Trabalho Prático Introdutório

1

Gerado por Doxygen 1.9.8

Capítulo 1

orgarquivos-trabalhos

Repositório para armazenar trabalhos em grupo da disciplina de Organização de Arquivos

Capítulo 2

Índice das Estruturas de Dados

2.1 Estruturas de Dados

Aqui estão as estruturas de dados, uniões e suas respectivas descrições:

Capítulo 3

Índice dos Arquivos

3.1 Lista de Arquivos

Esta é a lista de todos os arquivos documentados e suas respectivas descrições:

Capítulo 4

Estruturas

4.1 Referência da Estrutura _jogador

Estrutura que contém os dados de um jogador, pode representar um registro no arquivo de dados.

`#include <data-file.h>`

Campos de Dados

- int32_t id
- int32_t idade
- char * nome
- char * nac
- char * clube

4.1.1 Descrição detalhada

Estrutura que contém os dados de um jogador, pode representar um registro no arquivo de dados.

Observação

As strings contidas dentro da estrutura terminam com '\0' para compatibilidade do processamento pelas funções da biblioteca string.h

4.1.2 Campos

4.1.2.1 id

`int32_t _jogador::id`

id do jogador (32 bits ou 4 bytes)

4.1.2.2 idade

```
int32_t _jogador::idade
```

idade do jogador (32 bits ou 4 bytes)

4.1.2.3 nome

```
char* _jogador::nome
```

nome do jogador (string terminada em char nulo de tamanho variável)

4.1.2.4 nac

```
char* _jogador::nac
```

nacionalidade do jogador (string terminada em char nulo de tamanho variável)

4.1.2.5 clube

```
char* _jogador::clube
```

clube do jogador (string terminada em char nulo de tamanho variável)

A documentação para essa estrutura foi gerada a partir do seguinte arquivo:

- data-file.h

Capítulo 5

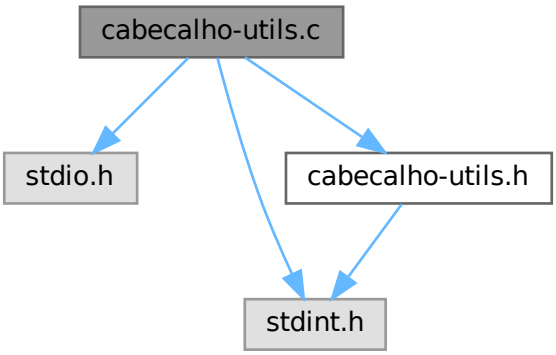
Arquivos

5.1 Referência do Arquivo cabecalho-utils.c

Source file para as funcionalidades do cabeçalho.

```
#include <stdio.h>
#include <stdint.h>
#include "cabecalho-utils.h"
```

Gráfico de dependência de inclusões para cabecalho-utils.c:



Funções

- int initialize_cabecalho (const unsigned char status, const int64_t topo, const int64_t prox_byte_offset, const int32_t nro_regarq, const int32_t nro_regrem, FILE ∗fp)

  Inicializa um novo cabecalho em um arquivo de dados binario.
- int check_status (FILE ∗fp)

  Funcao que verifica o status de um arquivo de dados binario.

5.1.1 Descrição detalhada

Source file para as funcionalidades do cabeçalho.

Alguma descrição específica sobre o arquivo...

Autores

Nicholas Eiti Dan; N°USP: 14600749
Neri??; N°USP: XXXXXXXX

Versão

1.0

5.1.2 Funções

5.1.2.1 initialize_cabecalho()

```
int initialize_cabecalho (
            const unsigned char status,
            const int64_t topo,
            const int64_t prox_byte_offset,
            const int32_t nro_regarq,
            const int32_t nro_regrem,
            FILE * fp )
```

Inicializa um novo cabecalho em um arquivo de dados binario.

Como a função é composta unicamente pela função fwrite, é possível ler o stderr para a identificação de erro.

Parâmetros

in	*status*	Valor a ser inserido no campo status.
in	*topo*	Valor a ser inserido no campo topo.
in	*prox_byte_offset*	Valor a ser inserido no campo proxByteOffset.
in	*nro_regarq*	Valor a ser inserido no campo nroRegArq.
in	*nro_regrem*	Valor a ser inserido no campo nroRegRem.
	fp	Ponteiro do arquivo binário de dados.

Valores Retornados

0	Operação realizada com sucesso.
-1	Houve uma falha durante a execução da funcionalidade.

Esse é o diagrama das funções que utilizam essa função:

5.1.2.2 check_status()

```
int check_status (
            FILE * fp )
```

Funcao que verifica o status de um arquivo de dados binario.

A função não reposiciona o ponteiro para o offset do status, então o ponteiro deve estar na posição correta (STATUS_OFFSET).

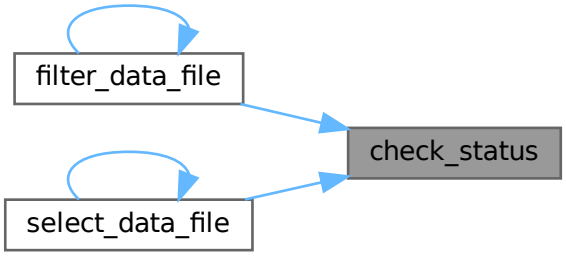
Parâmetros

fp	Ponteiro do arquivo binário de dados.

Valores Retornados

1	Status do arquivo é válido (status = '1')
0	Status do arquivo não é válido

Esse é o diagrama das funções que utilizam essa função:

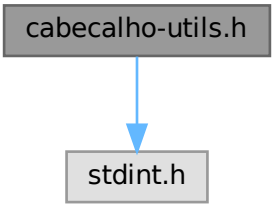


5.2 Referência do Arquivo cabecalho-utils.h

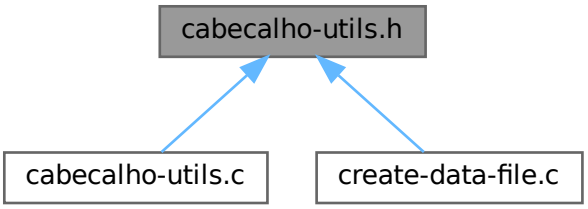
Header file para funcionalidades do cabeçalho.

```
#include <stdint.h>
```

Gráfico de dependência de inclusões para cabecalho-utils.h:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Definições e Macros

- #define **STATUS_OFFSET** 0
- #define **TOPO_OFFSET** 1
- #define **PROXBYTE_OFFSET** 9
- #define **NRO_REGARQ_OFFSET** 17
- #define **NRO_REGREM_OFFSET** 21
- #define **HEADER_END_OFFSET** 25

Funções

- int initialize_cabecalho (const unsigned char status, const int64_t topo, const int64_t prox_byte_offset, const int32_t nro_regarq, const int32_t nro_regrem, FILE ∗fp)

  Inicializa um novo cabecalho em um arquivo de dados binario.
- int check_status (FILE ∗fp)

  Funcao que verifica o status de um arquivo de dados binario.

5.2.1 Descrição detalhada

Header file para funcionalidades do cabeçalho.

Este arquivo contem as declaracoes de funcoes dedicadas a manipulacao do cabecalho do arquivo de dados como especificado no trabalho introdutório da disciplina de Organizacao de Arquivos.

ESTRUTURA DO CABEÇALHO:

| status (1 byte) | topo (8 bytes) | proxByteOffset (8 bytes) | nroRegArq (4 bytes) | nroRegRem (4 bytes) |

Autores

Nicholas Eiti Dan; N°USP: 14600749

Neri??; N°USP: XXXXXXXX

Versão

1.0

5.2.2 Funções

5.2.2.1 initialize_cabecalho()

```
int initialize_cabecalho (
            const unsigned char status,
            const int64_t topo,
            const int64_t prox_byte_offset,
            const int32_t nro_regarq,
            const int32_t nro_regrem,
            FILE * fp )
```

Inicializa um novo cabecalho em um arquivo de dados binario.

Como a função é composta unicamente pela função fwrite, é possível ler o stderr para a identificação de erro.

Parâmetros

in	*status*	Valor a ser inserido no campo status.
in	*topo*	Valor a ser inserido no campo topo.
in	*prox_byte_offset*	Valor a ser inserido no campo proxByteOffset.
in	*nro_regarq*	Valor a ser inserido no campo nroRegArq.
in	*nro_regrem*	Valor a ser inserido no campo nroRegRem.
	fp	Ponteiro do arquivo binário de dados.

Valores Retornados

0	Operação realizada com sucesso.
-1	Houve uma falha durante a execução da funcionalidade.

Esse é o diagrama das funções que utilizam essa função:

5.2.2.2 check_status()

```
int check_status (
            FILE * fp )
```

Funcao que verifica o status de um arquivo de dados binario.

A função não reposiciona o ponteiro para o offset do status, então o ponteiro deve estar na posição correta (STATUS_OFFSET).

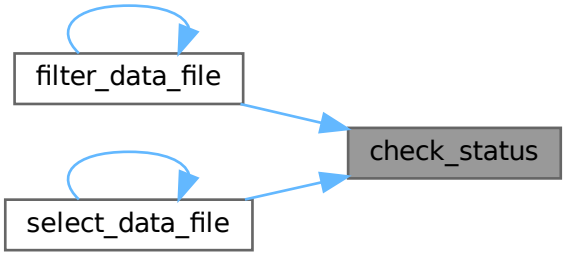
Parâmetros

fp	Ponteiro do arquivo binário de dados.

Valores Retornados

1	Status do arquivo é válido (status = '1')
0	Status do arquivo não é válido

Esse é o diagrama das funções que utilizam essa função:



5.3 cabecalho-utils.h

Ir para a documentação desse arquivo.

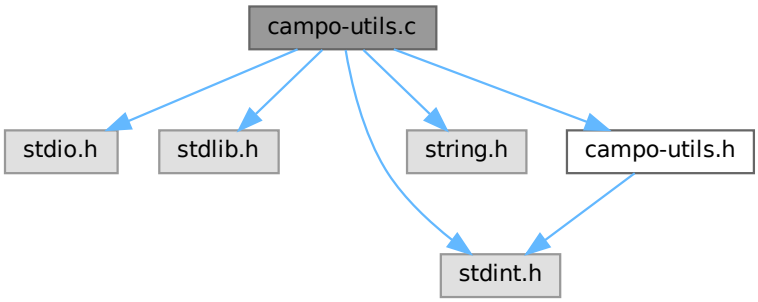
```
00001
00019 #ifndef CABCL_UTILS_H
00020 #define CABCL_UTILS_H
00021
00022 #include <stdint.h>
00023
00024 // Lista de offsets dos campos do cabecalho (campos estaticos)
00025
00026 #define STATUS_OFFSET 0
00027 #define TOPO_OFFSET 1
00028 #define PROXBYTE_OFFSET 9
00029 #define NRO_REGARQ_OFFSET 17
00030 #define NRO_REGREM_OFFSET 21
00031 #define HEADER_END_OFFSET 25
00032
00049 int initialize_cabecalho(const unsigned char status, const int64_t topo, const int64_t
      prox_byte_offset,
00050 const int32_t nro_regarq, const int32_t nro_regrem, FILE *fp);
00051
00063 int check_status(FILE *fp);
00064
00065 #endif
```

5.4 Referência do Arquivo campo-utils.c

Source file para as funcionalidades dos campos de um registro.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include "campo-utils.h"
```

Gráfico de dependência de inclusões para campo-utils.c:



Funções

- unsigned char **get_campoc** (FILE *fp)
- int32_t **get_campo32** (FILE *fp)
- int64_t **get_campo64** (FILE *fp)
- char * **get_campo_str** (FILE *fp)
- int set_campoc (const unsigned char c, FILE *fp)
- int set_campo32 (const int32_t val, FILE *fp)
- int set_campo64 (const int64_t val, FILE *fp)
- int set_campo_str (const char *str, int32_t *campo_len, FILE *fp)

5.4.1 Descrição detalhada

Source file para as funcionalidades dos campos de um registro.

Alguma descrição específica sobre o arquivo...

Autores

Nicholas Eiti Dan; N°USP: 14600749

Neri??; N°USP: XXXXXXXX

Versão

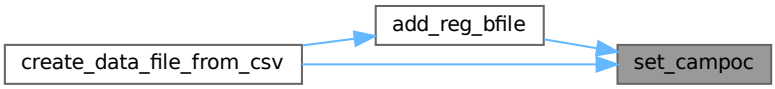
1.0

5.4.2 Funções

5.4.2.1 set_campoc()

```
int set_campoc (
            const unsigned char c,
            FILE * fp )
```

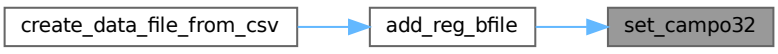
Atribui um char (string de 1 byte) no espaco apontado no arquivo Esse é o diagrama das funções que utilizam essa função:



5.4.2.2 set_campo32()

```
int set_campo32 (
            const int32_t val,
            FILE * fp )
```

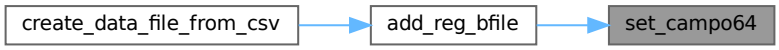
Atribui um inteiro de 32 bits (4 bytes) no espaco apontado no arquivo Esse é o diagrama das funções que utilizam essa função:



5.4.2.3 set_campo64()

```
int set_campo64 (
            const int64_t val,
            FILE * fp )
```

Atribui um inteiro de 64 bits (8 bytes) no espaco apontado no arquivo Esse é o diagrama das funções que utilizam essa função:



5.4.2.4 set_campo_str()

```
int set_campo_str (
            const char * str,
            int32_t * campo_len,
            FILE * fp )
```

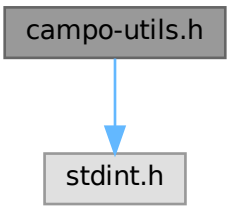
Atribui uma string e seu tamanho no espaco apontado no arquivo

RETORNA: Retorna a quantidade de bytes ocupados pelo campo (incluindo o campo de tamanho) Retorna -1 caso ocorra algum erro Esse é o diagrama das funções que utilizam essa função:

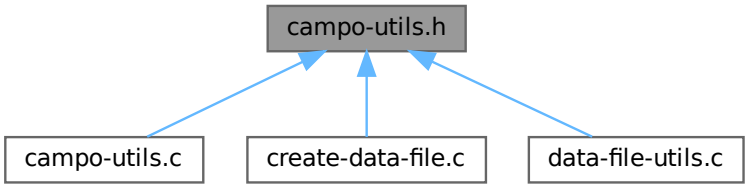
5.5 Referência do Arquivo campo-utils.h

Header file para funcionalidades dos campos de um registro.

`#include <stdint.h>`
Gráfico de dependência de inclusões para campo-utils.h:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Definições e Macros

- #define BUFFER_SIZE 200
- #define STATIC_REG_END_OFFSET 25

Funções

- unsigned char **get_campoc** (FILE ∗fp)
- int32_t **get_campo32** (FILE ∗fp)
- int64_t **get_campo64** (FILE ∗fp)
- char ∗ **get_campo_str** (FILE ∗fp)
- int set_campoc (const unsigned char c, FILE ∗fp)
- int set_campo32 (const int32_t val, FILE ∗fp)
- int set_campo64 (const int64_t val, FILE ∗fp)
- int set_campo_str (const char ∗str, int32_t ∗campo_len, FILE ∗fp)

5.5.1 Descrição detalhada

Header file para funcionalidades dos campos de um registro.

Este arquivo contém as declarações de várias utilidades para ler e escrever campos de um registro em um arquivo binário de dados.

Autores

Nicholas Eiti Dan; N°USP: 14600749

Neri??; N°USP: XXXXXXXX

Versão

1.0

5.5.2 Definições e macros

5.5.2.1 BUFFER_SIZE

```
#define BUFFER_SIZE 200
```

Tamanho alocado aos buffers de chars

5.5.2.2 STATIC_REG_END_OFFSET

```
#define STATIC_REG_END_OFFSET 25
```

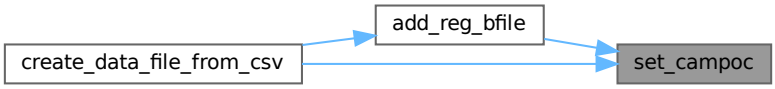
Offset logo apos os campos estaticos de um registro

5.5.3 Funções

5.5.3.1 set_campoc()

```
int set_campoc (
            const unsigned char c,
            FILE * fp )
```

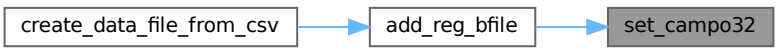
Atribui um char (string de 1 byte) no espaco apontado no arquivo Esse é o diagrama das funções que utilizam essa função:



5.5.3.2 set_campo32()

```
int set_campo32 (
            const int32_t val,
            FILE * fp )
```

Atribui um inteiro de 32 bits (4 bytes) no espaco apontado no arquivo Esse é o diagrama das funções que utilizam essa função:



5.5.3.3 set_campo64()

```
int set_campo64 (
            const int64_t val,
            FILE * fp )
```

Atribui um inteiro de 64 bits (8 bytes) no espaco apontado no arquivo Esse é o diagrama das funções que utilizam essa função:

5.5.3.4 set_campo_str()

```
int set_campo_str (
            const char * str,
            int32_t * campo_len,
            FILE * fp )
```

Atribui uma string e seu tamanho no espaco apontado no arquivo

RETORNA: Retorna a quantidade de bytes ocupados pelo campo (incluindo o campo de tamanho) Retorna -1 caso ocorra algum erro Esse é o diagrama das funções que utilizam essa função:

5.6 campo-utils.h

Ir para a documentação desse arquivo.

```
00001
00013 #ifndef CAMP_UTILS_H
00014 #define CAMP_UTILS_H
00015
00016 #include <stdint.h>
00017
00021 #define BUFFER_SIZE 200
00022
00026 #define STATIC_REG_END_OFFSET 25
00027
00028 unsigned char get_campoc(FILE *fp);
00029
00030 int32_t get_campo32(FILE *fp);
00031
00032 int64_t get_campo64(FILE *fp);
00033
00034 char* get_campo_str(FILE *fp);
00035
```

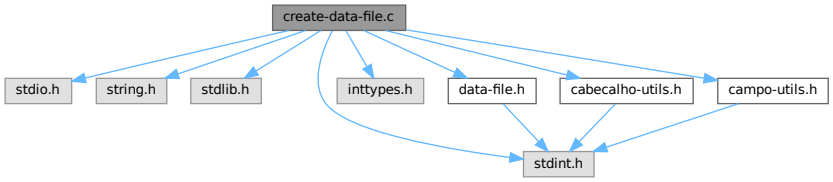
```
00039 int set_campoc(const unsigned char c, FILE *fp);
00040
00044 int set_campo32(const int32_t val, FILE *fp);
00045
00049 int set_campo64(const int64_t val, FILE *fp);
00050
00057 int set_campo_str(const char *str, int32_t *campo_len, FILE *fp);
00058
00059 #endif
```

5.7 Referência do Arquivo create-data-file.c

Implementação da funcionalidade 1 do trabalho definido.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdint.h>
#include <inttypes.h>
#include "data-file.h"
#include "cabecalho-utils.h"
#include "campo-utils.h"
```

Gráfico de dependência de inclusões para create-data-file.c:



Definições e Macros

- #define TRIM_NEWLINE(str) str[strcspn(str, "\n")] = 0

Funções

- static char * get_token_str (char **start_ptr, const char delim)
- static JOGADOR process_jogador (char *line)
- static int add_reg_bfile (const JOGADOR j, FILE *data_bfile_fptr)
- int create_data_file_from_csv (const char *input_filename, const char *output_filename)

5.7.1 Descrição detalhada

Implementação da funcionalidade 1 do trabalho definido.

Autores

Nicholas Eiti Dan; N°USP: 14600749

Neri??; N°USP: XXXXXXXX

Versão

1.0

5.7.2 Definições e macros

5.7.2.1 TRIM_NEWLINE

```
#define TRIM_NEWLINE(
              str ) str[strcspn(str, "\n")] = 0
```

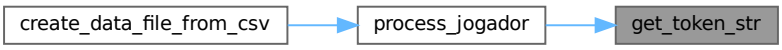
Remove o caracter newline ('
') de uma string

5.7.3 Funções

5.7.3.1 get_token_str()

```
static char * get_token_str (
             char ** start_ptr,
             const char delim )  [static]
```

Implementacao semelhante da funcao strtok com leitura de campos vazios Esse é o diagrama das funções que utilizam essa função:



5.7.3.2 process_jogador()

```
static JOGADOR process_jogador (
             char * line )  [static]
```

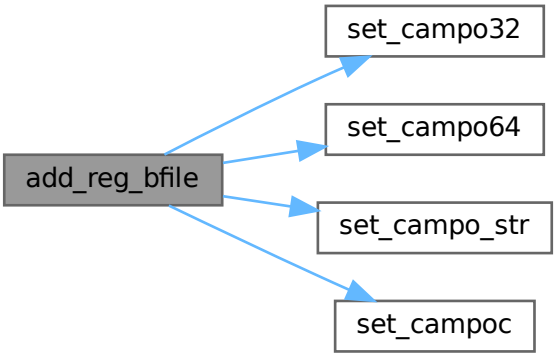
Processa uma linha extraida de um arquivo .csv (string com tokens separadas por ',') e retorna uma variavel do tipo JOGADOR com os dados preenchidos Este é o diagrama das funções utilizadas por essa função:

Esse é o diagrama das funções que utilizam essa função:

5.7.3.3 add_reg_bfile()

```
static int add_reg_bfile (
             const JOGADOR j,
             FILE * data_bfile_fptr )  [static]
```

Função local que acrescenta um registro no arquivo de data binario Este é o diagrama das funções utilizadas por essa função:



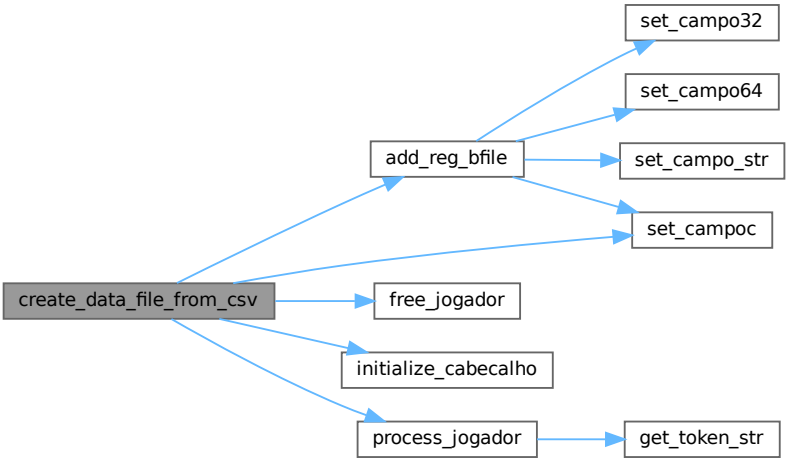
Esse é o diagrama das funções que utilizam essa função:

5.7.3.4 create_data_file_from_csv()

```
int create_data_file_from_csv (
            const char * input_filename,
            const char * output_filename )
```

Funcao da funcionalidade 1 - Criar um arquivo binario de dados a partir de um arquivo de dados .csv

RETORNA: Retorna 0 qundo o arquivo é criado com sucesso, caso nao retorna -1 Este é o diagrama das funções utilizadas por essa função:

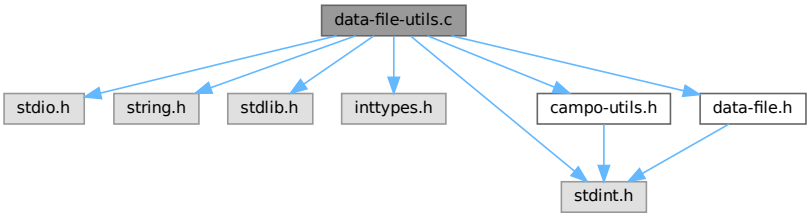


5.8 Referência do Arquivo data-file-utils.c

Source file de algumas utilidades gerais do arquivo binário de dados.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <inttypes.h>
#include <stdint.h>
#include "campo-utils.h"
#include "data-file.h"
```

Gráfico de dependência de inclusões para data-file-utils.c:



Funções

- void free_jogador (JOGADOR *j)
- void print_jogador (JOGADOR j)
- JOGADOR read_jogador_data (FILE *fptr)

5.8.1 Descrição detalhada

Source file de algumas utilidades gerais do arquivo binário de dados.

Alguma descrição específica sobre o arquivo...

Autores

Nicholas Eiti Dan; N°USP: 14600749

Neri??; N°USP: XXXXXXXX

Versão

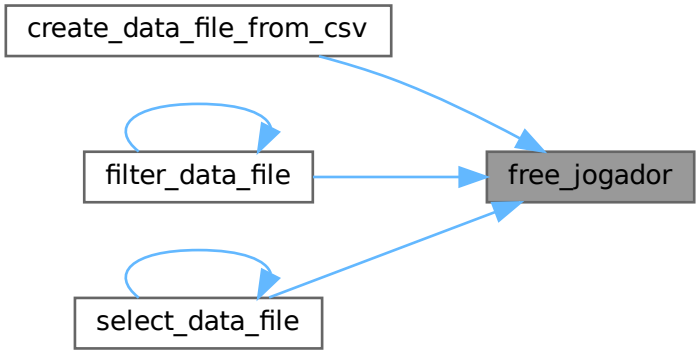
1.0

5.8.2 Funções

5.8.2.1 free_jogador()

```
void free_jogador (
            JOGADOR * j )
```

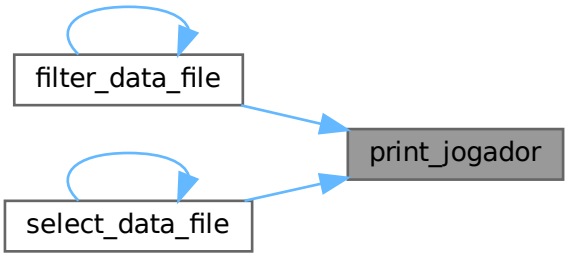
Limpa a memria alocada por um jogador (strings) Esse é o diagrama das funções que utilizam essa função:



5.8.2.2 print_jogador()

```
void print_jogador (
            JOGADOR j )
```

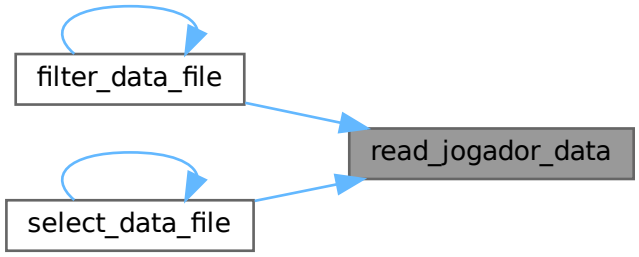
Funcao que imprime um jogador de acordo com os requisitos do projeto Esse é o diagrama das funções que utilizam essa função:



5.8.2.3 read_jogador_data()

```
JOGADOR read_jogador_data (
            FILE * fptr )
```

Le um registro em fptr e atribui os valores na variavel JOGADOR Esse é o diagrama das funções que utilizam essa função:

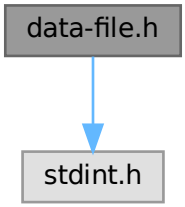


5.9 Referência do Arquivo data-file.h

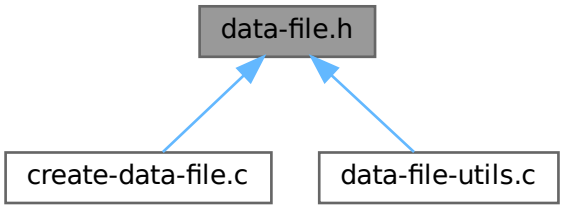
Header file para as funcionalidades relacionadas aos arquivos binarios de dado.

```
#include <stdint.h>
```

Gráfico de dependência de inclusões para data-file.h:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Estruturas de Dados

- struct _jogador

  Estrutura que contém os dados de um jogador, pode representar um registro no arquivo de dados.

Definições e Macros

- #define COLUMN_NAMES "id,idade,nomeJogador,nacionalidade,nomeClube"

Definições de Tipos

- typedef struct _jogador **JOGADOR**

  Tipo da estrutura representando os dados de um jogador.

Funções

- void free_jogador (JOGADOR ∗j)
- void print_jogador (JOGADOR j)
- JOGADOR read_jogador_data (FILE ∗fptr)
- int create_data_file_from_csv (const char ∗input_filename, const char ∗output_filename)
- int select_data_file (const char ∗input_filename)
- int filter_data_file (const int n, const char ∗input_filename)

5.9.1 Descrição detalhada

Header file para as funcionalidades relacionadas aos arquivos binarios de dado.

Este arquivo contém as declarações das funcionalidades exigidas pelo Trabalho Prático Introdutório como especificado na disciplina Organização de Arquivos.

Observação

Cada implementação está separada em um arquivo diferente para diminuir a linhas de código de cada arquivo,

Autores

Nicholas Eiti Dan; N°USP: 14600749

Neri??; N°USP: XXXXXXXX

Versão

1.0

5.9.2 Definições e macros

5.9.2.1 COLUMN_NAMES

```
#define COLUMN_NAMES "id,idade,nomeJogador,nacionalidade,nomeClube"
```

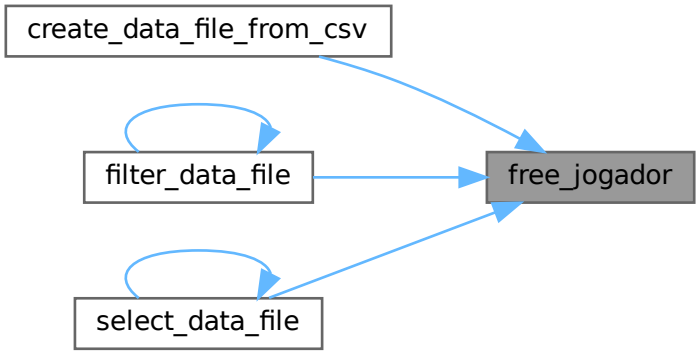
Nome das colunas como registrado no Trabalho Introdutório de Organização de Arquivos

5.9.3 Funções

5.9.3.1 free_jogador()

```
void free_jogador (
            JOGADOR * j )
```

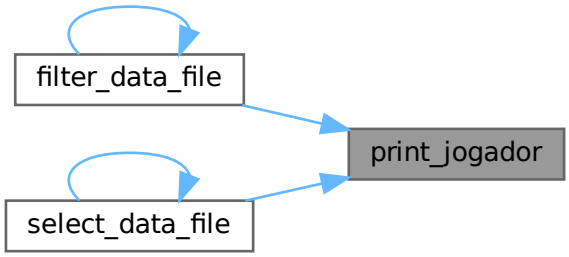
Limpa a memria alocada por um jogador (strings) Esse é o diagrama das funções que utilizam essa função:



5.9.3.2 print_jogador()

```
void print_jogador (
            JOGADOR j )
```

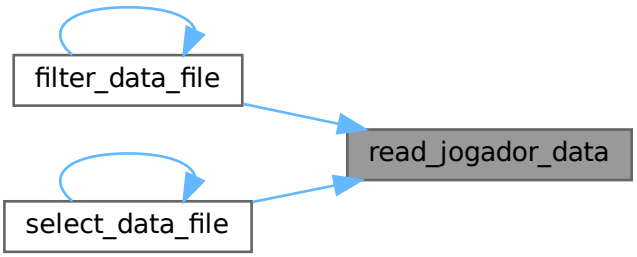
Funcao que imprime um jogador de acordo com os requisitos do projeto Esse é o diagrama das funções que utilizam essa função:



5.9.3.3 read_jogador_data()

```
JOGADOR read_jogador_data (
            FILE * fptr )
```

Le um registro em fptr e atribui os valores na variavel JOGADOR Esse é o diagrama das funções que utilizam essa função:

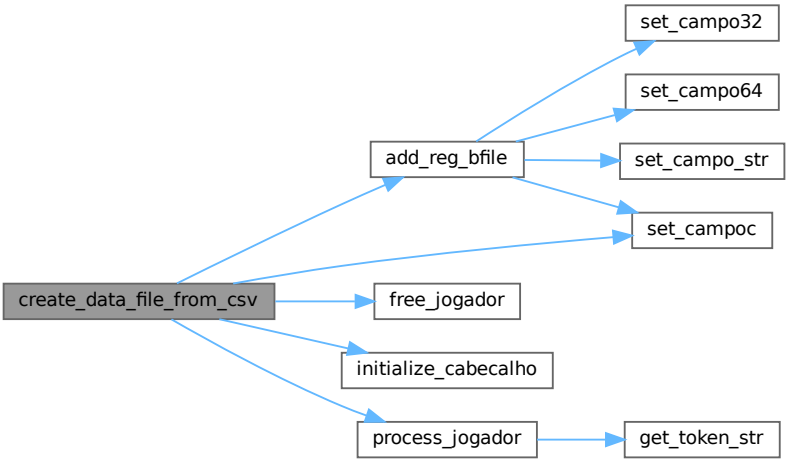


5.9.3.4 create_data_file_from_csv()

```
int create_data_file_from_csv (
            const char * input_filename,
            const char * output_filename )
```

Funcao da funcionalidade 1 - Criar um arquivo binario de dados a partir de um arquivo de dados .csv

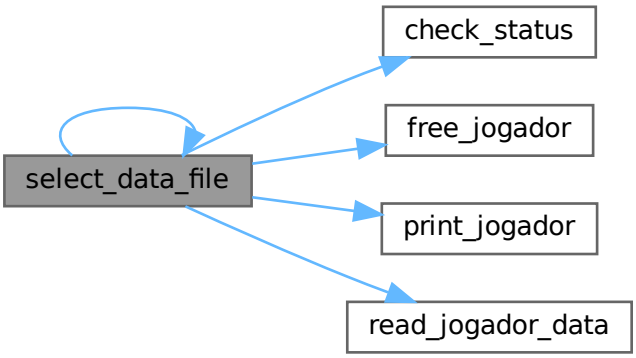
RETORNA: Retorna 0 qundo o arquivo é criado com sucesso, caso nao retorna -1 Este é o diagrama das funções utilizadas por essa função:



5.9.3.5 select_data_file()

```
int select_data_file (
            const char * input_filename )
```

Funcao da funcionalidade 2 - Imprime todos os registros validos (nao removidos) de um arquivo binario de dados
Este é o diagrama das funções utilizadas por essa função:



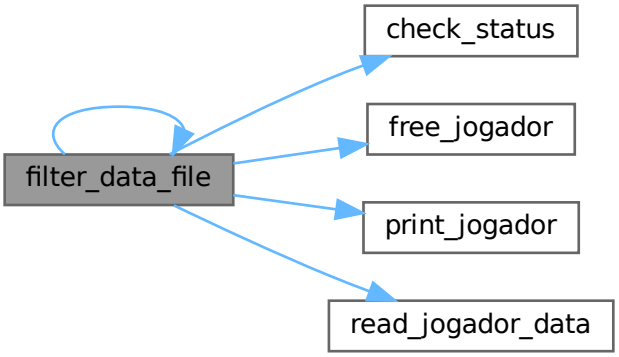
Esse é o diagrama das funções que utilizam essa função:

5.9.3.6 filter_data_file()

```
int filter_data_file (
            const int n,
            const char * input_filename )
```

Funcao da funcionalidade 3 - Filtra os registros validos (nao removidos) de um arquivo binario de dados

Se realiza n pesquisar, cada uma sendo rotulada como Busca x Este é o diagrama das funções utilizadas por essa função:



Esse é o diagrama das funções que utilizam essa função:

5.10 data-file.h

Ir para a documentação desse arquivo.

```
00001
00015 #ifndef DATA_FILE_H
00016 #define DATA_FILE_H
00017
00018 #include <stdint.h>
00019
00020 #define COLUMN_NAMES "id,idade,nomeJogador,nacionalidade,nomeClube"
00029 struct _jogador {
00030     int32_t id;
00031     int32_t idade;
00032     char *nome;
00033     char *nac;
00034     char *clube;
00035 };
00036
00041 typedef struct _jogador JOGADOR;
00042
00046 void free_jogador(JOGADOR *j);
00047
00051 void print_jogador(JOGADOR j);
00052
00056 JOGADOR read_jogador_data(FILE *fptr);
00057
```

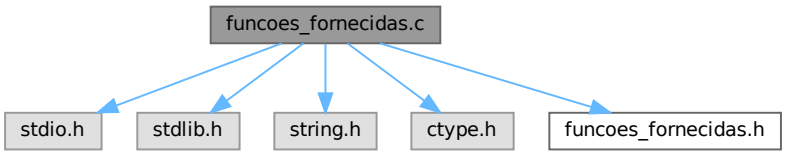
```
00063 int create_data_file_from_csv(const char *input_filename, const char *output_filename);
00064
00068 int select_data_file(const char *input_filename);
00069
00075 int filter_data_file(const int n, const char *input_filename);
00076
00077 #endif
```

5.11 Referência do Arquivo funcoes_fornecidas.c

Source file de funções fornecidas para o projeto.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "funcoes_fornecidas.h"
```

Gráfico de dependência de inclusões para funcoes_fornecidas.c:



Funções

- void binarioNaTela (char ∗nomeArquivoBinario)

  Função que imprime uma hash na tela de um arquivo binário de dados.
- void **scan_quote_string** (char ∗str)

5.11.1 Descrição detalhada

Source file de funções fornecidas para o projeto.

Arquivo disponibilizado na página da disciplina do run.codes pela Professora Cristina.

Autor

Cristina Dutra de Aguiar (cdac@icmc.usp.br)

Versão

1.0

5.11.2 Funções

5.11.2.1 binarioNaTela()

```
void binarioNaTela (
            char * nomeArquivoBinario )
```

Função que imprime uma hash na tela de um arquivo binário de dados.

Use essa função para comparação no run.codes. Lembre-se de ter fechado (fclose) o arquivo anteriormente. Ela vai abrir de novo para leitura e depois fechar (você não vai perder pontos por isso se usar ela).

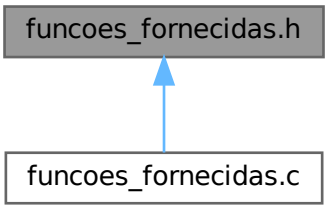
Parâmetros

nomeArquivoBinario	Nome do arquivo a ser hasheado.

5.12 Referência do Arquivo funcoes_fornecidas.h

Header file de funções fornecidas para o projeto.

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Funções

- void binarioNaTela (char ∗nomeArquivoBinario)

  Função que imprime uma hash na tela de um arquivo binário de dados.
- void **scan_quote_string** (char ∗str)

5.12.1 Descrição detalhada

Header file de funções fornecidas para o projeto.

Autor

Cristina Dutra de Aguiar (cdac@icmc.usp.br)

Versão

1.0

5.12.2 Funções

5.12.2.1 binarioNaTela()

```
void binarioNaTela (
            char * nomeArquivoBinario )
```

Função que imprime uma hash na tela de um arquivo binário de dados.

Use essa função para comparação no run.codes. Lembre-se de ter fechado (fclose) o arquivo anteriormente. Ela vai abrir de novo para leitura e depois fechar (você não vai perder pontos por isso se usar ela).

Parâmetros

nomeArquivoBinario	Nome do arquivo a ser hasheado.

5.13 funcoes_fornecidas.h

Ir para a documentação desse arquivo.

```
00001
00010 #ifndef FUNC_FORN
00011 #define FUNC_FORN
00012
00021 void binarioNaTela(char *nomeArquivoBinario);
00022
00023 void scan_quote_string(char *str);
00024
00025 #endif
```

Índice Remissivo