libfmanc

Generated by Doxygen 1.9.5

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Module Documentation

4.1 Core C source code

Files

- file analyze.c
- file fileMan.c

  These functions are made to do operate simple operation on files or file names, when there is no neee to analyze something like orccurrences, ...

4.1.1 Detailed Description

Author

Axel PASCON (a.k.a. brvtalcake)

Date

2022

4.2 Core C headers

Modules

- Core lib C headers
- Main C header

4.2.1 Detailed Description

Author

Axel PASCON (a.k.a. brvtalcake)

Date

2022

4.3 Core lib C headers

Files

- file analyze.h

  This header contains type definitions and function declarations that are written in this file .
- file fileMan.h

  This header contains macro definitions and function declarations that are written in this file. These functions are made to operate simple operation on files or file names, when there is no neee to analyze something like orccurrences, ...

4.3.1 Detailed Description

Author

Axel PASCON (a.k.a. brvtalcake)

Date

2022

4.4 C source code for my personnal project

Modules

- C source code for my personnal project, made by me
- C source code for my personnal project, made by me and flex

4.4.1 Detailed Description

Author

Axel PASCON (a.k.a. brvtalcake)

Date

2022

4.5 C source code for my personnal project, made by me

Files

- file fcmx.c

  This header contains function declarations that I will use for one of my project (which isn't still on gitHub)

4.5.1 Detailed Description

Author

Axel PASCON (a.k.a. brvtalcake)

Date

2022

4.6 C Headers for my personnal project

Modules

- C Headers for my personnal project, made by me
- C Headers for my personnal project, made by me and flex

4.6.1 Detailed Description

Author

Axel PASCON (a.k.a. brvtalcake)

Date

2022

4.7 C Headers for my personnal project, made by me

Files

- file fcmx.h

  This header contains function declarations that are wittent on this file and that I will use for one of my project (which isn't still on gitHub)

4.7.1 Detailed Description

Author

Axel PASCON (a.k.a. brvtalcake)

Date

2022

4.8 Main C header

Files

- file fmanc.h

  This is the main header of the lib, where all of the headers are included.

4.8.1 Detailed Description

Author

Axel PASCON (a.k.a. brvtalcake)

Date

2022

4.9 C source code for my personnal project, made by me and flex

Files

- file lex_yy.c

  This is the file with the lexical scanner.

4.9.1 Detailed Description

Author

Axel PASCON (a.k.a. brvtalcake)

Date

2022

4.10 C Headers for my personnal project, made by me and flex

Files

- file lex_yy.h

  This header contains a function written by flex to delete C-style comments in a file.

4.10.1 Detailed Description

Author

Axel PASCON (a.k.a. brvtalcake)

Date

2022

4.11 Flex source files

Files

- file noComments.l

4.11.1 Detailed Description

Author

Axel PASCON (a.k.a. brvtalcake)

Date

2022

Chapter 5

Data Structure Documentation

5.1 stringOccurrences Struct Reference

Data Fields

- size_t **charCount**
- long * **pos**

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

- analyze.h

Chapter 6

File Documentation

6.1 analyze.c File Reference

Functions

- long int **countCharInFile** (char ∗filePath)
- stringOccurrences ∗ **init_StringOccurences** (size_t sizeOfString)
- void **free_stringOccurrences** (stringOccurrences ∗toBeDeleted)
- stringOccurrences ∗ **searchStringInFile** (char ∗filePath, char ∗toSearch)
- int replaceStringInFile (char ∗filePath, char ∗toReplaceString, char ∗toAddString)

  This function replace a wide-character string by another one.

6.1.1 Detailed Description

These functions are made to analyze files content and make operations on it.

Author

Axel PASCON (a.k.a. brvtalcake)

Date

2022

6.1.2 Function Documentation

6.1.2.1 replaceStringInFile()

```
int replaceStringInFile (
            char * filePath,
            char * toReplaceString,
            char * toAddString )
```

This function replace a wide-character string by another one.

If there is a problem during the call of the function, it will print strerr(errno) on stderr, and/or return an appropriate value

Parameters

filePath	The file path of the document (for example : "./src/example.txt", or "exemple.txt" if the program is called from the same folder). For paths with backslashes, you will need to double it ("\\"), but no need to do this if you use a path that comes from argv arg of your main func.
toReplaceString	The string to replace
toAddString	The string to add

Returns

Returns 0 if all good.

Return values

-4	Problem when converting the char string to wide char string
-3	Problem when calling setlocale()
-2	Problem when trying to get the ext, name or path of the file
-1	Problem when opening files
1	Problem when writing into the new file
2	Problem when renaming or removing files
3	The string to replace isn't in the file

6.2 analyze.h File Reference

This header contains type definitions and function declarations that are written in this file .

Data Structures

- struct stringOccurrences

Functions

- long int **countCharInFile** (char ∗filePath)
- stringOccurrences ∗ **init_StringOccurences** (size_t sizeOfString)
- void **free_stringOccurrences** (stringOccurrences ∗toBeDeleted)
- stringOccurrences ∗ **searchStringInFile** (char ∗filePath, char ∗toSearch)
- int replaceStringInFile (char ∗filePath, char ∗toReplaceString, char ∗toAddString)

  This function replace a wide-character string by another one.

6.2.1 Detailed Description

This header contains type definitions and function declarations that are written in this file .

These functions are made to analyze files content and make operations on it.

Author

Axel PASCON (a.k.a. brvtalcake)

Date

2022

6.2.2 Function Documentation

6.2.2.1 replaceStringInFile()

```
int replaceStringInFile (
            char * filePath,
            char * toReplaceString,
            char * toAddString )
```

This function replace a wide-character string by another one.

If there is a problem during the call of the function, it will print strerr(errno) on stderr, and/or return an appropriate value

Parameters

filePath	The file path of the document (for example : "./src/example.txt", or "exemple.txt" if the program is called from the same folder). For paths with backslashes, you will need to double it ("\\"), but no need to do this if you use a path that comes from argv arg of your main func.
toReplaceString	The string to replace
toAddString	The string to add

Returns

Returns 0 if all good.

Return values

-4	Problem when converting the char string to wide char string
-3	Problem when calling setlocale()
-2	Problem when trying to get the ext, name or path of the file
-1	Problem when opening files
1	Problem when writing into the new file
2	Problem when renaming or removing files
3	The string to replace isn't in the file

6.3 analyze.h

Go to the documentation of this file.

```
1
34 #ifndef ANALYZE_H
35 #define ANALYZE_H
36
37 typedef struct
38 {
39     size_t charCount;
40     long *pos;
41 }stringOccurrences;
42
```

```
43 long int countCharInFile(char *filePath);
44 stringOccurrences *init_StringOccurences(size_t sizeOfString);
45 void free_stringOccurrences(stringOccurrences *toBeDeleted);
46 stringOccurrences *searchStringInFile(char *filePath, char *toSearch);
47
65 int replaceStringInFile(char *filePath, char *toReplaceString, char *toAddString);
66
67
68 #endif
69
```

6.4 fcmx.c File Reference

This header contains function declarations that I will use for one of my project (which isn't still on gitHub)

6.4.1 Detailed Description

This header contains function declarations that I will use for one of my project (which isn't still on gitHub)

Author

Axel PASCON (a.k.a. brvtalcake)

Date

2022

6.5 fcmx.h File Reference

This header contains function declarations that are wittent on this file and that I will use for one of my project (which isn't still on gitHub)

Functions

- int **copyFileWithoutStrings** (const unsigned int argc, char ∗filePath,...)

6.5.1 Detailed Description

This header contains function declarations that are wittent on this file and that I will use for one of my project (which isn't still on gitHub)

Author

Axel PASCON (a.k.a. brvtalcake)

Date

2022

6.6 fcmx.h

Go to the documentation of this file.

```
1
33 #ifndef FCMX_H
34 #define FCMX_H
35
36
37
38 int copyFileWithoutStrings(const unsigned int argc, char *filePath, ...); // to do
39
40
41
42 #endif
43
```

6.7 fileMan.c File Reference

These functions are made to do operate simple operation on files or file names, when there is no neee to analyze something like orccurrences, ...

Functions

- char * **copyFileWithoutTabAndLineBreak** (char *sourceFilePath, char **pathToCopy)
- void **fgetFileExtension** (char *sourceFileName, char *extension)
- void **fgetFileName** (char *sourceFilePath, char *fileName)
- void **fgetFilePath** (char *sourceFilePath, char *filePath)

6.7.1 Detailed Description

These functions are made to do operate simple operation on files or file names, when there is no neee to analyze something like orccurrences, ...

Author

Axel PASCON (a.k.a. brvtalcake)

Date

2022

6.8 fileMan.h File Reference

This header contains macro definitions and function declarations that are written in this file. These functions are made to operate simple operation on files or file names, when there is no neee to analyze something like orccurrences, ...

Macros

- #define **MAX_FEXT_SIZE** 50
- #define **MAX_FNAME_SIZE** 256
- #define **MAX_FPATH_SIZE** 512
- #define **getFileExtension**(sourceFileName, extension) char extension[MAX_FEXT_SIZE]; fgetFile↩ Extension(sourceFileName, extension)
- #define **getFileName**(sourceFilePath, name) char name[MAX_FNAME_SIZE]; fgetFileName(sourceFile↩ Path, name)
- #define **getFilePath**(sourceFilePath, path) char path[MAX_FPATH_SIZE]; fgetFilePath(sourceFilePath, path)

Functions

- char ∗ **copyFileWithoutTabAndLineBreak** (char ∗sourceFilePath, char ∗∗pathToCopy)
- int **copyFileWithoutStrings** (const unsigned int argc, char ∗filePath,...)
- void **fgetFileExtension** (char ∗sourceFileName, char ∗extension)
- void **fgetFileName** (char ∗sourceFilePath, char ∗fileName)
- void **fgetFilePath** (char ∗sourceFilePath, char ∗filePath)

6.8.1 Detailed Description

This header contains macro definitions and function declarations that are written in this file. These functions are made to operate simple operation on files or file names, when there is no neee to analyze something like orccurrences, ...

Author

Axel PASCON (a.k.a. brvtalcake)

Date

2022

6.9 fileMan.h

Go to the documentation of this file.

```
1
37 #ifndef FILEMAN_H
38 #define FILEMAN_H
39
40 #ifndef MAX_FEXT_SIZE
41 #define MAX_FEXT_SIZE 50
42 #endif
43
44 #ifndef MAX_FNAME_SIZE
45 #define MAX_FNAME_SIZE 256
46 #endif
47
48 #ifndef MAX_FPATH_SIZE
49 #define MAX_FPATH_SIZE 512
50 #endif
51
52 #ifndef getFileExtension
53 #define getFileExtension(sourceFileName, extension) char extension[MAX_FEXT_SIZE];
      fgetFileExtension(sourceFileName, extension)
54 #endif
55
56 #ifndef getFileName
57 #define getFileName(sourceFilePath, name) char name[MAX_FNAME_SIZE]; fgetFileName(sourceFilePath, name)
```

```
58 #endif
59
60 #ifndef getFilePath
61 #define getFilePath(sourceFilePath, path) char path[MAX_FPATH_SIZE]; fgetFilePath(sourceFilePath, path)
62 #endif
63
64
65 char *copyFileWithoutTabAndLineBreak(char *sourceFilePath, char **pathToCopy); // copied file will be
      named like <sourceFile name>_copied
66 int copyFileWithoutStrings(const unsigned int argc, char *filePath, ...); // to do
67 void fgetFileExtension(char *sourceFileName, char *extension);
68 void fgetFileName(char *sourceFilePath, char *fileName);
69 void fgetFilePath(char *sourceFilePath, char *filePath);
70
71
72 #endif
73
74
```

6.10 fmanc.h File Reference

This is the main header of the lib, where all of the headers are included.

6.10.1 Detailed Description

This is the main header of the lib, where all of the headers are included.

If you don't want to have troubles, just include this one instead of including the others one by one. If you want to use the functions defined in this or this source files, then write something like
`#define USE_FCMX TRUE`

(actually, just define USE_FCMX with a certain value, no matter what it is).

Author

Axel PASCON (a.k.a. brvtalcake)

Date

2022

6.11 fmanc.h

Go to the documentation of this file.

```
1
36 #ifndef FMANC_H
37 #define FMANC_H
38
39 #include "fileMan.h"
40 #include "analyze.h"
41 #if defined(USE_FCMX)
42 #include "fcmx.h"
43 #include "./notByMe/lex_yy.h"
44 #endif
45
46 #endif
47
```

6.12 notByMe/lex_yy.c File Reference

This is the file with the lexical scanner.

Macros

- #define **YY_INT_ALIGNED** short int
- #define **FLEX_SCANNER**
- #define **YY_FLEX_MAJOR_VERSION** 2
- #define **YY_FLEX_MINOR_VERSION** 6
- #define **YY_FLEX_SUBMINOR_VERSION** 4
- #define **FLEX_BETA**
- #define **FLEXINT_H**
- #define **INT8_MIN** (-128)
- #define **INT16_MIN** (-32767-1)
- #define **INT32_MIN** (-2147483647-1)
- #define **INT8_MAX** (127)
- #define **INT16_MAX** (32767)
- #define **INT32_MAX** (2147483647)
- #define **UINT8_MAX** (255U)
- #define **UINT16_MAX** (65535U)
- #define **UINT32_MAX** (4294967295U)
- #define **SIZE_MAX** (~(size_t)0)
- #define **yyconst** const
- #define **yynoreturn**
- #define **YY_NULL** 0
- #define **YY_SC_TO_UI**(c) ((YY_CHAR) (c))
- #define **BEGIN** (yy_start) = 1 + 2 *
- #define **YY_START** (((yy_start) - 1) / 2)
- #define **YYSTATE** YY_START
- #define **YY_STATE_EOF**(state) (YY_END_OF_BUFFER + state + 1)
- #define **YY_NEW_FILE** yyrestart(yyin)
- #define **YY_END_OF_BUFFER_CHAR** 0
- #define **YY_BUF_SIZE** 16384
- #define **YY_STATE_BUF_SIZE** ((YY_BUF_SIZE + 2) * sizeof(yy_state_type))
- #define **YY_TYPEDEF_YY_BUFFER_STATE**
- #define **YY_TYPEDEF_YY_SIZE_T**
- #define **EOB_ACT_CONTINUE_SCAN** 0
- #define **EOB_ACT_END_OF_FILE** 1
- #define **EOB_ACT_LAST_MATCH** 2
- #define **YY_LESS_LINENO**(n)
- #define **YY_LINENO_REWIND_TO**(ptr)
- #define yyless(n)
- #define **unput**(c) yyunput(c, (yytext_ptr))
- #define **YY_STRUCT_YY_BUFFER_STATE**
- #define **YY_BUFFER_NEW** 0
- #define **YY_BUFFER_NORMAL** 1
- #define **YY_BUFFER_EOF_PENDING** 2
- #define YY_CURRENT_BUFFER
- #define **YY_CURRENT_BUFFER_LVALUE** (yy_buffer_stack)[(yy_buffer_stack_top)]
- #define **YY_FLUSH_BUFFER** yy_flush_buffer(YY_CURRENT_BUFFER)
- #define **yy_new_buffer** yy_create_buffer
- #define yy_set_interactive(is_interactive)

- #define yy_set_bol(at_bol)
- #define **YY_AT_BOL**() (YY_CURRENT_BUFFER_LVALUE->yy_at_bol)
- #define **yywrap**() (/*CONSTCOND*/1)
- #define **YY_SKIP_YYWRAP**
- #define **yytext_ptr** yytext
- #define YY_DO_BEFORE_ACTION
- #define **YY_NUM_RULES** 17
- #define **YY_END_OF_BUFFER** 18
- #define **REJECT** reject_used_but_not_detected
- #define **yymore**() yymore_used_but_not_detected
- #define **YY_MORE_ADJ** 0
- #define **YY_RESTORE_YY_MORE_OFFSET**
- #define **INITIAL** 0
- #define **String** 1
- #define **Char** 2
- #define **Comment** 3
- #define **CPPComment** 4
- #define **YY_EXTRA_TYPE** void *
- #define **YY_READ_BUF_SIZE** 8192
- #define **ECHO** do { if (fwrite(yytext, (size_t) yyleng, 1, yyout)) {} } while (0)
- #define YY_INPUT(buf, result, max_size)
- #define **yyterminate**() return YY_NULL
- #define **YY_START_STACK_INCR** 25
- #define **YY_FATAL_ERROR**(msg) yy_fatal_error(msg)
- #define **YY_DECL_IS_OURS** 1
- #define **YY_DECL** int yylex (void)
- #define **YY_USER_ACTION**
- #define **YY_BREAK** /*LINTED*/break;
- #define **YY_RULE_SETUP** YY_USER_ACTION
- #define **YY_EXIT_FAILURE** 2
- #define yyless(n)
- #define **YYTABLES_NAME** "yytables"

Typedefs

- typedef signed char **flex_int8_t**
- typedef short int **flex_int16_t**
- typedef int **flex_int32_t**
- typedef unsigned char **flex_uint8_t**
- typedef unsigned short int **flex_uint16_t**
- typedef unsigned int **flex_uint32_t**
- typedef struct yy_buffer_state * **YY_BUFFER_STATE**
- typedef size_t **yy_size_t**
- typedef flex_uint8_t **YY_CHAR**
- typedef int **yy_state_type**

Functions

- void **yyrestart** (FILE ∗input_file)
- void **yy_switch_to_buffer** (YY_BUFFER_STATE new_buffer)
- YY_BUFFER_STATE **yy_create_buffer** (FILE ∗file, int size)
- void **yy_delete_buffer** (YY_BUFFER_STATE b)
- void **yy_flush_buffer** (YY_BUFFER_STATE b)
- void **yypush_buffer_state** (YY_BUFFER_STATE new_buffer)
- void **yypop_buffer_state** (void)
- YY_BUFFER_STATE **yy_scan_buffer** (char ∗base, yy_size_t size)
- YY_BUFFER_STATE **yy_scan_string** (const char ∗yy_str)
- YY_BUFFER_STATE **yy_scan_bytes** (const char ∗bytes, int len)
- void ∗ **yyalloc** (yy_size_t)
- void ∗ **yyrealloc** (void ∗, yy_size_t)
- void **yyfree** (void ∗)
- int **fileno** (FILE ∗)
- int **yylex_destroy** (void)
- int **yyget_debug** (void)
- void **yyset_debug** (int debug_flag)
- YY_EXTRA_TYPE **yyget_extra** (void)
- void **yyset_extra** (YY_EXTRA_TYPE user_defined)
- FILE ∗ **yyget_in** (void)
- void **yyset_in** (FILE ∗_in_str)
- FILE ∗ **yyget_out** (void)
- void **yyset_out** (FILE ∗_out_str)
- int **yyget_leng** (void)
- char ∗ **yyget_text** (void)
- int **yyget_lineno** (void)
- void **yyset_lineno** (int _line_number)
- int **yylex** (void)
- **if** (!(yy_init))
- int **deleteCStyleComments** (char ∗filePath)

Variables

- int **yyleng**
- FILE ∗ **yyin** = NULL
- FILE ∗ **yyout** = NULL
- int **yylineno** = 1
- char ∗ **yytext**
- int **yy_flex_debug** = 0
- YY_DECL
- char ∗ **yy_cp**
- char ∗ **yy_bp**
- int **yy_act**
- int(∗ **jj2_junk**)(void) = input

6.12.1 Detailed Description

This is the file with the lexical scanner.

I let it here, but if you know how to use these functions (like me, lol), don't use them. Just Use the one defined in lex_yy.h. For more informations about my func, just go on the header's description.

Author

Axel PASCON (a.k.a. brvtalcake)

Date

2022

6.12.2 Macro Definition Documentation

6.12.2.1 YY_CURRENT_BUFFER

```
#define YY_CURRENT_BUFFER
```

Value:

```
( (yy_buffer_stack) \
?  (yy_buffer_stack)[(yy_buffer_stack_top)] \
:  NULL)
```

6.12.2.2 YY_DO_BEFORE_ACTION

```
#define YY_DO_BEFORE_ACTION
```

Value:

```
    (yytext_ptr) = yy_bp; \
    yyleng = (int) (yy_cp - yy_bp); \
    (yy_hold_char) = *yy_cp; \
    *yy_cp = '\0'; \
    (yy_c_buf_p) = yy_cp;
```

6.12.2.3 YY_INPUT

```
#define YY_INPUT(
               buf,
               result,
               max_size )
```

Value:

```
    if ( YY_CURRENT_BUFFER_LVALUE->yy_is_interactive ) \
        { \
        int c = '*'; \
        int n; \
        for ( n = 0; n < max_size && \
                 (c = getc( yyin )) != EOF && c != '\n'; ++n ) \
            buf[n] = (char) c; \
        if ( c == '\n' ) \
            buf[n++] = (char) c; \
        if ( c == EOF && ferror( yyin ) ) \
            YY_FATAL_ERROR( "input in flex scanner failed" ); \
        result = n; \
        } \
    else \
        { \
        errno=0; \
        while ( (result = (int) fread(buf, 1, (yy_size_t) max_size, yyin)) == 0 && ferror(yyin)) \
            { \
            if( errno != EINTR) \
                { \
                YY_FATAL_ERROR( "input in flex scanner failed" ); \
                break; \
                } \
            errno=0; \
            clearerr(yyin); \
            } \
        }\
\
```

6.12.2.4 yy_set_bol

```
#define yy_set_bol(
               at_bol )
```

Value:

```
    { \
    if ( !  YY_CURRENT_BUFFER ){\
        yyensure_buffer_stack (); \
        YY_CURRENT_BUFFER_LVALUE =    \
            yy_create_buffer( yyin, YY_BUF_SIZE ); \
    } \
    YY_CURRENT_BUFFER_LVALUE->yy_at_bol = at_bol; \
    }
```

6.12.2.5 yy_set_interactive

```
#define yy_set_interactive(
               is_interactive )
```

Value:

```
    { \
    if ( !  YY_CURRENT_BUFFER ){ \
        yyensure_buffer_stack (); \
        YY_CURRENT_BUFFER_LVALUE =    \
            yy_create_buffer( yyin, YY_BUF_SIZE ); \
    } \
    YY_CURRENT_BUFFER_LVALUE->yy_is_interactive = is_interactive; \
    }
```

6.12.2.6 yyless [1/2]

```
#define yyless(
               n )
```

Value:

```
    do \
        { \
        /* Undo effects of setting up yytext.  */ \
        int yyless_macro_arg = (n); \
        YY_LESS_LINENO(yyless_macro_arg);\
        *yy_cp = (yy_hold_char); \
        YY_RESTORE_YY_MORE_OFFSET \
        (yy_c_buf_p) = yy_cp = yy_bp + yyless_macro_arg - YY_MORE_ADJ; \
        YY_DO_BEFORE_ACTION; /* set up yytext again */ \
        } \
    while ( 0 )
```

6.12.2.7 yyless [2/2]

```
#define yyless(
               n )
```

Value:

```
    do \
        { \
        /* Undo effects of setting up yytext.  */ \
        int yyless_macro_arg = (n); \
        YY_LESS_LINENO(yyless_macro_arg);\
        yytext[yyleng] = (yy_hold_char); \
        (yy_c_buf_p) = yytext + yyless_macro_arg; \
        (yy_hold_char) = *(yy_c_buf_p); \
        *(yy_c_buf_p) = '\0'; \
        yyleng = yyless_macro_arg; \
        } \
    while ( 0 )
```

6.12.3 Variable Documentation

6.12.3.1 YY_DECL

```
YY_DECL
```

Initial value:

```
{
    yy_state_type yy_current_state
```

The main scanner function which does all the work.

6.13 notByMe/lex_yy.h File Reference

This header contains a function written by flex to delete C-style comments in a file.

Functions

- int **deleteCStyleComments** (char ∗filePath)

6.13.1 Detailed Description

This header contains a function written by flex to delete C-style comments in a file.

Author

Axel PASCON (a.k.a. brvtalcake)

Date

2022

6.14 lex_yy.h

Go to the documentation of this file.

```
1
33 #ifndef LEX_YY_H
34 #define LEX_YY_H
35
36 int deleteCStyleComments(char *filePath);
37
38
39 #endif
```

6.15 notByMe/noComments.l File Reference

Functions

- int **fileno** (FILE ∗)
- int **yylex** (void)
- int **deleteCStyleComments** (char ∗filePath)

Variables

- void(∗ **jj_junk**)(int, char ∗) = yyunput
- int(∗ **jj2_junk**)(void) = input

6.15.1 Detailed Description

Author

Axel PASCON (a.k.a. brvtalcake)

Date

2022

Index