº novo gênero desse filme: ", i + 1);
00109         fgets(ret.movie.genre_list[i], MAX_STR_LEN, stdin);
00110         ret.movie.genre_list[i][strcspn(ret.movie.genre_list[i], "\n")] = '\0';
00111     }
00112
00113     return ret;
00114 }
00115
00121 Payload remove_movie() {
00122     Payload ret;
00123     memset(&ret, 0, sizeof(Payload));
00124     ret.op = DEL;
00125
00126     system("clear");
00127     printf("Remover filme");
00128
00129     printf("\nDigite o id do filme a ser removido: ");
00130     scanf("%hd", &ret.movie.id);
00131     getchar(); // ignores the leading \n
00132
00133     return ret;
00134 }
00149 void list_titles(Response response) {
00150     system("clear");
00151     printf("Lista de filmes:\n");
00152     printf(" -> id  - título\n");
```

```
00153     for (int i = 0; i < response.data.catalog.size; i++)
00154         printf(" -> %d - %s\n", response.data.catalog.movie_list[i].id,
00155                response.data.catalog.movie_list[i].title);
00156     wait_for_enter();
00157 }
00158
00164 void print_all_info(Movie movie) {
00165     printf(" -> %d - (%d) %s by %s |", movie.id, movie.year, movie.title,
00166            movie.director_name);
00167     for (int i = 0; i < movie.num_genres; i++)
00168         printf(" %s", movie.genre_list[i]);
00169     printf("\n");
00170 }
00171
00177 void list_all_info(Response response) {
00178     system("clear");
00179     printf("Informações dos filmes:\n");
00180     printf(" -> id  - (ano) título by diretor | Gêneros\n");
00181     for (int i = 0; i < response.data.catalog.size; i++)
00182         print_all_info(response.data.catalog.movie_list[i]);
00183     wait_for_enter();
00184 }
00185
00192 void list_info_by_genre(Response response) {
00193     char genre[MAX_STR_LEN];
00194     system("clear");
00195     printf("Digite o gênero: ");
00196     fgets(genre, MAX_STR_LEN, stdin);
00197     genre[strcspn(genre, "\n")] = '\0';
00198
00199     // REVIEW Do jeito que ele colocou na descrição, parecia que ele queria
00200     // que essa filtragem fosse feita pelo servidor.
00201     system("clear");
00202     printf("Lista de filmes:\n");
00203     printf(" -> título - nome do diretor - ano\n");
00204     Movie aux;
00205     for (int i = 0; i < response.data.catalog.size; i++) {
00206         aux = response.data.catalog.movie_list[i];
00207         if (contains_genre(&aux, genre))
00208             printf(" -> %s - %s - %d\n", aux.title, aux.director_name,
00209                    aux.year);
00210     }
00211     wait_for_enter();
00212 }
00213
00220 void list_info_by_id(Response response) {
00221     int id;
00222     system("clear");
00223     printf("Digite o id: ");
00224     scanf("%d", &id);
00225     getchar(); // ignores the leading \n
00226
00227     system("clear");
00228     printf(" -> id  - (ano) título by diretor | Gêneros\n");
00229     for (int i = 0; i < response.data.catalog.size; i++)
00230         if (response.data.catalog.movie_list[i].id == id) {
00231             print_all_info(response.data.catalog.movie_list[i]);
00232             break;
00233         }
00234
00235     wait_for_enter();
00236 }
00242 Payload (*handlers[])() = {post_movie, put_genre,  get_movies,  get_movies,
00243                            get_movies, get_movies, remove_movie};
00244 void (*get_handlers[])(Response) = {NULL,          NULL,
00245                                     list_titles,   list_info_by_genre,
00246                                     list_all_info, list_info_by_id};
00247
00251 void print_menu() {
00252     system("clear");
00253     printf("0 - Cadastrar um novo filme");
00254     printf("\n1 - Acrescentar um novo gênero em um filme");
00255     printf("\n2 - Listar títulos");
00256     printf("\n3 - Listar informações por gênero");
00257     printf("\n4 - Listar todas as informações de todos os filmes");
00258     printf("\n5 - Listar todas as informações de um filme");
00259     printf("\n6 - Remover filme");
00260     printf("\ne - exit");
00261     printf("\nDigite um comando: ");
00262 }
00263
00268 void send_exit() {
00269     Payload payload;
00270     payload.op = EXIT;
00271     if (send(SOCKFD, &payload, sizeof(Payload), 0) == -1)
00272         perror("send");
00273 }
```

```
00274
00280 void handle_get(char cmd) {
00281     Response response;
00282     if (recv(SOCKFD, &response, sizeof(Response), 0) == -1)
00283         perror("recv");
00284     get_handlers[cmd](response);
00285 }
00286
00292 void handle_user() {
00293     char cmd;
00294     print_menu();
00295     while (scanf("%c", &cmd) == 1) {
00296         getchar(); // ignores the leading \n
00297         if (cmd == 'e')
00298             break;
00299
00300         cmd -= '0'; // converts to number
00301         // Checks if is a valid command:
00302         if (cmd >= 0 && cmd < sizeof(handlers) / sizeof(void *)) {
00303             Payload payload = handlers[cmd]();
00304             if (send(SOCKFD, &payload, sizeof(Payload), 0) == -1)
00305                 perror("send");
00306             if (payload.op == GET)
00307                 handle_get(cmd);
00308         } else {
00309             printf("\nInvalid command\n");
00310             sleep(1);
00311         }
00312         print_menu();
00313     }
00314
00315     send_exit();
00316 }
00317
00318 void sigint_handler(int sig_num) { // Signal Handler for SIGINT
00319     system("clear");
00320     printf("client: exiting...\n");
00321     send_exit();
00322     close(SOCKFD);
00323     sleep(1);
00324     exit(0);
00325 }
00326
00327 int main(int argc, char *argv[]) {
00328     struct addrinfo hints, *servinfo, *p;
00329     int rv;
00330     char s[INET6_ADDRSTRLEN];
00331
00332     if (argc != 2) {
00333         fprintf(stderr, "usage: client hostname\n");
00334         exit(1);
00335     }
00336
00337     memset(&hints, 0, sizeof hints);
00338     hints.ai_family = AF_UNSPEC;
00339     hints.ai_socktype = SOCK_STREAM;
00340
00341     if ((rv = getaddrinfo(argv[1], PORT, &hints, &servinfo)) != 0) {
00342         fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(rv));
00343         return 1;
00344     }
00345
00346     // Loop through all the results and connect to the first we can:
00347     for (p = servinfo; p != NULL; p = p->ai_next) {
00348         SOCKFD = socket(p->ai_family, p->ai_socktype, p->ai_protocol);
00349         if (SOCKFD == -1) {
00350             perror("client: socket");
00351             continue;
00352         }
00353
00354         if (connect(SOCKFD, p->ai_addr, p->ai_addrlen) == -1) {
00355             perror("client: connect");
00356             close(SOCKFD);
00357             continue;
00358         }
00359
00360         break;
00361     }
00362
00363     if (p == NULL) {
00364         fprintf(stderr, "client: failed to connect\n");
00365         return 2;
00366     }
00367
00368     inet_ntop(p->ai_family, get_in_addr((struct sockaddr *)p->ai_addr), s,
00369               sizeof s);
00370     printf("client: connecting to %s\n", s);
```

```
00371     sleep(1);
00372
00373     freeaddrinfo(servinfo); // all done with this structure
00374
00375     // Make sure the socket will be cleaned and the user will send an EXIT:
00376     signal(SIGINT, sigint_handler);
00377
00378     handle_user();
00379     close(SOCKFD);
00380
00381     return 0;
00382 }
```

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)

  Função que espera por uma conexão.
- void backup ()

  Função para salvar o estado do catálogo.
- void load_backup ()

  Função para recuperar o backup.
- void sigint_handler (int sig_num)

  Função para finalizar o servidor.
- 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 ( )
```

Função para salvar o estado do catálogo.

Escrevemos o estado atual a um arquivo para arquiva-lo.

Definition at line 112 of file server.c.

4.11.2.2 del_movie()

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

Summary.

Description

Parameters

in	*movie*	Description
in	*socket*	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 )
```

Função que espera por uma conexão.

Em um laço infinito, esperamos uma nova conexão chegar.

Parameters

in	*socket*	Socket a escutar.

Definition at line 91 of file server.c.

4.11.2.5 load_backup()

```
void load_backup ( )
```

Função para recuperar o backup.

Lemos o arquivo que foi salvo anteriormente para recuperar o catálogo.

Definition at line 132 of file server.c.

4.11.2.6 main()

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

Definition at line 157 of file server.c.

4.11.2.7 post_movie()

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

Summary.

Description

Parameters

in	*movie*	Description
in	*socket*	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

in	*movie*	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 )
```

Função para finalizar o servidor.

Fechamos o socket, fazemos o backup do catálogo e terminamos o programa.

Definition at line 148 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
00091 void handle_client(int socket) {
00092     Payload payload;
00093     while (1) {
00094         if (recv(socket, &payload, sizeof(Payload), 0) == -1)
00095             perror("recv");
00096
00097         if (payload.op == EXIT)
00098             break;
00099         if (payload.op < 0 || payload.op > EXIT) {
00100             printf("\nInvalid operation\n");
00101             continue;
00102         }
00103 #pragma omp critical(Catalog)
00104         handlers[payload.op](payload.movie, socket); // execute the action
00105     }
00106 }
00107
00112 void backup() {
00113     FILE *f = fopen("catalog_database.data", "wb");
00114     if (f == NULL) {
```

```
00115         printf("\nError opening database file\n");
00116         return;
00117     }
00118
00119     char catalog_str[sizeof(Catalog)];
00120     // Coverts the Catalog to a byte stream:
00121 #pragma omp critical(Catalog)
00122     memcpy(catalog_str, &CATALOG, sizeof(Catalog));
00123     fwrite(catalog_str, sizeof(Catalog), 1, f);
00124     fclose(f);
00125 }
00126
00132 void load_backup() {
00133     FILE *f = fopen("catalog_database.data", "rb");
00134     if (f == NULL) {
00135         printf("\nError opening database file\n");
00136         return;
00137     }
00138
00139     fread(&CATALOG, sizeof(Catalog), 1, f);
00140     fclose(f);
00141 }
00142
00148 void sigint_handler(int sig_num) { // Signal Handler for SIGINT
00149     close(SOCKFD);
00150     system("clear");
00151     printf("server: exiting...\n");
00152     backup();
00153     sleep(1);
00154     exit(0);
00155 }
00156
00157 int main(int argc, char *argv[]) {
00158     system("clear");
00159     CATALOG.size = 0;
00160     int new_fd, code; // listen on sock_fd, new connection on new_fd
00161     struct addrinfo hints, *servinfo, *p;
00162     struct sockaddr_storage their_addr; // connector's address information
00163     socklen_t sin_size;
00164     struct sigaction sa;
00165     int yes = 1;
00166     char s[INET6_ADDRSTRLEN];
00167     int rv;
00168
00169     if (argc == 2 && strcmp(argv[1], "load") == 0)
00170         load_backup();
00171
00172     memset(&hints, 0, sizeof hints);
00173     hints.ai_family = AF_UNSPEC; // either ipv4 or ipv6
00174     hints.ai_socktype = SOCK_STREAM;
00175     hints.ai_flags = AI_PASSIVE; // use my IP
00176
00177     if ((rv = getaddrinfo(NULL, PORT, &hints, &servinfo)) != 0) {
00178         fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(rv));
00179         return 1;
00180     }
00181
00182     // Loop through all the results and bind to the first we can:
00183     for (p = servinfo; p != NULL; p = p->ai_next) {
00184         SOCKFD = socket(p->ai_family, p->ai_socktype, p->ai_protocol);
00185         if (SOCKFD == -1) {
00186             perror("server: socket");
00187             continue;
00188         }
00189
00190         // Check if the socket is ALL clear to be used.
00191         code = setsockopt(SOCKFD, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(int));
00192         if (code == -1) {
00193             perror("setsockopt");
00194             exit(1);
00195         }
00196
00197         if (bind(SOCKFD, p->ai_addr, p->ai_addrlen) == -1) {
00198             close(SOCKFD);
00199             perror("server: bind");
00200             continue;
00201         }
00202
00203         break;
00204     }
00205
00206     freeaddrinfo(servinfo); // all done with this structure
00207
00208     if (p == NULL) {
00209         fprintf(stderr, "server: failed to bind\n");
00210         exit(1);
00211     }
```

```
00212
00213     if (listen(SOCKFD, BACKLOG) == -1) {
00214         perror("listen");
00215         exit(1);
00216     }
00217
00218     sa.sa_handler = sigchld_handler; // reap all dead processes
00219     sigemptyset(&sa.sa_mask);
00220     sa.sa_flags = SA_RESTART;
00221     if (sigaction(SIGCHLD, &sa, NULL) == -1) {
00222         perror("sigaction");
00223         exit(1);
00224     }
00225
00226     // Make sure the socket will be cleaned:
00227     signal(SIGINT, sigint_handler);
00228
00229     printf("server: waiting for connections...\n");
00230
00231 #pragma omp parallel
00232 #pragma omp single nowait
00233     while (1) { // main accept() loop
00234         sin_size = sizeof(their_addr);
00235         new_fd = accept(SOCKFD, (struct sockaddr *)&their_addr, &sin_size);
00236         if (new_fd == -1) {
00237             perror("accept");
00238             continue;
00239         }
00240
00241         inet_ntop(their_addr.ss_family,
00242                   get_in_addr((struct sockaddr *)&their_addr), s, sizeof s);
00243         printf("server: got connection from %s\n", s);
00244
00245 #pragma omp task firstprivate(new_fd) // this is the child task
00246         {
00247             handle_client(new_fd);
00248             close(new_fd);
00249 #pragma omp critical(Backup)
00250             backup();
00251         }
00252     }
00253 }
```

4.13 src/utils/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 30

4.17.1 Macro Definition Documentation

4.17.1.1 MAX_STR_LEN

```
#define MAX_STR_LEN 30
```

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 30
00005
00006 #endif // MC833_PROJETO_UTILS_H
```

Index