

Entrées et Sorties Bufferisées

Generated by Doxygen 1.8.11

Contents

1	Lectures et Écriture bufferisées	1
2	Data Structure Index	3
2.1	Data Structures	3
3	File Index	5
3.1	File List	5
4	Data Structure Documentation	7
4.1	bfile Struct Reference	7
4.1.1	Detailed Description	7
5	File Documentation	9
5.1	bfile.c File Reference	9
5.1.1	Detailed Description	10
5.1.2	Function Documentation	10
5.1.2.1	bClose(bfile *bf)	10
5.1.2.2	bFill(bfile *bf)	10
5.1.2.3	bFlush(bfile *bf)	10
5.1.2.4	bOpen(const char *path, char mode)	11
5.1.2.5	bRead(void *p, int size, int nb_element, bfile *bf)	11
5.1.2.6	bWrite(void *p, int size, int nb_element, bfile *bf)	11
5.2	format_in_out.c File Reference	12
5.2.1	Detailed Description	12
5.2.2	Function Documentation	12
5.2.2.1	convert_int_to_str(int x)	12
5.2.2.2	fbWrite(bfile *bf, char *format,...)	13
5.2.2.3	is_separator(char c)	13
	Index	15

Chapter 1

Lectures et Écriture bufferisées

Barona Stephanie, Grand Maxence

Compilation

```
1 make all
```

Exécution

main

Ce programme lit un fichier donné en argument, et écrit le contenu lu dans un autre fichier. Ce programme est compilé normalement.

```
1 ./main <filename1> <filename2>
```

- filename1 : Le fichier à lire
- filename2 : Le fichier à écrire

main_static

Ce programme lit un fichier donné en argument, et écrit le contenu lu dans un autre fichier. Ce programme est compilé avec une bibliothèque statique.

```
1 ./main_static <filename1> <filename2>
```

- filename1 : Le fichier à lire
- filename2 : Le fichier à écrire

main_dyn

Ce programme lit un fichier donné en argument, et écrit le contenu lu dans un autre fichier. Ce programme est compilé avec une bibliothèque dynamique.

```
1 ./main_dyn <filename1> <filename2>
```

- filename1 : Le fichier à lire
- filename2 : Le fichier à écrire

generator

Ce programme écrit un fichier au contenu et à la taille aléatoire dans un fichier.

```
1 ./generator <filename>
```

- filename : Le fichier cible

test_format

Ce programme écrit une chaîne formatée dans un fichier, puis la relit et vérifie que les valeurs lues sont toujours les mêmes.

```
1 ./test_format <filename1>
```

- filename1 : Le fichier où sera écrite puis lue la chaîne formatée

Tests

```
1 make tests
```

Le tests sont effectuées par le script script_test.sh :

- Test les fonctions d'écritures et lectures bufférisées avec le programme test_format
- Génère des fichiers au contenu et à la taille aléatoire, donne ce fichier en entré du programme main (resp main_static main_dyn) et vérifie que le fichier <filename2> est identique au fichier <filename1>

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

bfile	Structure pour les écritures et lectures bufferisées	7
--------------	--	---

Chapter 3

File Index

3.1 File List

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

bfile.c	
Implémentation des écritures et lectures bufférisées	9
format_in_out.c	
Implémentation des écritures et lectures formatées	12

Chapter 4

Data Structure Documentation

4.1 bfile Struct Reference

Structure pour les écritures et lectures bufferisées.

Data Fields

- FILE * **f**
- char **mode**
- char * **buffer**
- unsigned **file_seek**
- unsigned **buffer_seek**
- size_t **size_buffer**
- unsigned **eof**

4.1.1 Detailed Description

Structure pour les écritures et lectures bufferisées.

Parameters

<i>f</i>	: le fichier
<i>mode</i>	: Mode de lecteur, E pour écriture, L pour lecture
<i>buffer</i>	: Notre tampon
<i>file_seek</i>	: Curseur de f
<i>file_seek</i>	: Curseur de buffer
<i>size_buffer</i>	: taille du buffer
<i>eof</i>	: booléen indiquant si il reste des choses 'a lire dans le fichier

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

- **bfile.c**

Chapter 5

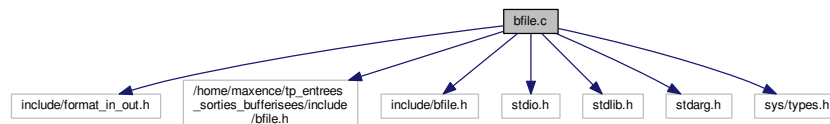
File Documentation

5.1 bfile.c File Reference

Implémentation des écritures et lectures bufférisées.

```
#include "include/format_in_out.h"
#include "include/bfile.h"
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <sys/types.h>
```

Include dependency graph for bfile.c:



Data Structures

- struct **bfile**

Structure pour les écritures et lectures bufferisées.

Functions

- **bfile * bOpen** (const char *path, char mode)

Le but de la fonction est d'ouvrir le fichier path avec le mode d'ouverture en lecture 'L' ou en écriture 'E' et d'allouer la structure de bfile avec son tampon et son fichier. La fonction renvoie NULL si le fichier ne peut pas être ouvert, un bfile sinon.

- int **bClose** (bfile *bf)

Le but de la fonction est de fermer et libérée le fichier pointé sur la structure de donnée de type bfile.

- int **bWrite** (void *p, int size, int nb_element, bfile *bf)

Le but de la fonction est d'écrire nb_element de size octets stockés à l'emplacement mémoire pointé par p, dans le tampon.

- int **bRead** (void *p, int size, int nb_element, **bfile** *bf)

Le but de la fonction est de lire nb_element de size octets dans bf et y stocker à l'emplacement mémoire pointé par p/.

- int **bFlush** (**bfile** *bf)

Le but de la fonction est xde vider le tampon dans le fichier f.

- int **bFill** (**bfile** *bf)

Le but de la fonction est de remplir le tampon avec le fichier f.

5.1.1 Detailed Description

Implémentation des écritures et lectures bufférisées.

5.1.2 Function Documentation

5.1.2.1 int bClose (**bfile** * bf)

Le but de la fonction est de fermer et libérée le fichier pointé sur la structure de donnée de type bfile.

Parameters

<i>bf</i>	: la structure à fermer
-----------	-------------------------

Returns

Un entier retourné par la fonction fclose

5.1.2.2 int bFill (**bfile** * bf)

Le but de la fonction est de remplir le tampon avec le fichier f.

Parameters

<i>bf</i>	
-----------	--

Returns

Le nombre d'octet lu dans le fichier

5.1.2.3 int bFlush (**bfile** * bf)

Le but de la fonction est xde vider le tampon dans le fichier f.

Parameters

<i>bf</i>	
-----------	--

Returns

Le nombre d'octet écrit dans le fichier

5.1.2.4 bfile * bOpen (const char * *path*, char *mode*)

Le but de la fonction est d'ouvrir le fichier *path* avec le mode d'ouverture en lecture 'L' ou en écriture 'E' et d'allouer la structure de *bfile* avec son tampon et son fichier. La fonction renvoie NULL si le fichier ne peut pas être ouvert, un *bfile* sinon.

Parameters

<i>path</i>	: le nom du fichier
<i>mode</i>	: le mode d'ouverture

Returns

le *bfile* créé

5.1.2.5 int bRead (void * *p*, int *size*, int *nb_element*, bfile * *bf*)

Le but de la fonction est de lire *nb_element* de *size* octets dans *bf* et y stocker à l'emplacement mémoire pointé par *p*.

Parameters

<i>p</i>	: le pointeur à remplir
<i>size</i>	: la taille des données à lire
<i>nb_element</i>	: le nombre de fois que nous lisons.
<i>bf</i>	

Returns

Le nombre d'octets lus.

5.1.2.6 int bWrite (void * *p*, int *size*, int *nb_element*, bfile * *bf*)

Le but de la fonction est d'écrire *nb_element* de *size* octets stockés à l'emplacement mémoire pointé par *p*, dans le tampon.

Parameters

<i>p</i>	: le pointeur à écrire
<i>size</i>	: la taille des données à écrire
<i>nb_element</i>	: le nombre de fois où on écrit.
<i>bf</i>	

Returns

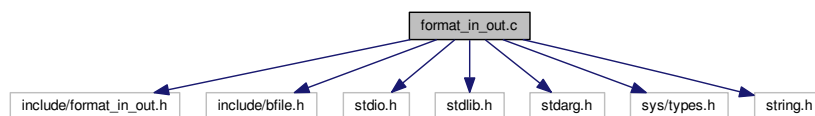
le nombre d'octets écrits

5.2 format_in_out.c File Reference

Implémentation des écritures et lectures formatées.

```
#include "include/format_in_out.h"
#include "include/bfile.h"
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <sys/types.h>
#include <string.h>
```

Include dependency graph for format_in_out.c:



Functions

- char * **convert_int_to_str** (int x)
Le but de la fonction est de convertir un entier en chaîne.
- unsigned **is_separator** (char c)
Le but de la fonction est de vérifier qu'un caractère est un séparateur ou non.
- int **fbWrite** (bfile *bf, char *format,...)
Le but de la fonction est d'écrire la chaîne format dans bf->f en remplaçant les % par les valeurs des variables.
- int **fbRead** (bfile *bf, char *format,...)

5.2.1 Detailed Description

Implémentation des écritures et lectures formatées.

5.2.2 Function Documentation

5.2.2.1 char * convert_int_to_str (int x)

Le but de la fonction est de convertir un entier en chaîne.

Parameters

<i>x</i>	: l'entier à convertir en chaîne
----------	----------------------------------

Returns

Une chaîne de caractère

5.2.2.2 int fbWrite (bfile * *bf*, char * *format*, ...)

Le but de la fonction est d'écrire la chaîne format dans bf->f en remplaçant les % par les valeurs des variables.

Parameters

<i>bf</i>	: la structure du fichier bufférisé
<i>format</i>	: La chaîne formatée
...	: Les différentes variables à écrire

Returns

Le nombre d'octet écrit

5.2.2.3 unsigned is_separator (char *c*)

Le but de la fonction est de vérifier qu'un caractère est un séparateur ou non.

Parameters

<i>c</i>	: lcaractère à tester
----------	-----------------------

Returns

Un booléen, true si c est séparateur, false sinon

Index

- bClose
 - bfile.c, 10
- bFill
 - bfile.c, 10
- bFlush
 - bfile.c, 10
- bOpen
 - bfile.c, 11
- bRead
 - bfile.c, 11
- bWrite
 - bfile.c, 11
- bfile, 7
- bfile.c, 9
 - bClose, 10
 - bFill, 10
 - bFlush, 10
 - bOpen, 11
 - bRead, 11
 - bWrite, 11
- convert_int_to_str
 - format_in_out.c, 12
- fbWrite
 - format_in_out.c, 13
- format_in_out.c, 12
 - convert_int_to_str, 12
 - fbWrite, 13
 - is_separator, 13
- is_separator
 - format_in_out.c, 13