File System Manager

Generated by Doxygen 1.9.6

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 _MBR Struct Reference

`#include <FileManagerLibrary.h>`

Data Fields

- uint8_t bootcode [446]
- partition part [4]
- uint16_t signature

3.1.1 Detailed Description

Defines the structure of the MBR.

3.1.2 Field Documentation

3.1.2.1 bootcode

`uint8_t bootcode[446]`

3.1.2.2 part

`partition part[4]`

3.1.2.3 signature

```
uint16_t signature
```

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

- C:/Users/trish/Desktop/Code/School/CSC430/FileManager/include/FileManagerLibrary.h

3.2 _partition Struct Reference

```
#include <FileManagerLibrary.h>
```

Data Fields

- uint8_t bootable
- uint8_t first_chs [3]
- uint8_t type
- uint8_t last_chs [3]
- uint32_t lba_offset
- uint32_t sector_count

3.2.1 Detailed Description

Defines the structure of a partition in the MBR.

3.2.2 Field Documentation

3.2.2.1 bootable

```
uint8_t bootable
```

3.2.2.2 first_chs

```
uint8_t first_chs[3]
```

3.2.2.3 last_chs

```
uint8_t last_chs[3]
```

3.2.2.4 lba_offset

```
uint32_t lba_offset
```

3.2.2.5 sector_count

```
uint32_t sector_count
```

3.2.2.6 type

```
uint8_t type
```

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

- C:/Users/trish/Desktop/Code/School/CSC430/FileManager/include/FileManagerLibrary.h

3.3 BootSector Struct Reference

```
#include <FileManagerLibrary.h>
```

Data Fields

- uint8_t jumpCode [3]
- uint8_t oemName [8]
- uint16_t bytesPerSector
- uint8_t sectorsPerCluster
- uint16_t reservedSectors
- uint8_t numCopiesOfFAT
- uint16_t maxRootDirEntries
- uint16_t numSectorsSmall
- uint8_t mediaDescriptor
- uint16_t sectorsPerFAT
- uint16_t sectorsPerTrack
- uint16_t numHeads
- uint32_t numHiddenSectors
- uint32_t numSectorsLarge
- uint16_t logicalDriveNumber
- uint8_t extendedSignature
- uint32_t serialNumber
- uint8_t volumeName [11]
- uint8_t fatName [8]
- uint8_t executableCode [448]
- uint16_t executableMarker

3.3.1 Detailed Description

Defines the structure of a boot sector.

3.3.2 Field Documentation

3.3.2.1 bytesPerSector

```
uint16_t bytesPerSector
```

3.3.2.2 executableCode

```
uint8_t executableCode[448]
```

3.3.2.3 executableMarker

```
uint16_t executableMarker
```

3.3.2.4 extendedSignature

```
uint8_t extendedSignature
```

3.3.2.5 fatName

```
uint8_t fatName[8]
```

3.3.2.6 jumpCode

```
uint8_t jumpCode[3]
```

3.3.2.7 logicalDriveNumber

```
uint16_t logicalDriveNumber
```

3.3.2.8 maxRootDirEntries

```
uint16_t maxRootDirEntries
```

3.3.2.9 mediaDescriptor

```
uint8_t mediaDescriptor
```

3.3.2.10 numCopiesOfFAT

```
uint8_t numCopiesOfFAT
```

3.3.2.11 numHeads

```
uint16_t numHeads
```

3.3.2.12 numHiddenSectors

```
uint32_t numHiddenSectors
```

3.3.2.13 numSectorsLarge

```
uint32_t numSectorsLarge
```

3.3.2.14 numSectorsSmall

```
uint16_t numSectorsSmall
```

3.3.2.15 oemName

```
uint8_t oemName[8]
```

3.3.2.16 reservedSectors

```
uint16_t reservedSectors
```

3.3.2.17 sectorsPerCluster

```
uint8_t sectorsPerCluster
```

3.3.2.18 sectorsPerFAT

```
uint16_t sectorsPerFAT
```

3.3.2.19 sectorsPerTrack

```
uint16_t sectorsPerTrack
```

3.3.2.20 serialNumber

```
uint32_t serialNumber
```

3.3.2.21 volumeName

```
uint8_t volumeName[11]
```

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

- C:/Users/trish/Desktop/Code/School/CSC430/FileManager/include/FileManagerLibrary.h

3.4 FAT16Table Struct Reference

`#include <FileManagerLibrary.h>`

Data Fields

- uint16_t entry [32768]

3.4.1 Detailed Description

Defines the structure of a FAT16 table.

3.4.2 Field Documentation

3.4.2.1 entry

`uint16_t entry[32768]`

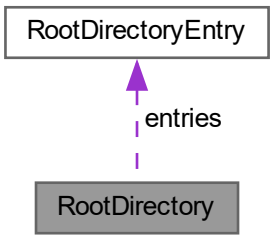
The documentation for this struct was generated from the following file:

- C:/Users/trish/Desktop/Code/School/CSC430/FileManager/include/FileManagerLibrary.h

3.5 RootDirectory Struct Reference

`#include <FileManagerLibrary.h>`

Collaboration diagram for RootDirectory:



Data Fields

- RootDirectoryEntry entries [512]
- int count

3.5.1 Field Documentation

3.5.1.1 count

```
int count
```

3.5.1.2 entries

```
RootDirectoryEntry entries[512]
```

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

- C:/Users/trish/Desktop/Code/School/CSC430/FileManager/include/FileManagerLibrary.h

3.6 RootDirectoryEntry Struct Reference

```
#include <FileManagerLibrary.h>
```

Data Fields

- uint8_t filename [8]
- uint8_t ext [3]
- uint8_t attributes
- uint16_t reserved1
- uint16_t creationTime
- uint16_t creationDate
- uint16_t lastAccessDate
- uint16_t reserved2
- uint16_t modifiedTime
- uint16_t modifiedDate
- uint16_t startingCluster
- uint32_t fileSize

3.6.1 Detailed Description

Defines the structure of a root directory entry.

3.6.2 Field Documentation

3.6.2.1 attributes

```
uint8_t attributes
```

3.6.2.2 creationDate

```
uint16_t creationDate
```

3.6.2.3 creationTime

```
uint16_t creationTime
```

3.6.2.4 ext

```
uint8_t ext[3]
```

3.6.2.5 filename

```
uint8_t filename[8]
```

3.6.2.6 fileSize

```
uint32_t fileSize
```

3.6.2.7 lastAccessDate

```
uint16_t lastAccessDate
```

3.6.2.8 modifiedDate

```
uint16_t modifiedDate
```

3.6.2.9 modifiedTime

```
uint16_t modifiedTime
```

3.6.2.10 reserved1

```
uint16_t reserved1
```

3.6.2.11 reserved2

```
uint16_t reserved2
```

3.6.2.12 startingCluster

```
uint16_t startingCluster
```

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

- C:/Users/trish/Desktop/Code/School/CSC430/FileManager/include/FileManagerLibrary.h

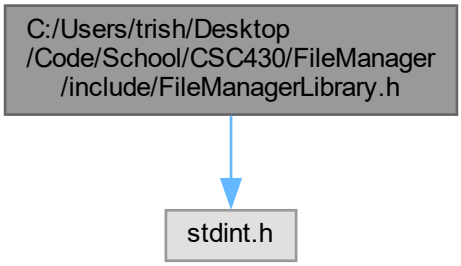
Chapter 4

File Documentation

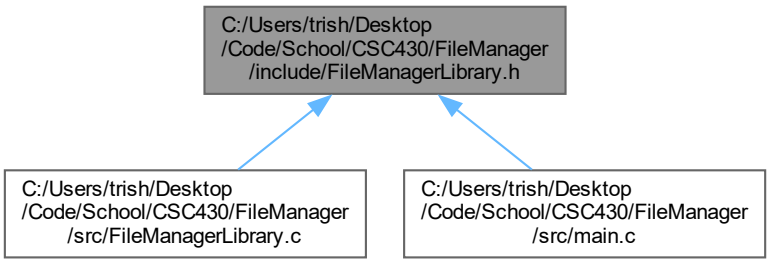
4.1 C:/Users/trish/Desktop/Code/School/CSC430/FileManager/include/↩ FileManagerLibrary.h File Reference

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

Include dependency graph for FileManagerLibrary.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct _partition
- struct BootSector
- struct _MBR
- struct FAT16Table
- struct RootDirectoryEntry
- struct RootDirectory

Typedefs

- typedef struct RootDirectory ROOTDIRECTORY

Functions

- struct _partition __attribute__ ((packed)) partition
- int ReadMBR (const char *filename)
- int ParseUSB (const char *filename)
- void dumpMBR ()
- void dumpBootSector ()
- void printData ()
- void readPartitions ()
- int readLBA (uint32_t offset)
- int readFatTables (uint32_t offset)
- int ReadDir (uint32_t offset)
- int ChangeDirectory (const char *dirName)
- void backToRootDir ()
- void dumpDir ()
- void readFile (char *filename)
- void dumpFAT16Table ()

Variables

- uint8_t bootable
- uint8_t first_chs [3]
- uint8_t type
- uint8_t last_chs [3]
- uint32_t lba_offset
- uint32_t sector_count
- uint8_t jumpCode [3]
- uint8_t oemName [8]
- uint16_t bytesPerSector
- uint8_t sectorsPerCluster
- uint16_t reservedSectors
- uint8_t numCopiesOfFAT
- uint16_t maxRootDirEntries
- uint16_t numSectorsSmall
- uint8_t mediaDescriptor
- uint16_t sectorsPerFAT
- uint16_t sectorsPerTrack
- uint16_t numHeads
- uint32_t numHiddenSectors

- uint32_t numSectorsLarge
- uint16_t logicalDriveNumber
- uint8_t extendedSignature
- uint32_t serialNumber
- uint8_t volumeName [11]
- uint8_t fatName [8]
- uint8_t executableCode [448]
- uint16_t executableMarker
- uint8_t bootcode [446]
- partition part [4]
- uint16_t signature
- uint16_t entry [32768]
- uint8_t filename [8]
- uint8_t ext [3]
- uint8_t attributes
- uint16_t reserved1
- uint16_t creationTime
- uint16_t creationDate
- uint16_t lastAccessDate
- uint16_t reserved2
- uint16_t modifiedTime
- uint16_t modifiedDate
- uint16_t startingCluster
- uint32_t fileSize

4.1.1 Typedef Documentation

4.1.1.1 ROOTDIRECTORY

```
typedef struct RootDirectory ROOTDIRECTORY
```

4.1.2 Function Documentation

4.1.2.1 __attribute__()

```
struct RootDirectoryEntry __attribute__ (
            (packed)  )
```

Defines the structure of a partition in the MBR.

Defines the structure of a boot sector.

Defines the structure of the MBR.

Defines the structure of a FAT16 table.

Defines the structure of a root directory entry.

4.1.2.2 backToRootDir()

```
void backToRootDir ( )
```

Changes the current working directory back to the root directory.

4.1.2.3 ChangeDirectory()

```
int ChangeDirectory (
            const char * dirName )
```

Changes the current working directory to the specified directory.

Parameters

dirName	the name of the directory to change to.

Returns

0 if the directory was successfully changed, otherwise returns a negative integer.

4.1.2.4 dumpBootSector()

```
void dumpBootSector ( )
```

Prints the contents of the boot sector to the console.

4.1.2.5 dumpDir()

```
void dumpDir ( )
```

Prints the contents of the current working directory to the console.

4.1.2.6 dumpFAT16Table()

```
void dumpFAT16Table ( )
```

Prints the contents of the File Allocation Table (FAT) to the console.

4.1.2.7 dumpMBR()

```
void dumpMBR ( )
```

Prints the contents of the MBR to the console.

4.1.2.8 ParseUSB()

```
int ParseUSB (
            const char * filename )
```

Parses the USB device specified by the filename and populates the necessary data structures.

Parameters

filename	the name of the USB device to parse.

Returns

0 if the USB device was successfully parsed, otherwise returns a negative integer.

4.1.2.9 printData()

```
void printData ( )
```

Prints the parsed data from the USB device to the console.

4.1.2.10 ReadDir()

```
int ReadDir (
            uint32_t offset )
```

Parses a directory given the offset of the directory

Parameters

offset	location of the Directory

Returns

EXIT_SUCCESS OR EXIT_FAILURE

4.1.2.11 readFatTables()

```
int readFatTables (
            uint32_t offset )
```

Parses the fat tables via offset recieved from the LBA

Parameters

offset	location of the Fat Tables

Returns

EXIT_SUCCESS OR EXIT_FAILURE

4.1.2.12 readFile()

```
void readFile (
            char * filename )
```

Reads and prints the contents of the specified file to the console.

Parameters

filename	the name of the file to read and print.

4.1.2.13 readLBA()

```
int readLBA (
            uint32_t offset )
```

Reads the LBA recieved from the Partitions.

Parameters

offset	location of the LBA

Returns

EXIT_SUCCESS OR EXIT_FAILURE

4.1.2.14 ReadMBR()

```
int ReadMBR (
            const char * filename )
```

Reads the Master Boot Record (MBR) from the specified file.

Parameters

filename	the name of the file to read the MBR from.

Returns

0 if the MBR was successfully read, otherwise returns a negative integer.

4.1.2.15 readPartitions()

```
void readPartitions ( )
```

Reads the partitions in the MBR

4.1.3 Variable Documentation

4.1.3.1 attributes

```
uint8_t attributes
```

4.1.3.2 bootable

```
uint8_t bootable
```

4.1.3.3 bootcode

```
uint8_t bootcode[446]
```

4.1.3.4 bytesPerSector

```
uint16_t bytesPerSector
```

4.1.3.5 creationDate

```
uint16_t creationDate
```

4.1.3.6 creationTime

```
uint16_t creationTime
```

4.1.3.7 entry

```
uint16_t entry[32768]
```

4.1.3.8 executableCode

```
uint8_t executableCode[448]
```

4.1.3.9 executableMarker

```
uint16_t executableMarker
```

4.1.3.10 ext

```
uint8_t ext[3]
```

4.1.3.11 extendedSignature

```
uint8_t extendedSignature
```

4.1.3.12 fatName

```
uint8_t fatName[8]
```

4.1.3.13 filename

```
uint8_t filename[8]
```

4.1.3.14 fileSize

```
uint32_t fileSize
```

4.1.3.15 first_chs

```
uint8_t first_chs[3]
```

4.1.3.16 jumpCode

```
uint8_t jumpCode[3]
```

4.1.3.17 last_chs

```
uint8_t last_chs[3]
```

4.1.3.18 lastAccessDate

```
uint16_t lastAccessDate
```

4.1.3.19 lba_offset

```
uint32_t lba_offset
```

4.1.3.20 logicalDriveNumber

```
uint16_t logicalDriveNumber
```

4.1.3.21 maxRootDirEntries

```
uint16_t maxRootDirEntries
```

4.1.3.22 mediaDescriptor

```
uint8_t mediaDescriptor
```

4.1.3.23 modifiedDate

```
uint16_t modifiedDate
```

4.1.3.24 modifiedTime

```
uint16_t modifiedTime
```

4.1.3.25 numCopiesOfFAT

```
uint8_t numCopiesOfFAT
```

4.1.3.26 numHeads

```
uint16_t numHeads
```

4.1.3.27 numHiddenSectors

```
uint32_t numHiddenSectors
```

4.1.3.28 numSectorsLarge

```
uint32_t numSectorsLarge
```

4.1.3.29 numSectorsSmall

```
uint16_t numSectorsSmall
```

4.1.3.30 oemName

```
uint8_t oemName[8]
```

4.1.3.31 part

```
partition part[4]
```

4.1.3.32 reserved1

```
uint16_t reserved1
```

4.1.3.33 reserved2

```
uint16_t reserved2
```

4.1.3.34 reservedSectors

```
uint16_t reservedSectors
```

4.1.3.35 sector_count

```
uint32_t sector_count
```

4.1.3.36 sectorsPerCluster

```
uint8_t sectorsPerCluster
```

4.1.3.37 sectorsPerFAT

```
uint16_t sectorsPerFAT
```

4.1.3.38 sectorsPerTrack

```
uint16_t sectorsPerTrack
```

4.1.3.39 serialNumber

```
uint32_t serialNumber
```

4.1.3.40 signature

```
uint16_t signature
```

4.1.3.41 startingCluster

```
uint16_t startingCluster
```

4.1.3.42 type

```
uint8_t type
```

4.1.3.43 volumeName

```
uint8_t volumeName[11]
```

4.2 FileManagerLibrary.h

Go to the documentation of this file.

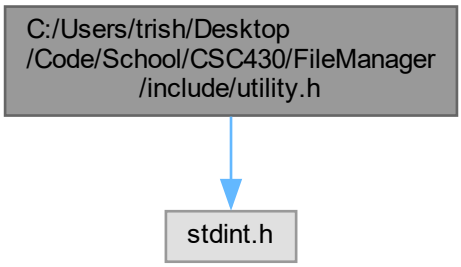
```
00001 #ifndef FILESYSTEMMANAGER_FILEMANAGERLIBRARY_H
00002 #define FILESYSTEMMANAGER_FILEMANAGERLIBRARY_H
00003
00004 #include <stdint.h>
00005
00009 typedef struct _partition {
00010     uint8_t bootable;
00011     uint8_t first_chs[3];
00012     uint8_t type;
00013     uint8_t last_chs[3];
00014     uint32_t lba_offset;
00015     uint32_t sector_count;
00016 } __attribute__((packed)) partition;
00017
00021 typedef struct BootSector {
00022     uint8_t jumpCode[3];
00023     uint8_t oemName[8];
00024     uint16_t bytesPerSector;
00025     uint8_t sectorsPerCluster;
00026     uint16_t reservedSectors;
00027     uint8_t numCopiesOfFAT;
00028     uint16_t maxRootDirEntries;
00029     uint16_t numSectorsSmall;
00030     uint8_t mediaDescriptor;
00031     uint16_t sectorsPerFAT;
00032     uint16_t sectorsPerTrack;
```

```
00033     uint16_t numHeads;
00034     uint32_t numHiddenSectors;
00035     uint32_t numSectorsLarge;
00036     uint16_t logicalDriveNumber;
00037     uint8_t extendedSignature;
00038     uint32_t serialNumber;
00039     uint8_t volumeName[11];
00040     uint8_t fatName[8];
00041     uint8_t executableCode[448];
00042     uint16_t executableMarker;
00043 } __attribute__((packed)) BootSector;
00044
00048 typedef struct _MBR {
00049     uint8_t bootcode[446];
00050     partition part[4];
00051     uint16_t signature;
00052 } __attribute__((packed)) MBR;
00053
00057 typedef struct FAT16Table {
00058     uint16_t entry[32768];
00059 } __attribute__((packed)) FAT16Table;
00060
00064 typedef struct RootDirectoryEntry {
00065     uint8_t filename[8];
00066     uint8_t ext[3];
00067     uint8_t attributes;
00068     uint16_t reserved1;
00069     uint16_t creationTime;
00070     uint16_t creationDate;
00071     uint16_t lastAccessDate;
00072     uint16_t reserved2;
00073     uint16_t modifiedTime;
00074     uint16_t modifiedDate;
00075     uint16_t startingCluster;
00076     uint32_t fileSize;
00077 } __attribute__((packed)) RootDirectoryEntry;
00078
00079 typedef struct RootDirectory {
00080     RootDirectoryEntry entries[512];
00081     int count;
00082 } ROOTDIRECTORY;
00083
00090 int ReadMBR(const char* filename);
00091
00098 int ParseUSB(const char* filename);
00099
00103 void dumpMBR();
00104
00108 void dumpBootSector();
00109
00113 void printData();
00114
00118 void readPartitions();
00119
00125 int readLBA(uint32_t offset);
00126
00132 int readFatTables(uint32_t offset);
00133
00139 int ReadDir(uint32_t offset);
00140
00147 int ChangeDirectory(const char* dirName);
00148
00152 void backToRootDir();
00153
00157 void dumpDir();
00158
00164 void readFile(char *filename);
00165
00169 void dumpFAT16Table();
00170
00171 #endif
00172
```

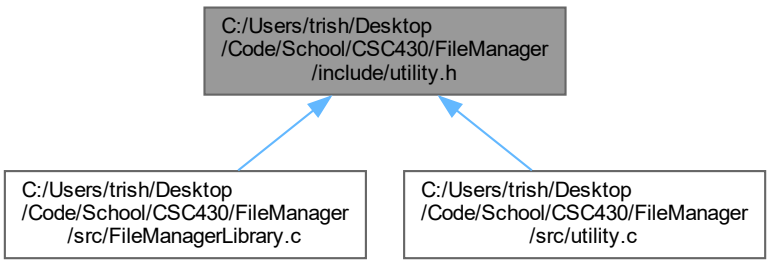
4.3 C:/Users/trish/Desktop/Code/School/CSC430/File↩ Manager/include/utility.h File Reference

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

Include dependency graph for utility.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define FILE_ATTRIBUTE_READONLY 0x01
- #define FILE_ATTRIBUTE_HIDDEN 0x02
- #define FILE_ATTRIBUTE_SYSTEM 0x04
- #define FILE_ATTRIBUTE_VOLUME 0x08
- #define FILE_ATTRIBUTE_LFN
- #define FILE_ATTRIBUTE_DIRECTORY 0x10
- #define FILE_ATTRIBUTE_ARCHIVE 0x20

Functions

- const char * PartitionTypeString (uint8_t type_code)
- const char * FileAttributeString (uint8_t attribute)
- const char * MediaTypeString (int media_type)
- const char * HumanNumberString (uint64_t size)
- const char * EightDotThreeString (const uint8_t name[8], const uint8_t ext[3])
- const char * GetPathSeparator (const char *path)
- void PrintHexDump (const char *buffer, int count)

4.3.1 Macro Definition Documentation

4.3.1.1 FILE_ATTRIBUTE_ARCHIVE

```
#define FILE_ATTRIBUTE_ARCHIVE 0x20
```

4.3.1.2 FILE_ATTRIBUTE_DIRECTORY

```
#define FILE_ATTRIBUTE_DIRECTORY 0x10
```

4.3.1.3 FILE_ATTRIBUTE_HIDDEN

```
#define FILE_ATTRIBUTE_HIDDEN 0x02
```

4.3.1.4 FILE_ATTRIBUTE_LFN

```
#define FILE_ATTRIBUTE_LFN
```

Value:

```
    (FILE_ATTRIBUTE_READONLY | FILE_ATTRIBUTE_HIDDEN | \
     FILE_ATTRIBUTE_SYSTEM | FILE_ATTRIBUTE_VOLUME)
```

4.3.1.5 FILE_ATTRIBUTE_READONLY

```
#define FILE_ATTRIBUTE_READONLY 0x01
```

4.3.1.6 FILE_ATTRIBUTE_SYSTEM

```
#define FILE_ATTRIBUTE_SYSTEM 0x04
```

4.3.1.7 FILE_ATTRIBUTE_VOLUME

```
#define FILE_ATTRIBUTE_VOLUME 0x08
```

4.3.2 Function Documentation

4.3.2.1 EightDotThreeString()

```
const char * EightDotThreeString (
            const uint8_t name[8],
            const uint8_t ext[3] )
```

4.3.2.2 FileAttributeString()

```
const char * FileAttributeString (
            uint8_t attribute )
```

4.3.2.3 GetPathSeparator()

```
const char * GetPathSeparator (
            const char * path )
```

4.3.2.4 HumanNumberString()

```
const char * HumanNumberString (
            uint64_t size )
```

4.3.2.5 MediaTypeString()

```
const char * MediaTypeString (
            int media_type )
```

4.3.2.6 PartitionTypeString()

```
const char * PartitionTypeString (
            uint8_t type_code )
```

4.3.2.7 PrintHexDump()

```
void PrintHexDump (
            const char * buffer,
            int count )
```

4.4 utility.h

Go to the documentation of this file.

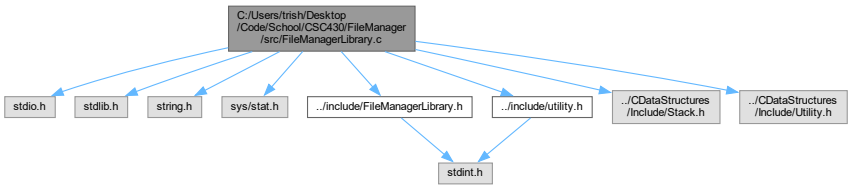
```
00001 #pragma once
00002
00003 #include <stdint.h>
00004
00005
00006 #define FILE_ATTRIBUTE_READONLY 0x01
00007 #define FILE_ATTRIBUTE_HIDDEN 0x02
00008 #define FILE_ATTRIBUTE_SYSTEM 0x04
00009 #define FILE_ATTRIBUTE_VOLUME 0x08
00010 #define FILE_ATTRIBUTE_LFN \
00011     (FILE_ATTRIBUTE_READONLY | FILE_ATTRIBUTE_HIDDEN | \
00012      FILE_ATTRIBUTE_SYSTEM | FILE_ATTRIBUTE_VOLUME)
00013 #define FILE_ATTRIBUTE_DIRECTORY 0x10
00014 #define FILE_ATTRIBUTE_ARCHIVE 0x20
00015
00016 const char* PartitionTypeString(uint8_t type_code);
00017 const char* FileAttributeString(uint8_t attribute);
00018 const char* MediaTypeString(int media_type);
00019 const char* HumanNumberString(uint64_t size);
00020 const char* EightDotThreeString(const uint8_t name[8], const uint8_t ext[3]);
00021 const char* GetPathSeparator(const char* path);
00022
00023 void PrintHexDump(const char* buffer, int count);
```

4.5 C:/Users/trish/Desktop/Code/School/CSC430/FileManager/src/FileManagerLibrary.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>
#include "../include/FileManagerLibrary.h"
#include "../include/utility.h"
```

```
#include "../CDataStructures/Include/Stack.h"
#include "../CDataStructures/Include/Utility.h"
```

Include dependency graph for FileManagerLibrary.c:



Macros

- #define DIRECTORY_ATTRIBUTE_SUBDIRECTORY 0x10

Functions

- void printData ()
- int ParseUSB (const char ∗filename)
- int ReadMBR (const char ∗filename)
- void readPartitions ()
- int readLBA (uint32_t offset)
- int readFatTables (uint32_t offset)
- int ReadDir (uint32_t offset)
- uint32_t getOffset (char ∗subDirectoryName)
- void changeDirectory (char ∗subDirectoryName)
- void backToRootDir ()
- void dumpDir ()
- void dumpMBR ()
- void dumpBootSector ()
- void dumpFAT16Table ()
- void readFile (char ∗filename)

4.5.1 Macro Definition Documentation

4.5.1.1 DIRECTORY_ATTRIBUTE_SUBDIRECTORY

```
#define DIRECTORY_ATTRIBUTE_SUBDIRECTORY 0x10
```

4.5.2 Function Documentation

4.5.2.1 backToRootDir()

```
void backToRootDir ( )
```

Changes the current working directory back to the root directory.

4.5.2.2 changeDirectory()

```
void changeDirectory (
            char * subDirectoryName )
```

4.5.2.3 dumpBootSector()

```
void dumpBootSector ( )
```

Prints the contents of the boot sector to the console.

4.5.2.4 dumpDir()

```
void dumpDir ( )
```

Prints the contents of the current working directory to the console.

4.5.2.5 dumpFAT16Table()

```
void dumpFAT16Table ( )
```

Prints the contents of the File Allocation Table (FAT) to the console.

4.5.2.6 dumpMBR()

```
void dumpMBR ( )
```

Prints the contents of the MBR to the console.

4.5.2.7 getOffset()

```
uint32_t getOffset (
            char * subDirectoryName )
```

4.5.2.8 ParseUSB()

```
int ParseUSB (
            const char * filename )
```

Parses the USB device specified by the filename and populates the necessary data structures.

Parameters

filename	the name of the USB device to parse.

Returns

0 if the USB device was successfully parsed, otherwise returns a negative integer.

4.5.2.9 printData()

```
void printData ( )
```

Prints the parsed data from the USB device to the console.

4.5.2.10 ReadDir()

```
int ReadDir (
            uint32_t offset )
```

Parses a directory given the offset of the directory

Parameters

offset	location of the Directory

Returns

EXIT_SUCCESS OR EXIT_FAILURE

4.5.2.11 readFatTables()

```
int readFatTables (
            uint32_t offset )
```

Parses the fat tables via offset recieved from the LBA

Parameters

offset	location of the Fat Tables

Returns

EXIT_SUCCESS OR EXIT_FAILURE

4.5.2.12 readFile()

```
void readFile (
            char * filename )
```

Reads and prints the contents of the specified file to the console.

Parameters

filename	the name of the file to read and print.

4.5.2.13 readLBA()

```
int readLBA (
            uint32_t offset )
```

Reads the LBA recieved from the Partitions.

Parameters

offset	location of the LBA

Returns

EXIT_SUCCESS OR EXIT_FAILURE

4.5.2.14 ReadMBR()

```
int ReadMBR (
            const char * filename )
```

Reads the Master Boot Record (MBR) from the specified file.

Parameters

filename	the name of the file to read the MBR from.

Returns

0 if the MBR was successfully read, otherwise returns a negative integer.

4.5.2.15 readPartitions()

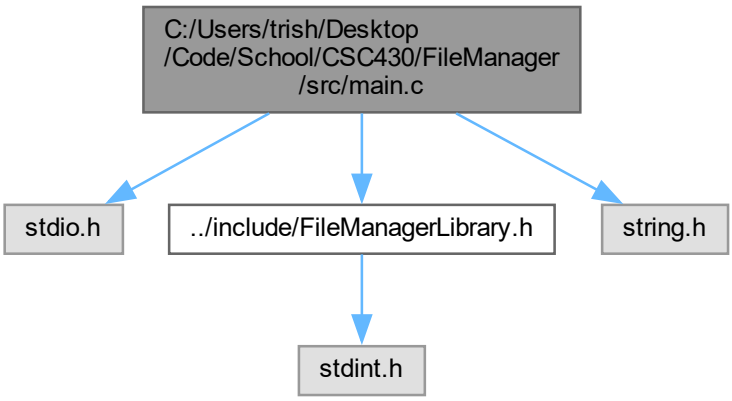
```
void readPartitions ( )
```

Reads the partitions in the MBR

4.6 C:/Users/trish/Desktop/Code/School/CSC430/FileManager/src/main.c File Reference

```
#include <stdio.h>
#include "../include/FileManagerLibrary.h"
#include <string.h>
```

Include dependency graph for main.c:



Functions

- void print_help ()
- int main ()

4.6.1 Function Documentation

4.6.1.1 main()

```
int main ( )
```

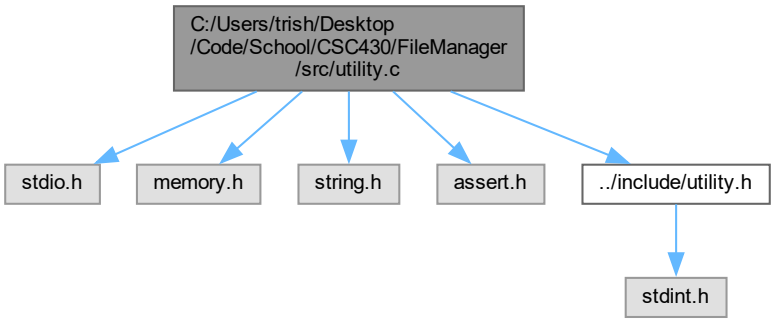
4.6.1.2 print_help()

```
void print_help ( )
```

4.7 C:/Users/trish/Desktop/Code/School/CSC430/File↩ Manager/src/utility.c File Reference

```
#include <stdio.h>
#include <memory.h>
#include <string.h>
#include <assert.h>
#include "../include/utility.h"
```

Include dependency graph for utility.c:



Functions

- const char * PartitionTypeString (uint8_t type_code)
- const char * MediaTypeString (int media_type)
- const char * HumanNumberString (uint64_t size)
- const char * FileAttributeString (uint8_t attribute)
- const char * EightDotThreeString (const uint8_t name[8], const uint8_t ext[3])
- const char * GetPathSeparator (const char *path)
- void PrintHexDump (const char *buffer, int count)

4.7.1 Function Documentation

4.7.1.1 EightDotThreeString()

```
const char * EightDotThreeString (
            const uint8_t name[8],
            const uint8_t ext[3] )
```

4.7.1.2 FileAttributeString()

```
const char * FileAttributeString (
            uint8_t attribute )
```

4.7.1.3 GetPathSeparator()

```
const char * GetPathSeparator (
            const char * path )
```

4.7.1.4 HumanNumberString()

```
const char * HumanNumberString (
            uint64_t size )
```

4.7.1.5 MediaTypeString()

```
const char * MediaTypeString (
            int media_type )
```

4.7.1.6 PartitionTypeString()

```
const char * PartitionTypeString (
            uint8_t type_code )
```

4.7.1.7 PrintHexDump()

```
void PrintHexDump (
            const char * buffer,
            int count )
```

Index