File Management System

Generated by Doxygen 1.8.7

Mon Jun 16 2014 16:18:38

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 Fms Namespace Reference

Classes

- struct Dat

  Defined structure for Disk DAT.
- struct Dir

  Defined structure for Directory / Sub-Directory.
- class Disk

  Represent a disk in memory.
- class Dms

  Disk Management System.
- class Fcb

  File control block.
- struct FileHeader

  Defined structure for file header.
- struct Sector

  Represent a Sector on the HDD.
- struct VolumeHeader

  Defined structure for Volume Header.

Typedefs

- typedef std::bitset < CLUSTERSINDISK > DatType

  DAT / FAT bitset for marking which clusters are free / taken.
- typedef uint32_t FileSize

  Type for file size in bytes.
- typedef uint32_t SectorId

  Type for Sector ID.
- typedef uint32_t ClusterId

  Type for Cluster ID.
- typedef std::array< Sector, CLUSTERSIZE > Cluster

  Type of cluster (array of sectors)
- typedef uint32_t RecordId

  Type for Record ID.

Functions

- std::ostream & operator<< (std::ostream &out, const Dat &dat)
- std::ostream & operator<< (std::ostream &out, const Dir &dir)
- std::ostream & operator<< (std::ostream &out, const Disk &disk)
- std::string getDate ()

  Receieve the current date in DDMMYYYY format.
- std::wstring stringToWString (const std::string &s)

  Convert string to wstring.
- std::string convertDate (const char date[])

  Convert date from DDMMYY format to something readable.
- const std::string removePath (const std::string &fileName)

  Remove path from full path.
- std::ostream & operator<< (std::ostream &out, const Sector &sector)

Variables

- const ClusterId CLUSTERSINDISK = 1600

  Amount of clusters in the disk.
- const SectorId ENTRIESPERSECTOR = 14

  Amount of directory entries per sector.
- const ClusterId CLUSTERSRESERVED =4

  Amount of clusters used for HDD structure.
- const SectorId SIZEOFSECTOR = 1024

  Bytes that can be stored in a single sector.
- const SectorId CLUSTERSIZE = 2

  Amount of sectors in cluster.

5.1.1 Typedef Documentation

5.1.1.1 typedef std::array<Sector, CLUSTERSIZE> Fms::Cluster

Type of cluster (array of sectors)

5.1.1.2 typedef uint32_t Fms::ClusterId

Type for Cluster ID.

5.1.1.3 typedef std::bitset<CLUSTERSINDISK> Fms::DatType

DAT / FAT bitset for marking which clusters are free / taken.

5.1.1.4 typedef uint32_t Fms::FileSize

Type for file size in bytes.

5.1.1.5 typedef uint32_t Fms::RecordId

Type for Record ID.

5.1.1.6 typedef uint32_t Fms::SectorId

Type for Sector ID.

5.1.2 Function Documentation

5.1.2.1 FMS_API std::string Fms::convertDate (const char *date[]*)

Convert date from DDMMYY format to something readable.

Parameters

date	Char array of size 10 in format DDMMYYYY

Returns

String of DD/MM/YYYY

5.1.2.2 FMS_API std::string Fms::getDate ( )

Receieve the current date in DDMMYYYY format.

Returns

Date in DDMMYYYY format

5.1.2.3 std::ostream& Fms::operator<< (std::ostream & *out*, const Dat & *dat*)

Parameters

out	Stream object
dat	The Dat to print

Returns

The stream object

5.1.2.4 std::ostream& Fms::operator<< (std::ostream & *out*, const Sector & *sector*)

Parameters

out	Stream object
sector	The sector to print

Returns

The stream object

5.1.2.5 std::ostream& Fms::operator<< (std::ostream & *out*, const Dir & *dir*)

Parameters

out	Stream object
dir	The Dir to print

Returns

The stream object

5.1.2.6 std::ostream& Fms::operator<< (std::ostream & *out*, const Disk & *disk*)

Parameters

out	Stream object
disk	The disk to print

Returns

The stream object

5.1.2.7 const std::string Fms::removePath (const std::string & *fileName*)

Remove path from full path.

C:\path\to\file.bin -> file.bin

Parameters

fileName	Full path or file name

Returns

File name

5.1.2.8 FMS_API std::wstring Fms::stringToWString (const std::string & *s*)

Convert string to wstring.

Parameters

s	String to convert

Returns

The string as wstring

5.1.3 Variable Documentation

5.1.3.1 const ClusterId Fms::CLUSTERSINDISK = 1600

Amount of clusters in the disk.

5.1.3.2 const SectorId Fms::CLUSTERSIZE = 2

Amount of sectors in cluster.

5.1.3.3 const ClusterId Fms::CLUSTERSRESERVED =4

Amount of clusters used for HDD structure.

5.1.3.4 const SectorId Fms::ENTRIESPERSECTOR = 14

Amount of directory entries per sector.

This is strictly defined by the assignment. We would have used math to calculate it, but it seems that according to our math it should be 15. None the less, it could be replaced with: SECTORSIZE / sizeof(DirEntry) for more general behaviour.

5.1.3.5 const SectorId Fms::SIZEOFSECTOR = 1024

Bytes that can be stored in a single sector.

5.2 FmsUtils Namespace Reference

Classes

- struct Student

  Student item.
- struct StudentRecord

  Student record for saving to disk.

Functions

- std::ostream & operator<< (std::ostream &o, const Student &s)
- std::istream & operator>> (std::istream &i, Student &s)

5.2.1 Function Documentation

5.2.1.1 std::ostream& FmsUtils::operator<< (std::ostream & *o*, const Student & *s*)

5.2.1.2 std::istream& FmsUtils::operator>> (std::istream & *i*, Student & *s*)

Parameters

i	Stream object
s	The student to print

Returns

The stream object

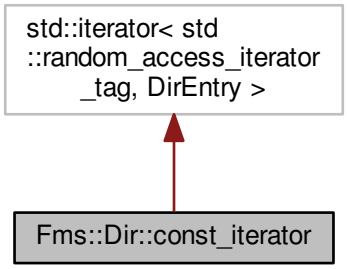
Chapter 6

Class Documentation

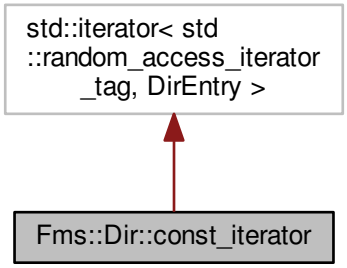
6.1 Fms::Dir::const_iterator Class Reference

```
#include <Dir.h>
```

Inheritance diagram for Fms::Dir::const_iterator:



Collaboration diagram for Fms::Dir::const_iterator:



Public Member Functions

- FMS_API const_iterator (const Dir *p, SectorId loc=0)
- FMS_API const_iterator (const iterator &)
- FMS_API bool operator== (const const_iterator &) const
- FMS_API bool operator!= (const const_iterator &) const
- FMS_API bool operator> (const const_iterator &) const
- FMS_API bool operator< (const const_iterator &) const
- FMS_API bool operator>= (const const_iterator &) const
- FMS_API bool operator<= (const const_iterator &) const
- FMS_API const DirEntry & operator* ()
- FMS_API const DirEntry * operator-> ()
- FMS_API const DirEntry & operator[] (int)
- FMS_API const_iterator operator++ ()
- FMS_API const_iterator operator++ (int)
- FMS_API const_iterator operator-- ()
- FMS_API const_iterator operator-- (int)
- FMS_API const_iterator operator+ (const int &) const
- FMS_API const_iterator operator- (const int &) const
- FMS_API const_iterator operator+= (const int &)
- FMS_API const_iterator operator-= (const int &)
- FMS_API int operator- (const const_iterator &) const

6.1.1 Constructor & Destructor Documentation

6.1.1.1 Fms::Dir::const_iterator::const_iterator (const Dir * *p*, SectorId *loc* = 0)

6.1.1.2 Fms::Dir::const_iterator::const_iterator (const iterator & *it*)

6.1.2 Member Function Documentation

6.1.2.1 bool Fms::Dir::const_iterator::operator!= (const const_iterator & *it*) const

6.1.2.2 const Dir::DirEntry & Fms::Dir::const_iterator::operator* ( )

6.1.2.3 Dir::const_iterator Fms::Dir::const_iterator::operator+ (const int & *n*) const

6.1.2.4 Dir::const_iterator Fms::Dir::const_iterator::operator++ ( )

6.1.2.5 Dir::const_iterator Fms::Dir::const_iterator::operator++ (int)

6.1.2.6 Dir::const_iterator Fms::Dir::const_iterator::operator+= (const int & *n*)

6.1.2.7 Dir::const_iterator Fms::Dir::const_iterator::operator- (const int & *n*) const

6.1.2.8 int Fms::Dir::const_iterator::operator- (const const_iterator & *it*) const

6.1.2.9 Dir::const_iterator Fms::Dir::const_iterator::operator-- ( )

6.1.2.10 Dir::const_iterator Fms::Dir::const_iterator::operator-- (int)

6.1.2.11 Dir::const_iterator Fms::Dir::const_iterator::operator-= (const int & *n*)

6.1.2.12 const Dir::DirEntry * Fms::Dir::const_iterator::operator-> ( )

6.1.2.13 bool Fms::Dir::const_iterator::operator< (const const_iterator & *it*) const

6.1.2.14 bool Fms::Dir::const_iterator::operator<= (const const_iterator & *it*) const

6.1.2.15 bool Fms::Dir::const_iterator::operator== (const const_iterator & *it*) const

6.1.2.16 bool Fms::Dir::const_iterator::operator> (const const_iterator & *it*) const

6.1.2.17 bool Fms::Dir::const_iterator::operator>= (const const_iterator & *it*) const

6.1.2.18 const Dir::DirEntry & Fms::Dir::const_iterator::operator[] (int)

The documentation for this class was generated from the following files:

- Dir.h
- Dir.cpp

6.2 Fms::Dat Struct Reference

Defined structure for Disk DAT.

```
#include <Dat.h>
```

Public Member Functions

- FMS_API Dat ()
  Default constructor.

Public Attributes

- SectorId sectorNr
  Sector where this item is located.
- DatType dat
  Mark which clusters are free / taken.
- char emptyArea [SIZEOFSECTOR-sizeof(DatType)-sizeof(ClusterId)]
  Empty area in order for item to be the same size as sector.

Friends

- FMS_API friend std::ostream & operator<< (std::ostream &, const Dat &)
  Pipe Dat object to ostream.

6.2.1 Detailed Description

Defined structure for Disk DAT.

The DAT should use a single sector to store its data. This data contains information about the disk.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Fms::Dat::Dat ( )

Default constructor.

6.2.3 Friends And Related Function Documentation

6.2.3.1 FMS_API friend std::ostream& operator<< (std::ostream & *out*, const Dat & *dat*) [friend]

Pipe Dat object to ostream.

Parameters

out	Stream object
dat	The Dat to print

Returns

The stream object

6.2.4 Member Data Documentation

6.2.4.1 DatType Fms::Dat::dat

Mark which clusters are free / taken.

6.2.4.2 char Fms::Dat::emptyArea[SIZEOFSECTOR-sizeof(DatType)-sizeof(ClusterId)]

Empty area in order for item to be the same size as sector.

6.2.4.3 SectorId Fms::Dat::sectorNr

Sector where this item is located.

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

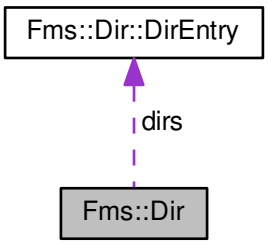
- Dat.h
- Dat.cpp

6.3 Fms::Dir Struct Reference

Defined structure for Directory / Sub-Directory.

`#include <Dir.h>`

Collaboration diagram for Fms::Dir:



Classes

- class const_iterator
- struct DirEntry

  Defined structure for Directory Entry.
- class iterator

Public Member Functions

- FMS_API Dir ()

  Default constructor.
- FMS_API Dir (const char *)

  Contruct a directory from char stream.
- FMS_API std::unique_ptr< char[]> sectorOutputAsCharStream () const

  Make valid char stream output for the directory.
- FMS_API Dir (std::vector< Sector >)

  Contruct a directory from a vector of sectors.
- FMS_API std::vector< Sector > sectorOutputAsVector () const

  Create a vector of sectors with the directory structure.
- FMS_API DirEntry & operator[] (SectorId i)

  Operator [] abstraction.
- FMS_API const DirEntry & operator[] (SectorId i) const

  Operator [] abstraction, const version.
- FMS_API bool operator== (const Dir &) const

  Operator ==.
- FMS_API bool operator!= (const Dir &) const

  Operator !=.
- FMS_API iterator begin ()

  Get begin iterator.
- FMS_API iterator end ()

  Get end iterator.
- FMS_API const_iterator begin () const

  Get begin const_iterator.
- FMS_API const_iterator end () const

  Get end const_iterator.
- FMS_API size_t size () const

  Return the amount of entries the directory can hold.
- FMS_API uint32_t activeEntries () const

  Returns the amount of active entries in the directory.

Public Attributes

- SectorId sectorNr

  Sector where this item is located.
- DirEntry dirs [CLUSTERSIZE][ENTRIESPERSECTOR]

  Array of all the directories / files pointed from this directory.

Friends

- FMS_API friend std::ostream & operator<< (std::ostream &, const Dir &)

  Pipe Dir object to ostream.

6.3.1 Detailed Description

Defined structure for Directory / Sub-Directory.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Fms::Dir::Dir ( )

Default constructor.

See also

Dir(const char ∗), Dir(std::vector<Sector>)

6.3.2.2 Fms::Dir::Dir (const char ∗ *input*)

Contruct a directory from char stream.

Size of input is CLUSTERSIZE ∗ SECTORSIZE.

See also

Dir(), Dir(std::vector<Sector>), sectorOutputAsCharStream()

6.3.2.3 Fms::Dir::Dir (std::vector< **Sector** > *sectors*)

Contruct a directory from a vector of sectors.

Sector count is CLUSTERSIZE.

See also

Dir(), Dir(const char ∗), sectorOutputAsVector()

6.3.3 Member Function Documentation

6.3.3.1 size_t Fms::Dir::activeEntries ( ) const

Returns the amount of active entries in the directory.

Returns

Amount of active entries in the directory.

6.3.3.2 **Dir::iterator** Fms::Dir::begin ( )

Get begin iterator.

Returns

Iterator to the start of the directory

6.3.3.3 Dir::const_iterator Fms::Dir::begin ( ) const

Get begin const_iterator.

Returns

const_iterator to the start of the directory

6.3.3.4 Dir::iterator Fms::Dir::end ( )

Get end iterator.

Returns

Iterator to the end of the directory

6.3.3.5 Dir::const_iterator Fms::Dir::end ( ) const

Get end const_iterator.

Returns

const_iterator to the end of the directory

6.3.3.6 bool Fms::Dir::operator!= (const Dir & *dir*) const

Operator !=.

Parameters

dir	Directory

Returns

this != dir

6.3.3.7 bool Fms::Dir::operator== (const Dir & *dir*) const

Operator ==.

Parameters

dir	Directory

Returns

this == dir

6.3.3.8 Dir::DirEntry & Fms::Dir::operator[] (SectorId *i*)

Operator [] abstraction.

Parameters

i	Index

Returns

Const reference to the entry

See also

operator[](SectorId) const

6.3.3.9 const Dir::DirEntry & Fms::Dir::operator[] (SectorId *i*) const

Operator [] abstraction, const version.

Parameters

i	Index

Returns

Reference to the entry

See also

operator[](SectorId)

6.3.3.10 std::unique_ptr< char[] > Fms::Dir::sectorOutputAsCharStream ( ) const

Make valid char stream output for the directory.

Size of output is CLUSTERSIZE ∗ SECTORSIZE.

See also

Dir(const char ∗)

6.3.3.11 std::vector< Sector > Fms::Dir::sectorOutputAsVector ( ) const

Create a vector of sectors with the directory structure.

Amount of sectors in the vector is CLUSTERSIZE

See also

Dir(std::vector<Sector>)

6.3.3.12 size_t Fms::Dir::size ( ) const

Return the amount of entries the directory can hold.

Returns

Amount of entries the directory can hold.

6.3.4 Friends And Related Function Documentation

6.3.4.1 FMS_API friend std::ostream& operator<< (std::ostream & *out*, const Dir & *dir*) [friend]

Pipe Dir object to ostream.

Parameters

out	Stream object
dir	The Dir to print

Returns

The stream object

6.3.5 Member Data Documentation

6.3.5.1 DirEntry Fms::Dir::dirs[CLUSTERSIZE][ENTRIESPERSECTOR]

Array of all the directories / files pointed from this directory.

6.3.5.2 SectorId Fms::Dir::sectorNr

Sector where this item is located.

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

- Dir.h
- Dir.cpp

6.4 Fms::Dir::DirEntry Struct Reference

Defined structure for Directory Entry.

```
#include <Dir.h>
```

Public Member Functions

- FMS_API DirEntry ()

  Default constructor.
- FMS_API bool operator== (const DirEntry &) const

  Operator ==.
- FMS_API bool operator!= (const DirEntry &) const

  Operator !=.

Public Attributes

- char fileName [12]

  Name of file.
- char fileOwner [12]

  Owner of file.
- SectorId fileAddr

  First sector used by the file.
- char crDate [10]

  Date of when the file was created.
- FileSize fileSize

  File size in bytes.
- RecordId eofRecNr

Location of EOF for the file.

- RecordId maxRecSize

  Maximum record size.

- RecordId actualRecSize

  Actual record size.

- char recFormat [2]

  Format of record.

- SectorId keyOffset

  Offset before key in a record.

- SectorId keySize

  Size in bytes of key.

- char keyType [2]

  Type of Key.

- uint8_t entryStatus

  Status of entry. 0 - Wasn't used since last format. 1 - Active, file exist. 2 - Inactive, deleted file.

6.4.1 Detailed Description

Defined structure for Directory Entry.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 Fms::Dir::DirEntry::DirEntry ( )

Default constructor.

6.4.3 Member Function Documentation

6.4.3.1 bool Fms::Dir::DirEntry::operator!= (const DirEntry & *entry*) const

Operator !=.

Parameters

entry	Directory Entry

Returns

this != dirEntry

6.4.3.2 bool Fms::Dir::DirEntry::operator== (const DirEntry & *entry*) const

Operator ==.

Parameters

entry	Directory Entry

Returns

this == dirEntry

6.4.4 Member Data Documentation

6.4.4.1 RecordId Fms::Dir::DirEntry::actualRecSize

Actual record size.

6.4.4.2 char Fms::Dir::DirEntry::crDate[10]

Date of when the file was created.

6.4.4.3 uint8_t Fms::Dir::DirEntry::entryStatus

Status of entry. 0 - Wasn't used since last format. 1 - Active, file exist. 2 - Inactive, deleted file.

6.4.4.4 RecordId Fms::Dir::DirEntry::eofRecNr

Location of EOF for the file.

6.4.4.5 SectorId Fms::Dir::DirEntry::fileAddr

First sector used by the file.

6.4.4.6 char Fms::Dir::DirEntry::fileName[12]

Name of file.

6.4.4.7 char Fms::Dir::DirEntry::fileOwner[12]

Owner of file.

6.4.4.8 FileSize Fms::Dir::DirEntry::fileSize

File size in bytes.

6.4.4.9 SectorId Fms::Dir::DirEntry::keyOffset

Offset before key in a record.

6.4.4.10 SectorId Fms::Dir::DirEntry::keySize

Size in bytes of key.

6.4.4.11 char Fms::Dir::DirEntry::keyType[2]

Type of Key.

I - int. F - Float. D - Double. C - Array of characters.

6.4.4.12 RecordId Fms::Dir::DirEntry::maxRecSize

Maximum record size.

6.4.4.13 char Fms::Dir::DirEntry::recFormat[2]

Format of record.

D - Directory. F - Fixed length file. V - Varried length file.

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

- Dir.h
- Dir.cpp

6.5 Fms::Disk Class Reference

Represent a disk in memory.

```
#include <Disk.h>
```

Classes

- class FatReader

  Helper class for reading FAT Table.

Public Member Functions

- FMS_API Disk ()

  Default constructor.
- FMS_API Disk (std::string &, std::string &, char)

  Construct using disk name.
- FMS_API ~Disk ()

  Destrcutor. Enforces safe unmount.
- FMS_API void createDisk (std::string &diskName, std::string &diskOwner)

  Create a new unformatted disk.
- FMS_API void mountDisk (std::string &diskName)

  Mount a disk.
- FMS_API void unmountDisk ()

  Unmount mounted disk.
- FMS_API void recreateDisk (std::string &diskOwner)

  Recreates the disk.
- std::fstream * getDskfl ()

  Get the pointer to the opened disk.
- FMS_API void seekToSector (SectorId)

  Move the sector needle to the requested sector.
- FMS_API void writeFreeSector (SectorId, Sector *)

  Write data to a free specified sector.
- FMS_API void writeSector (SectorId, Sector *)

  Write data to specified sector.
- FMS_API void writeSector (Sector *)

  Write data to sector.

- FMS_API void readSector (SectorId, Sector ∗)

  Read data from sector.
- FMS_API void readSector (Sector ∗)

  Read data from sector.
- FMS_API void format (std::string)

  Formats the disk.
- FMS_API ClusterId howMuchEmpty () const

  Returns the amount of free clusters in the disk.
- void dealloc (DatType &)

  Mark the sectors in given FAT as free.
- void alloc (DatType &, SectorId, uint32_t)

  Allocate sectors to file.
- void allocExtend (DatType &, SectorId, uint32_t)

  Allocate more sectors to file.
- FMS_API const Dat & getDat () const

  Return the disk dat.
- FMS_API const Dir & getDir () const

  Return current visible directory in disk.
- FMS_API const VolumeHeader & getVolumeHeader () const

  Return the VolumeHeader of the mounted disk.
- FMS_API void createFile (const std::string &fileName, const std::string &ownerName, const std::string &file↩Type, size_t recordLength, SectorId numSectors, const std::string &keyType, size_t keyOffset, uint32_t fit=1, size_t keyLength=1)

  Create a new file.
- FMS_API void delFile (const std::string &, const std::string &)

  Delete a file.
- FMS_API void extendFile (const std::string &, const std::string &, SectorId, uint32_t fit=1)

  Extend the size of a file.
- FMS_API FileHeader findFile (const std::string &, size_t &)

  Find file in disk.
- FMS_API void defrag (size_t rounds=0)

  Defrag the disk.
- FMS_API std::unique_ptr< Fcb > openFile (const std::string &fileName, const std::string &userName, const std::string &mode)

  Open a file.
- FMS_API void uploadFile (const std::string &, const std::string &, const std::string &, uint32_t fit=1, uint8_t recSizeOption=0)

  Upload a file from real disk to disk.
- FMS_API void downloadFile (const std::string &, const std::string &, const std::string &)

  Download a file from disk to real disk.
- FMS_API ClusterId sizeOfFile (const std::string &fName)

  Returns size of file in Clusters.
- FMS_API std::vector< std::pair < ClusterId, SectorId > > locationsVectorOfFile (const std::string &fName)

  Returns size vector of file.

Static Public Member Functions

- static ClusterId findFirstClusterOfFile (DatType)

  What should be the first cluster of the file if it's from the start.

Friends

- FMS_API friend std::ostream & operator<< (std::ostream &, const Disk &)

  Pipe disk object to ostream.

6.5.1 Detailed Description

Represent a disk in memory.

This could be used to mount a virtual disk from the HDD in the correct format, and do various tasks on it.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 Fms::Disk::Disk ( )

Default constructor.

See also

Disk(std::string &, std::string &, char)

6.5.2.2 Fms::Disk::Disk (std::string & *diskName*, std::string & *ownerName*, char *code*)

Construct using disk name.

Parameters

diskName	Disk name (file name)
ownerName	Owner of disk
code	'c' - Create, 'm' - Mount

Warning

An exception may be thrown when the flag in incorrect / request to create a file for an existing file / request to mount non existing file.

See also

Disk()

6.5.2.3 Fms::Disk::~Disk ( )

Destrcutor. Enforces safe unmount.

6.5.3 Member Function Documentation

6.5.3.1 void Fms::Disk::alloc (DatType & *fat*, SectorId *numSectors*, uint32_t *mode*)

Allocate sectors to file.

Parameters

fat	FAT which will receieve the sectors.
numSectors	Amount of sectors to be allocated.
mode	Allocation method. 0 - First fit. 1 - Best fit. 2 - Worst fit.

See also

dealloc(DatType &) allocExtend(DatType &, SectorId, uint32_t)

6.5.3.2 void Fms::Disk::allocExtend (DatType & *fat*, SectorId *numSectors*, uint32_t *mode*)

Allocate more sectors to file.

Parameters

fat	FAT which will receieve the sectors.
numSectors	Amount of sectors to be allocated in addition to what the FAT already has.
mode	Allocation method. 0 - First fit. 1 - Best fit. 2 - Worst fit.

See also

alloc(DatType &, SectorId, uint32_t) dealloc(DatType &)

6.5.3.3 void Fms::Disk::createDisk (std::string & *diskName*, std::string & *diskOwner*)

Create a new unformatted disk.

Parameters

diskName	Name of disk
diskOwner	Name of disk owner

6.5.3.4 void Fms::Disk::createFile (const std::string & *fileName*, const std::string & *ownerName*, const std::string & *fileType*, size_t *recordLength*, SectorId *numSectors*, const std::string & *keyType*, size_t *keyOffset*, uint32_t *fit* = 1, size_t *keyLength* = 1)

Create a new file.

The file is in the root directory. No other folder is available as per the spec.

Parameters

fileName	Name of file to create
ownerName	name of file owner
fileType	Type of file. F - Fixed length. V - Variable length.
recordLength	Length of each record / Maximum length for variable length.
numSectors	Amount of sectors that file should have
keyType	Record key type. I - Integer, F - Float, D - Double, C - Char.
keyOffset	Offset of key in record
fit	Fit type. 0 - First fit. 1 - Best fit. 2 - Worst fit.
keyLength	When key type is char: Size of key

See also

delFile(const std::string &, const std::string &) extendFile(const std::string &, const std::string &, SectorId, uint32_t fit = 1)

6.5.3.5 void Fms::Disk::dealloc (DatType & *fat*)

Mark the sectors in given FAT as free.

This works by marking the DAT as (DAT | FAT). Afterwards, the given FAT would be filled with 0's.

Parameters

fat	FAT to free.

See also

alloc(DatType &, SectorId, uint32_t) allocExtend(DatType &, SectorId, uint32_t)

6.5.3.6 void Fms::Disk::defrag (size_t *rounds* = 0)

Defrag the disk.

Try to defrag the root directory

Parameters

rounds	Maximum amount of rounds to try and defrag, 0 for unlimited.

6.5.3.7 void Fms::Disk::delFile (const std::string & *fileName*, const std::string & *ownerName*)

Delete a file.

The file is in the root directory. No other folder is available as per the spec.

Parameters

fileName	Name of file to delete
ownerName	Name of file owner

See also

delFile(const std::string &, const std::string &)
createFile(const std::string &fileName, const std::string &ownerName, const std::string &fileType, size_t recordLength, SectorId numSectors, const std::string &keyType, size_t keyOffset, size_t keyLength = 1)

6.5.3.8 void Fms::Disk::downloadFile (const std::string & *dest*, const std::string & *fName*, const std::string & *fOwner*)

Download a file from disk to real disk.

Parameters

dest	Path / Filename of destination.
fName	File name on disk.
fOwner	Owner name.

6.5.3.9 void Fms::Disk::extendFile (const std::string & *fileName*, const std::string & *ownerName*, SectorId *amount*, uint32_t *fit* = 1)

Extend the size of a file.

The file is in the root directory. No other folder is available as per the spec.

Parameters

fileName	Name of file to extend
ownerName	Name of file owner
amount	Amount of sectors to add
fit	Fit type. 0 - First fit. 1 - Best fit. 2 - Worst fit.

See also

delFile(const std::string &, const std::string &)
createFile(const std::string &fileName, const std::string &ownerName, const std::string &fileType, size_t recordLength, SectorId numSectors, const std::string &keyType, size_t keyOffset, uint32_t fit = 1, size_t keyLength = 1)

6.5.3.10 FileHeader Fms::Disk::findFile (const std::string & *fileName*, size_t & *ePlace*)

Find file in disk.

The file is in the root directory. No other folder is available as per the spec.

Parameters

fileName	Name of file to find
ePlace	Entry place of file info after the func ending.

Returns

FileHeader obj of the file

6.5.3.11 ClusterId Fms::Disk::findFirstClusterOfFile (DatType *fat*) [static]

What should be the first cluster of the file if it's from the start.

Parameters

fat	The fat to check against

Returns

Suggested value for firfst cluster of file.

6.5.3.12 void Fms::Disk::format (std::string *ownerName*)

Formats the disk.

Parameters

ownerName	The name of the disk owner

6.5.3.13 const Dat & Fms::Disk::getDat ( ) const

Return the disk dat.

Returns

Current dat of disk

6.5.3.14 const Dir & Fms::Disk::getDir ( ) const

Return current visible directory in disk.

In reality it's going to be the rootDir, as only that is supported (i.e. no subfolders avilable, though it can be extended to).

Returns

Current dir in disk

6.5.3.15 fstream * Fms::Disk::getDskfl ( )

Get the pointer to the opened disk.

Returns

A pointer (fstream) to the disk file.

6.5.3.16 const VolumeHeader & Fms::Disk::getVolumeHeader ( ) const

Return the VolumeHeader of the mounted disk.

Returns

Current Vhd

6.5.3.17 ClusterId Fms::Disk::howMuchEmpty ( ) const

Returns the amount of free clusters in the disk.

Returns

Amount of free clusters

6.5.3.18 std::vector< std::pair< ClusterId, SectorId > > Fms::Disk::locationsVectorOfFile (const std::string & *fName*)

Returns size vector of file.

Parameters

fName	Name of file

Returns

Vector of allocations

6.5.3.19 void Fms::Disk::mountDisk (std::string & *diskName*)

Mount a disk.

Parameters

diskName	Name of the disk

6.5.3.20 unique_ptr< Fcb > Fms::Disk::openFile (const std::string & *fileName*, const std::string & *userName*, const std::string & *mode*)

Open a file.

Parameters

fileName	Name of the file
userName	Name of user requesting the file
mode	Mode of openning. I - Input. O - Output. IO - Input/Output. E - Edit.

Returns

Unique_ptr to an Fcb for the file.

6.5.3.21 void Fms::Disk::readSector (SectorId *target*, Sector * *sector*)

Read data from sector.

After reading, the needle will move to the next sector.

Parameters

target	Id of sector to be read
sector	Sector pointer to receive the data

See also

readSector(Sector*)

6.5.3.22 void Fms::Disk::readSector (Sector * *sector*)

Read data from sector.

After reading, the needle will move to the next sector.

Parameters

sector	Pointer to receive the data

See also

readSector(SectorId, Sector*)

6.5.3.23 void Fms::Disk::recreateDisk (std::string & *diskOwner*)

Recreates the disk.

Can only be used after a disk has been chosen (via createDisk or mountDisk).

Parameters

diskOwner	Name of disk owner

6.5.3.24 void Fms::Disk::seekToSector (SectorId *target*)

Move the sector needle to the requested sector.

6.5.3.25 ClusterId Fms::Disk::sizeOfFile (const std::string & *fName*)

Returns size of file in Clusters.

Parameters

fName	Name of file

Returns

Size of file

6.5.3.26 void Fms::Disk::unmountDisk ( )

Unmount mounted disk.

6.5.3.27 void Fms::Disk::uploadFile (const std::string & *src*, const std::string & *fName*, const std::string & *fOwner*, uint32_t *fit* = 1, uint8_t *recSizeOption* = 0)

Upload a file from real disk to disk.

Parameters

src	Path / Filename of source.
fName	File name on disk to be created.
fOwner	Owner name.
fit	Allocation fit. 0 - First fit. 1 - Best fit. 2 - Worst fit.
recSizeOption	Record size option. 0 - 30, 1 - 64, 2 - 98, 3 - 200.

6.5.3.28 void Fms::Disk::writeFreeSector (SectorId *target*, Sector * *sector*)

Write data to a free specified sector.

Data can only be written to a sector marked as free. After writing, the needle will move to the next sector.

Parameters

target	Id of sector to be written
sector	data to be written

See also

writeSector(Sector*) writeSector(SectorId, Sector*)

6.5.3.29 void Fms::Disk::writeSector (SectorId *target*, Sector * *sector*)

Write data to specified sector.

After writing, the needle will move to the next sector.

Parameters

target	Id of sector to be written
sector	data to be written

See also

writeSector(Sector∗) writeFreeSector(SectorId, Sector∗);

6.5.3.30 void Fms::Disk::writeSector (Sector ∗ *sector*)

Write data to sector.

After writing, the needle will move to the next sector.

Parameters

sector	Data to be written.

See also

writeSector(SectorId, Sector∗) writeFreeSector(SectorId, Sector∗);

6.5.4 Friends And Related Function Documentation

6.5.4.1 FMS_API friend std::ostream& operator<< (std::ostream & *out*, const Disk & *disk*) [friend]

Pipe disk object to ostream.

Parameters

out	Stream object
disk	The disk to print

Returns

The stream object

The documentation for this class was generated from the following files:

- Disk.h
- Disk.cpp

6.6 Fms::Dms Class Reference

Disk Management System.

```
#include <Dms.h>
```

Public Member Functions

- FMS_API Dms (size_t fcbArrSize=5)

  Constructor.
- FMS_API ∼Dms ()

  Destructor.

- FMS_API Fcb ∗ openFile (Disk ∗, const std::string &, const std::string &, const std::string &)

  Open a file.
- FMS_API Fcb ∗ lookForFcb (Disk ∗, const std::string &)

  Find a free Fcb to open a file.

6.6.1 Detailed Description

Disk Management System.

This is used to manage multiple Fcb objects as well as limit the openning of already open files. This is the system behind that manages which files are open and in what way.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Fms::Dms::Dms (size_t *fcbArrSize* = 5)

Constructor.

Parameters

fcbArrSize	of Fcb items this instance should be capable of handling.

6.6.2.2 Fms::Dms::∼Dms ( )

Destructor.

6.6.3 Member Function Documentation

6.6.3.1 Fcb ∗ Fms::Dms::lookForFcb (Disk ∗ *d*, const std::string & *fName*)

Find a free Fcb to open a file.

Parameters

d	Disk
fName	File name

Returns

Pointer to Fcb. Valid only in the scope of Dms. In case none were found value is NULL.

6.6.3.2 Fcb ∗ Fms::Dms::openFile (Disk ∗ *d*, const std::string & *fileName*, const std::string & *userName*, const std::string & *mode*)

Open a file.

Parameters

d	Disk
fileName	File name

userName	User that requests to open the file
mode	Openning opetion

Returns

Pointer to Fcb. Valid only in the scope of Dms.

The documentation for this class was generated from the following files:

- Dms.h
- Dms.cpp

6.7 Fms::Disk::FatReader Class Reference

Helper class for reading FAT Table.

```
#include <Disk.h>
```

Public Member Functions

- FatReader (const DatType &, ClusterId)
- ClusterId operator∗ ()
- ClusterId operator++ ()
- ClusterId operator++ (int)
- ClusterId operator-- ()
- ClusterId operator-- (int)
- std::vector< std::pair < ClusterId, SectorId > > getLocationsVector () const

  Returns all the locations the file has allocated.

6.7.1 Detailed Description

Helper class for reading FAT Table.

This is used for moving through FAT Table without having to worry about the bits.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 Fms::Disk::FatReader::FatReader (const DatType & *fat,* ClusterId *phyStart*)

6.7.3 Member Function Documentation

6.7.3.1 std::vector< std::pair< ClusterId, SectorId > > Fms::Disk::FatReader::getLocationsVector ( ) const

Returns all the locations the file has allocated.

Returns

std::vector<std::pair<Location,Amount>>

6.7.3.2 ClusterId Fms::Disk::FatReader::operator∗ ( )

6.7.3.3 ClusterId Fms::Disk::FatReader::operator++ ( )

6.7.3.4 ClusterId Fms::Disk::FatReader::operator++ (int)

6.7.3.5 ClusterId Fms::Disk::FatReader::operator-- ( )

6.7.3.6 ClusterId Fms::Disk::FatReader::operator-- (int)

The documentation for this class was generated from the following files:

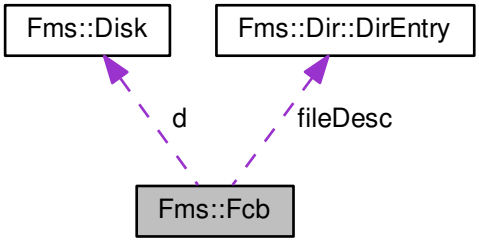
- Disk.h
- Disk.cpp

6.8 Fms::Fcb Class Reference

File control block.

`#include <Fcb.h>`

Collaboration diagram for Fms::Fcb:



Public Types

- enum OpenMode : int { OpenMode::I, OpenMode::O, OpenMode::IO }

  The method which the FCB can be used.

Public Member Functions

- FMS_API Fcb ()

  Default constructor.
- FMS_API Fcb (Disk ∗)

  Constructor that sets disk ptr automatically.
- FMS_API ∼Fcb ()

  Destructor.
- FMS_API void closeFile ()

  Close the file.

- FMS_API void flushFile ()

  Save changes written in current buffer to disk.
- FMS_API std::unique_ptr< char[]> read (uint32_t mode=0)

  Read a record.
- FMS_API void write (char *)

  Write a record.
- FMS_API void seek (uint32_t mode, int32_t amount)

  Seek to record.
- FMS_API void updateCancel ()

  Cancel update record.
- FMS_API void delRecord ()

  Delete the current record.
- FMS_API void update (char *)

  Update the current record.
- FMS_API bool eof ()

Public Attributes

- Disk * d

  Pointer to the Disk this FCB effects.
- Dir::DirEntry fileDesc

  Current file description.
- DatType fat

  Fat of current read file.
- OpenMode mode

  Method which the FCB is being used.

6.8.1 Detailed Description

File control block.

This is used to open a file and maniplulate each and every record inside it. It enables moving through the records, reading them as well as writing them.

6.8.2 Member Enumeration Documentation

6.8.2.1 enum Fms::Fcb::OpenMode : int [strong]

The method which the FCB can be used.

Enumerator

I

O

IO

6.8.3 Constructor & Destructor Documentation

6.8.3.1 Fms::Fcb::Fcb ( )

Default constructor.

6.8.3.2 Fms::Fcb::Fcb (Disk * *disk*)

Constructor that sets disk ptr automatically.

6.8.3.3 Fms::Fcb::~Fcb ( )

Destructor.

6.8.4 Member Function Documentation

6.8.4.1 void Fms::Fcb::closeFile ( )

Close the file.

6.8.4.2 void Fms::Fcb::delRecord ( )

Delete the current record.

Delete the current record by setting the record key as 0/NULL. After deletion it moves to next record.

6.8.4.3 bool Fms::Fcb::eof ( )

Check for EOF.

Returns

bool whenever or not logical EOF was reached.

6.8.4.4 void Fms::Fcb::flushFile ( )

Save changes written in current buffer to disk.

6.8.4.5 std::unique_ptr< char[] > Fms::Fcb::read (uint32_t *mode* = 0)

Read a record.

The returned char[] size is of fileDesc.actualRecSize.

Parameters

mode	0 - Read and move forward. 1 - Read and lock for changing the record.

Returns

Char array of the string. Size is fileDesc.actualRecSize.

See also

read(char*, uint32_t)

6.8.4.6 void Fms::Fcb::seek (uint32_t *mode*, int32_t *amount*)

Seek to record.

Moves the needle to the requested record. Buffer gets updated.

Parameters

mode	0 - From start. 1 - From current location. 2 - From EOF.
amount	Amount of records to move.

6.8.4.7 void Fms::Fcb::update (char * *rec*)

Update the current record.

Update the current record with a new one.

Parameters

rec	Char stream at the size of fileDesc.actualRecSize to be written instead of current record.

6.8.4.8 void Fms::Fcb::updateCancel ( )

Cancel update record.

6.8.4.9 void Fms::Fcb::write (char * *src*)

Write a record.

Parameters

src	Char stream at the size of fileDesc.actualRecSize to be written.

6.8.5 Member Data Documentation

6.8.5.1 Disk* Fms::Fcb::d

Pointer to the Disk this FCB effects.

6.8.5.2 DatType Fms::Fcb::fat

Fat of current read file.

6.8.5.3 Dir::DirEntry Fms::Fcb::fileDesc

Current file description.

6.8.5.4 OpenMode Fms::Fcb::mode

Method which the FCB is being used.

The documentation for this class was generated from the following files:

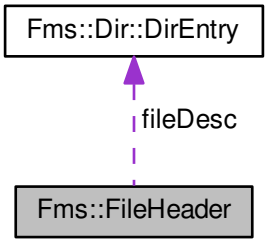
- Fcb.h
- Fcb.cpp

6.9 Fms::FileHeader Struct Reference

Defined structure for file header.

```
#include <FileHeader.h>
```

Collaboration diagram for Fms::FileHeader:



Public Member Functions

- FMS_API bool operator== (const FileHeader &f) const
- FMS_API bool operator!= (const FileHeader &f) const

Public Attributes

- SectorId sectorNr

  Sector where this item is located.
- Dir::DirEntry fileDesc

  Descritpion of file.
- DatType FAT

  Mark which sectors are being used by the file.
- char emptyArea [SIZEOFSECTOR-sizeof(SectorId)-sizeof(Dir::DirEntry)-sizeof(DatType)]

  Empty area in order for item to be the same size as sector.

6.9.1 Detailed Description

Defined structure for file header.

6.9.2 Member Function Documentation

6.9.2.1 bool Fms::FileHeader::operator!= (const FileHeader & *f*) const

6.9.2.2 bool Fms::FileHeader::operator== (const FileHeader & *f*) const

6.9.3 Member Data Documentation

6.9.3.1 char Fms::FileHeader::emptyArea[SIZEOFSECTOR-sizeof(SectorId)-sizeof(Dir::DirEntry)-sizeof(DatType)]

Empty area in order for item to be the same size as sector.

6.9.3.2 DatType Fms::FileHeader::FAT

Mark which sectors are being used by the file.

6.9.3.3 Dir::DirEntry Fms::FileHeader::fileDesc

Descritpion of file.

6.9.3.4 SectorId Fms::FileHeader::sectorNr

Sector where this item is located.

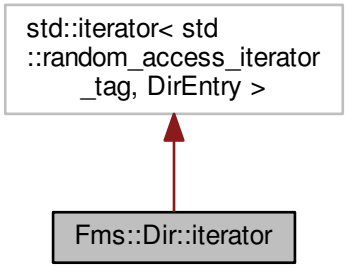
The documentation for this struct was generated from the following files:

- FileHeader.h
- FileHeader.cpp

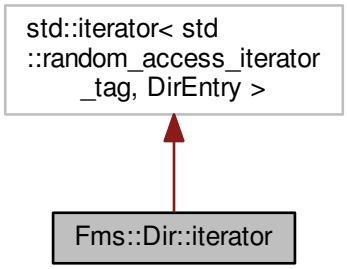
6.10 Fms::Dir::iterator Class Reference

```
#include <Dir.h>
```

Inheritance diagram for Fms::Dir::iterator:



Collaboration diagram for Fms::Dir::iterator:



Public Member Functions

- FMS_API iterator (Dir ∗p, SectorId loc=0)
- FMS_API bool operator== (const iterator &) const
- FMS_API bool operator!= (const iterator &) const
- FMS_API bool operator> (const iterator &) const
- FMS_API bool operator< (const iterator &) const
- FMS_API bool operator>= (const iterator &) const
- FMS_API bool operator<= (const iterator &) const
- FMS_API DirEntry & operator∗ ()
- FMS_API DirEntry ∗ operator-> ()
- FMS_API DirEntry & operator[] (int)
- FMS_API iterator operator++ ()
- FMS_API iterator operator++ (int)
- FMS_API iterator operator-- ()
- FMS_API iterator operator-- (int)
- FMS_API iterator operator+ (const int &) const
- FMS_API iterator operator- (const int &) const
- FMS_API iterator operator+= (const int &)
- FMS_API iterator operator-= (const int &)
- FMS_API int operator- (const iterator &) const

6.10.1 Constructor & Destructor Documentation

6.10.1.1 Fms::Dir::iterator::iterator (Dir ∗ *p*, SectorId *loc* = 0)

6.10.2 Member Function Documentation

6.10.2.1 bool Fms::Dir::iterator::operator!= (const iterator & *it*) const

6.10.2.2 Dir::DirEntry & Fms::Dir::iterator::operator∗ ( )

6.10.2.3 Dir::iterator Fms::Dir::iterator::operator+ (const int & *n*) const

6.10.2.4 Dir::iterator Fms::Dir::iterator::operator++ ( )

6.10.2.5 Dir::iterator Fms::Dir::iterator::operator++ (int)

6.10.2.6 Dir::iterator Fms::Dir::iterator::operator+= (const int & *n*)

6.10.2.7 Dir::iterator Fms::Dir::iterator::operator- (const int & *n*) const

6.10.2.8 int Fms::Dir::iterator::operator- (const iterator & *it*) const

6.10.2.9 Dir::iterator Fms::Dir::iterator::operator-- ( )

6.10.2.10 Dir::iterator Fms::Dir::iterator::operator-- (int)

6.10.2.11 Dir::iterator Fms::Dir::iterator::operator-= (const int & *n*)

6.10.2.12 Dir::DirEntry * Fms::Dir::iterator::operator-> ( )

6.10.2.13 bool Fms::Dir::iterator::operator< (const iterator & *it*) const

6.10.2.14 bool Fms::Dir::iterator::operator<= (const iterator & *it*) const

6.10.2.15 bool Fms::Dir::iterator::operator== (const iterator & *it*) const

6.10.2.16 bool Fms::Dir::iterator::operator> (const iterator & *it*) const

6.10.2.17 bool Fms::Dir::iterator::operator>= (const iterator & *it*) const

6.10.2.18 Dir::DirEntry & Fms::Dir::iterator::operator[] (int)

The documentation for this class was generated from the following files:

- Dir.h
- Dir.cpp

6.11 Fms::Sector Struct Reference

Represent a Sector on the HDD.

```
#include <Sector.h>
```

Public Member Functions

- FMS_API Sector ()

  Default constructor.
- FMS_API bool operator== (const Sector &) const

  Operator ==.
- FMS_API bool operator!= (const Sector &) const

  Operator !=.

Public Attributes

- SectorId sectorNr

  Sector where this item is located.
- char rawData [SIZEOFSECTOR-sizeof(SectorId)]

  Data in various format.

Friends

- FMS_API friend std::ostream & operator<< (std::ostream &, const Sector &)

  Pipe sector object to ostream.

6.11.1 Detailed Description

Represent a Sector on the HDD.

Sector is the most basic unit that can be used to store data.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 Fms::Sector::Sector ( )

Default constructor.

6.11.3 Member Function Documentation

6.11.3.1 bool Fms::Sector::operator!= (const Sector & *s*) const

Operator !=.

Parameters

s	Sector

Returns

this != sector

6.11.3.2 bool Fms::Sector::operator== (const Sector & *s*) const

Operator ==.

Parameters

s	Sector

Returns

this == sector

6.11.4 Friends And Related Function Documentation

6.11.4.1 FMS_API friend std::ostream& operator<< (std::ostream & *out*, const Sector & *sector*) [friend]

Pipe sector object to ostream.

Parameters

out	Stream object

sector	The sector to print

Returns

The stream object

6.11.5 Member Data Documentation

6.11.5.1 char Fms::Sector::rawData[SIZEOFSECTOR-sizeof(SectorId)]

Data in various format.

6.11.5.2 SectorId Fms::Sector::sectorNr

Sector where this item is located.

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

- Sector.h
- Sector.cpp

6.12 FmsUtils::Student Struct Reference

Student item.

```
#include <Student.h>
```

Public Member Functions

- FMS_API Student ()

  Default constructor.
- FMS_API Student (std::unique_ptr< char[] > c)

  Create student from char stream.
- FMS_API Student (char *c)

  Create student from char stream.
- FMS_API std::unique_ptr< char[] > toCharStream ()

  Create charstream of record.

Public Attributes

- int32_t id

  Student ID.
- char name [42]

  Student name.
- char address [150]

  Student address.
- int32_t average

  Student average.

6.12.1 Detailed Description

Student item.

This is a simple student.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 FmsUtils::Student::Student ( )

Default constructor.

6.12.2.2 FmsUtils::Student::Student (std::unique_ptr< char[] > *c*)

Create student from char stream.

The size of the char stream should be 200 bytes

Parameters

c	Char stream

See also

Record(char*)

6.12.2.3 FmsUtils::Student::Student (char * *c*)

Create student from char stream.

The size of the char stream should be 200 bytes

Parameters

c	Char stream

See also

Record(std::unique_ptr<char[]>)

6.12.3 Member Function Documentation

6.12.3.1 std::unique_ptr< char[] > FmsUtils::Student::toCharStream ( )

Create charstream of record.

The length is 204 bytes.

Returns

Char stream that can be used to create the struct

6.12.4 Member Data Documentation

6.12.4.1 char FmsUtils::Student::address[150]

Student address.

6.12.4.2 int32_t FmsUtils::Student::average

Student average.

6.12.4.3 int32_t FmsUtils::Student::id

Student ID.

6.12.4.4 char FmsUtils::Student::name[42]

Student name.

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

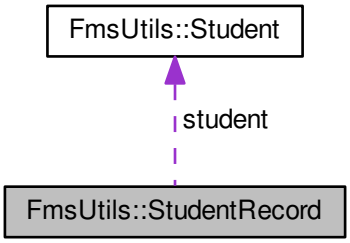
- Student.h
- Student.cpp

6.13 FmsUtils::StudentRecord Struct Reference

Student record for saving to disk.

`#include <Student.h>`

Collaboration diagram for FmsUtils::StudentRecord:



Public Member Functions

- FMS_API StudentRecord ()

  Default constructor.
- FMS_API StudentRecord (std::unique_ptr< char[]> c)

  Create student from char stream.
- FMS_API StudentRecord (char *c)

  Create student from char stream.
- FMS_API std::unique_ptr< char[]> toCharStream ()

  Create charstream of record.

Static Public Member Functions

- static FMS_API void uploadStudentsFile (Fms::Disk &d, std::string fName, std::string src)

  Upload students file.
- static FMS_API void downloadStudentsFile (Fms::Disk &d, std::string fName, std::string dst)

  Download students file.
- static FMS_API std::string studentFileAsString (Fms::Disk &d, std::string fName)

  Get all students as string.

Public Attributes

- int32_t key
- Student student

Friends

- FMS_API friend std::ostream & operator<< (std::ostream &, const Student &)
- FMS_API friend std::istream & operator>> (std::istream &, Student &)

  Pipe student object from istream.

6.13.1 Detailed Description

Student record for saving to disk.

This is the way to save the students to the disk as records.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 FmsUtils::StudentRecord::StudentRecord ( )

Default constructor.

6.13.2.2 FmsUtils::StudentRecord::StudentRecord (std::unique_ptr< char[] > *c*)

Create student from char stream.

The size of the char stream should be 204 bytes

Parameters

c	Char stream

See also

StudentRecord(char*)

6.13.2.3 FmsUtils::StudentRecord::StudentRecord (char * *c*)

Create student from char stream.

The size of the char stream should be 204 bytes

Parameters

c	Char stream

See also

StudentRecord(std::unique_ptr<char[]>)

6.13.3 Member Function Documentation

6.13.3.1 void FmsUtils::StudentRecord::downloadStudentsFile (Fms::Disk & *d*, std::string *fName*, std::string *dst*) [static]

Download students file.

Parameters

d	Disk to be downloaded from. It has to be mounted.
fName	File name on disk.
dst	File name on real disk.

6.13.3.2 std::string FmsUtils::StudentRecord::studentFileAsString (Fms::Disk & *d*, std::string *fName*) [static]

Get all students as string.

Parameters

d	Disk
fName	Name of students file

Returns

String with all students

6.13.3.3 std::unique_ptr< char[]> FmsUtils::StudentRecord::toCharStream ( )

Create charstream of record.

The length is 204 bytes.

Returns

Char stream that can be used to create the struct

6.13.3.4 void FmsUtils::StudentRecord::uploadStudentsFile (Fms::Disk & *d*, std::string *fName*, std::string *src*) [static]

Upload students file.

Parameters

d	Disk to be uploaded to. It has to be mounted.

fName	File name on disk.
src	File name on real disk.

6.13.4 Friends And Related Function Documentation

6.13.4.1 FMS_API friend std::ostream& operator<< (std::ostream & *o*, const Student & *s*) [friend]

6.13.4.2 FMS_API friend std::istream& operator>> (std::istream & *i*, Student & *s*) [friend]

Pipe student object from istream.

Parameters

i	Stream object
s	The student to print

Returns

The stream object

6.13.5 Member Data Documentation

6.13.5.1 int32_t FmsUtils::StudentRecord::key

6.13.5.2 Student FmsUtils::StudentRecord::student

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

- Student.h
- Student.cpp

6.14 Fms::VolumeHeader Struct Reference

Defined structure for Volume Header.

```
#include <VolumeHeader.h>
```

Public Member Functions

- FMS_API VolumeHeader ()

  Constrcutor for VolumeHeader.

Public Attributes

- SectorId sectorNr

  Sector where this item is located.
- char diskName [12]

  Name of disk.
- char diskOwner [12]

  Name of disk owner.
- char prodDate [10]

  When was the HDD manufactured.

- ClusterId clusQty

  Amount of clusters in disk.
- ClusterId dataClusQty

  Amount of clusters to be used by data in disk.
- SectorId addrDat

  Sector number where DAT is stored.
- ClusterId addrRootDir

  Address of the root directory in clusters (as per assignment).
- SectorId addrDatCpy

  Sector where a copy of the dat is located.
- ClusterId addrRootDirCpy

  Address of root directory copy in clusters (as per assignment).
- ClusterId addrDataStart

  First cluster which should be used for data.
- char formatDate [10]

  Date of when the disk was last formatted.
- bool isFormated

  Is the disk formatted?
- char emptyArea [SIZEOFSECTOR-sizeof(SectorId)-sizeof(char[12])-sizeof(char[12])-sizeof(char[10])-sizeof(ClusterId)-sizeof(ClusterId)-sizeof(SectorId)-sizeof(ClusterId)-sizeof(SectorId)-sizeof(ClusterId)-sizeof(ClusterId)-sizeof(char[10])-sizeof(bool)]

  Empty area in order for item to be the same size as sector.

6.14.1 Detailed Description

Defined structure for Volume Header.

The volume header should use a single sector to store its data. This data contains information about the disk.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 Fms::VolumeHeader::VolumeHeader ( )

Constrcutor for VolumeHeader.

6.14.3 Member Data Documentation

6.14.3.1 SectorId Fms::VolumeHeader::addrDat

Sector number where DAT is stored.

6.14.3.2 ClusterId Fms::VolumeHeader::addrDataStart

First cluster which should be used for data.

6.14.3.3 SectorId Fms::VolumeHeader::addrDatCpy

Sector where a copy of the dat is located.

6.14.3.4 ClusterId Fms::VolumeHeader::addrRootDir

Address of the root directory in clusters (as per assignment).

6.14.3.5 ClusterId Fms::VolumeHeader::addrRootDirCpy

Address of root directory copy in clusters (as per assignment).

6.14.3.6 ClusterId Fms::VolumeHeader::clusQty

Amount of clusters in disk.

6.14.3.7 ClusterId Fms::VolumeHeader::dataClusQty

Amount of clusters to be used by data in disk.

6.14.3.8 char Fms::VolumeHeader::diskName[12]

Name of disk.

6.14.3.9 char Fms::VolumeHeader::diskOwner[12]

Name of disk owner.

6.14.3.10 char Fms::VolumeHeader::emptyArea[SIZEOFSECTOR-sizeof(SectorId)-sizeof(char[12])-sizeof(char[12])-sizeof(char[10])-sizeof(ClusterId)-sizeof(ClusterId)-sizeof(SectorId)-sizeof(ClusterId)-sizeof(SectorId)-sizeof(ClusterId)-sizeof(ClusterId)-sizeof(char[10])-sizeof(bool)]

Empty area in order for item to be the same size as sector.

6.14.3.11 char Fms::VolumeHeader::formatDate[10]

Date of when the disk was last formatted.

6.14.3.12 bool Fms::VolumeHeader::isFormated

Is the disk formatted?

6.14.3.13 char Fms::VolumeHeader::prodDate[10]

When was the HDD manufactured.

6.14.3.14 SectorId Fms::VolumeHeader::sectorNr

Sector where this item is located.

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

- VolumeHeader.h
- VolumeHeader.cpp

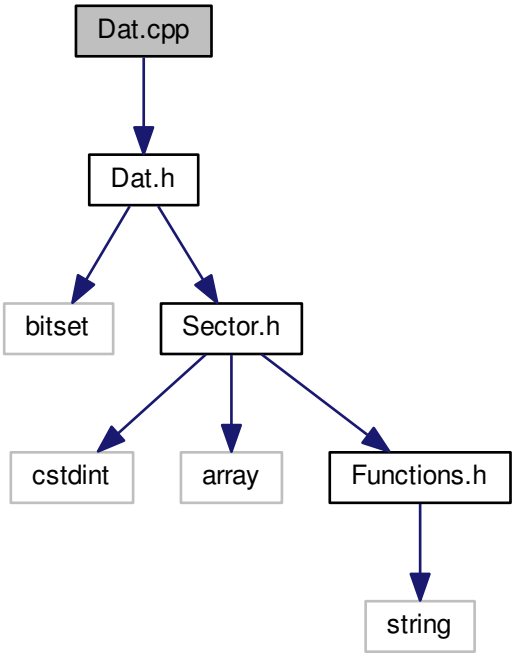
Chapter 7

File Documentation

7.1 Dat.cpp File Reference

```
#include "Dat.h"
```

Include dependency graph for Dat.cpp:



Namespaces

- Fms

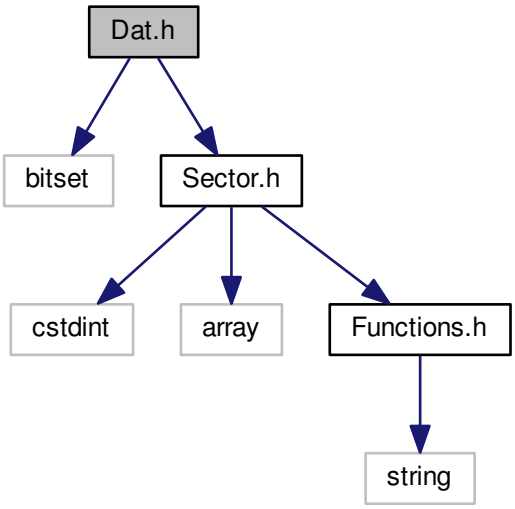
Functions

- std::ostream & Fms::operator<< (std::ostream &out, const Dat &dat)

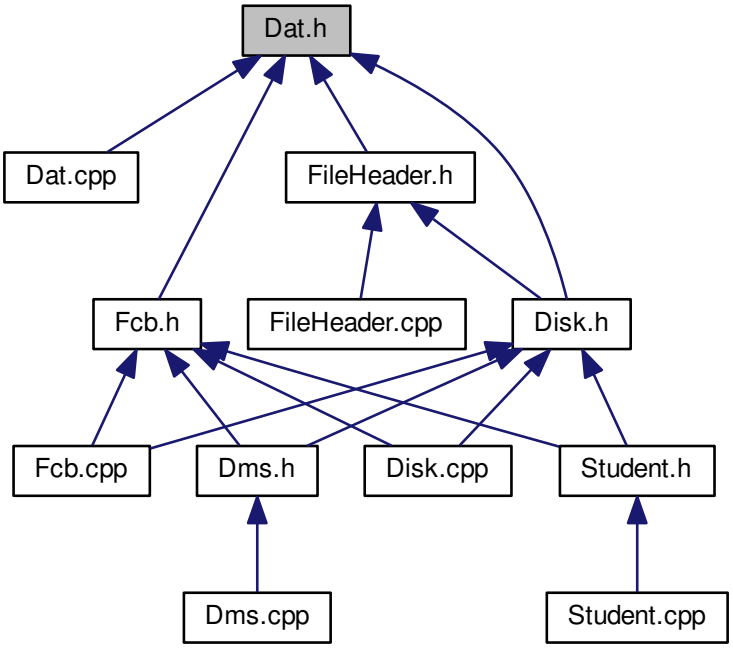
7.2 Dat.h File Reference

```
#include <bitset>
#include "Sector.h"
```

Include dependency graph for Dat.h:



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



Classes

- struct Fms::Dat

  Defined structure for Disk DAT.

Namespaces

- Fms

Macros

- #define FMS_API __declspec(dllimport)

Typedefs

- typedef std::bitset
  < CLUSTERSINDISK > Fms::DatType

  DAT / FAT bitset for marking which clusters are free / taken.

Variables

- const ClusterId Fms::CLUSTERSINDISK = 1600

  Amount of clusters in the disk.

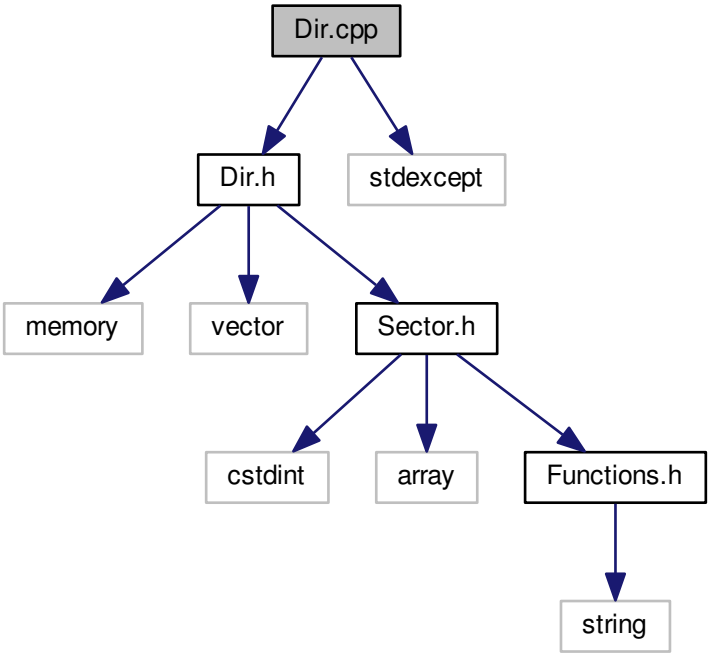
7.2.1 Macro Definition Documentation

7.2.1.1 #define FMS_API __declspec(dllimport)

7.3 Dir.cpp File Reference

```
#include "Dir.h"
#include <stdexcept>
```

Include dependency graph for Dir.cpp:



Namespaces

- Fms

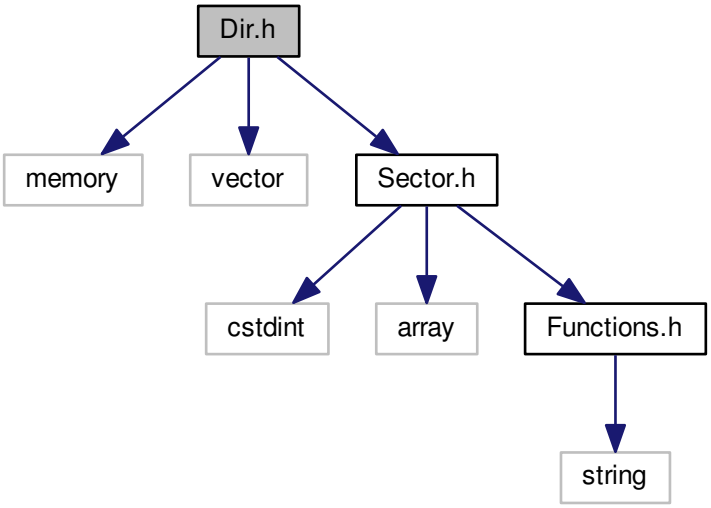
Functions

- std::ostream & Fms::operator<< (std::ostream &out, const Dir &dir)

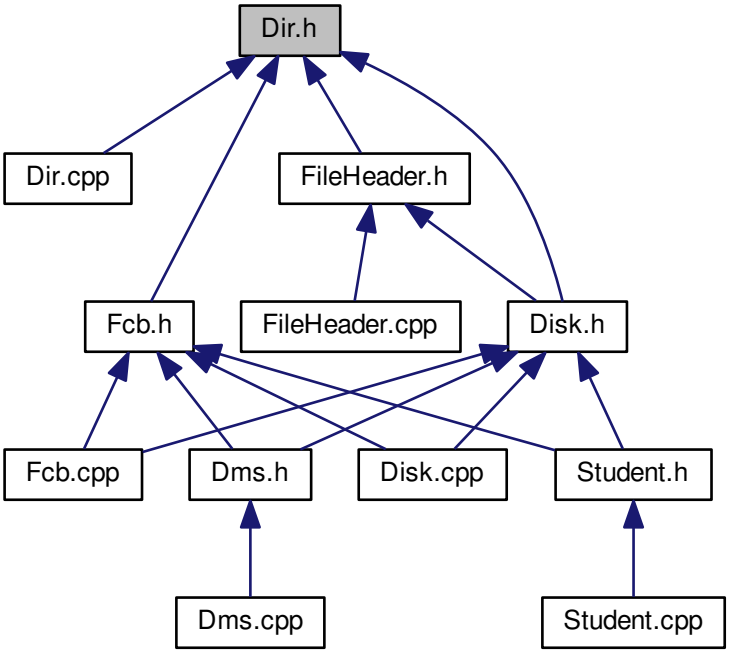
7.4 Dir.h File Reference

```
#include <memory>
#include <vector>
#include "Sector.h"
```

Include dependency graph for Dir.h:



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



Classes

- struct Fms::Dir

  Defined structure for Directory / Sub-Directory.

- struct Fms::Dir::DirEntry

  Defined structure for Directory Entry.

- class Fms::Dir::iterator
- class Fms::Dir::const_iterator

Namespaces

- Fms

Macros

- #define FMS_API __declspec(dllimport)

Typedefs

- typedef uint32_t Fms::FileSize

  Type for file size in bytes.

Variables

- const SectorId Fms::ENTRIESPERSECTOR = 14

  Amount of directory entries per sector.

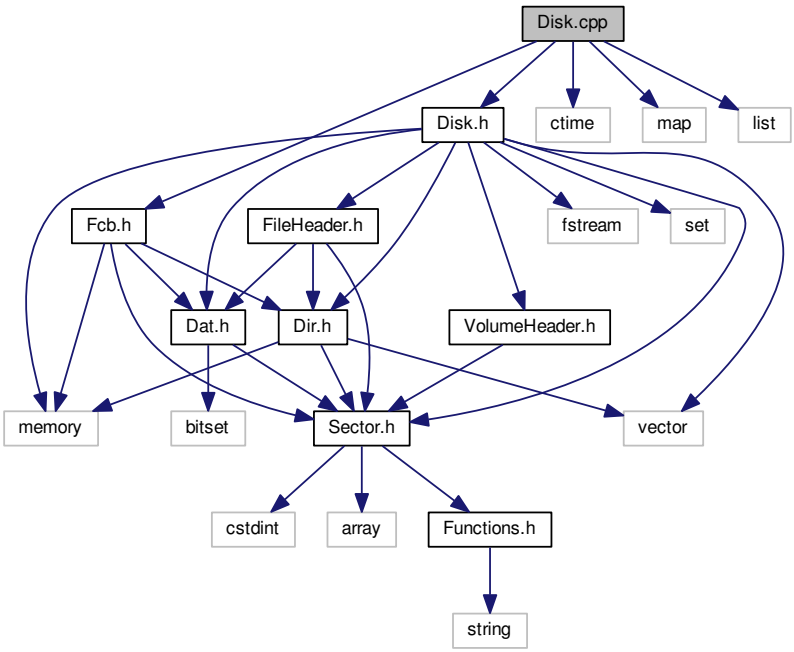
7.4.1 Macro Definition Documentation

7.4.1.1 #define FMS_API __declspec(dllimport)

7.5 Disk.cpp File Reference

```
#include "Disk.h"
#include "Fcb.h"
#include <ctime>
#include <map>
#include <list>
```

Include dependency graph for Disk.cpp:



Namespaces

- Fms

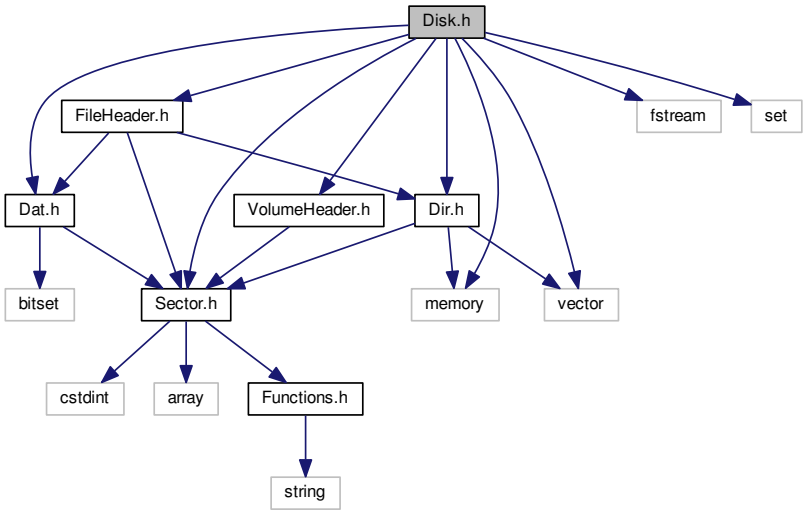
Functions

- std::ostream & Fms::operator<< (std::ostream &out, const Disk &disk)

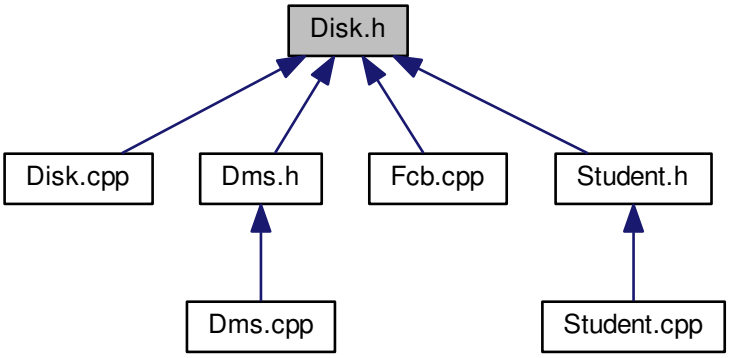
7.6 Disk.h File Reference

```
#include "Sector.h"
#include "Dat.h"
#include "Dir.h"
#include "VolumeHeader.h"
#include "FileHeader.h"
#include <fstream>
#include <memory>
#include <vector>
#include <set>
```

Include dependency graph for Disk.h:



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



Classes

- class Fms::Disk

  Represent a disk in memory.
- class Fms::Disk::FatReader

  Helper class for reading FAT Table.

Namespaces

- Fms

Macros

- #define FMS_API __declspec(dllimport)

Variables

- const ClusterId Fms::CLUSTERSRESERVED =4

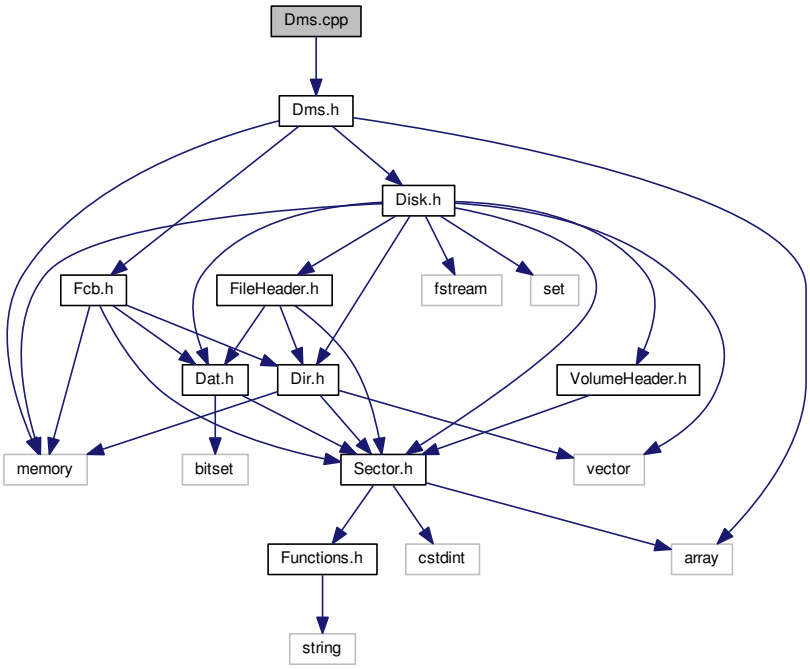
  Amount of clusters used for HDD structure.

7.6.1 Macro Definition Documentation

7.6.1.1 #define FMS_API __declspec(dllimport)

7.7 Dms.cpp File Reference

`#include "Dms.h"`
Include dependency graph for Dms.cpp:



Namespaces

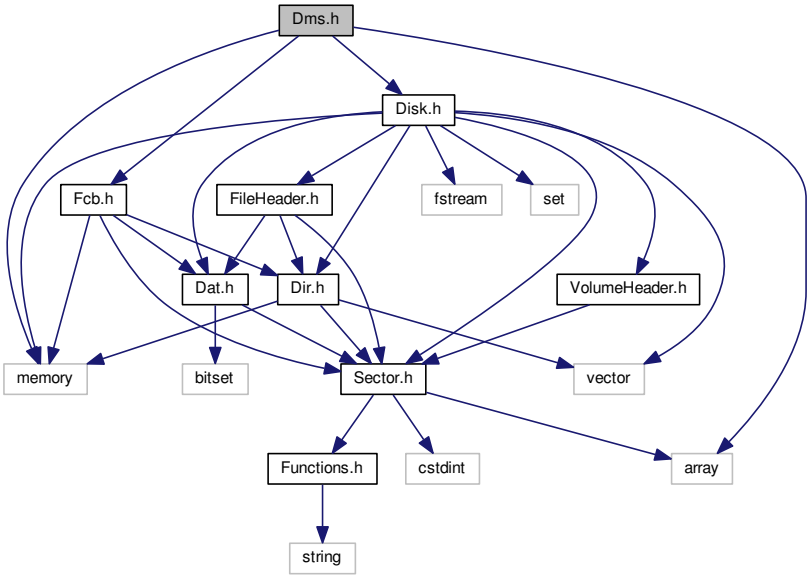
- Fms

7.8 Dms.h File Reference

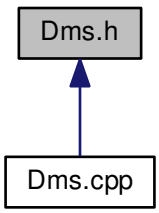
`#include "Fcb.h"`

```
#include "Disk.h"
#include <memory>
#include <array>
```

Include dependency graph for Dms.h:



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



Classes

- class Fms::Dms

  Disk Management System.

Namespaces

- Fms

Macros

- #define FMS_API __declspec(dllimport)

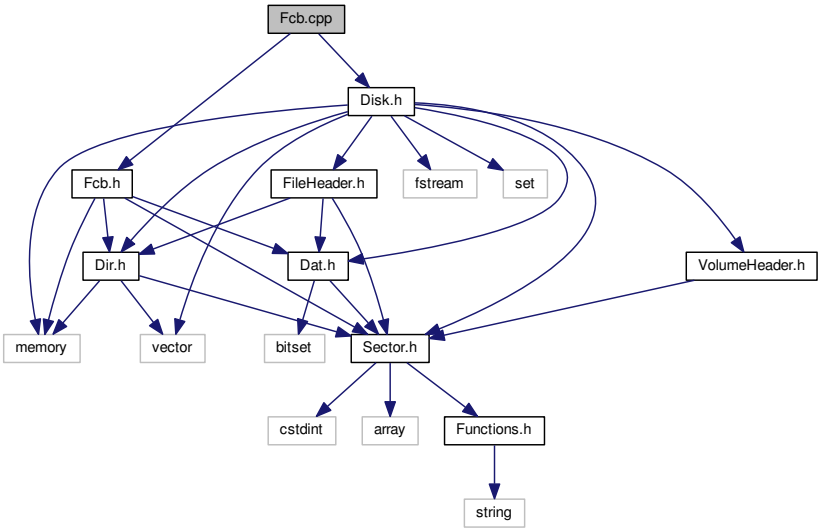
7.8.1 Macro Definition Documentation

7.8.1.1 #define FMS_API __declspec(dllimport)

7.9 Fcb.cpp File Reference

```
#include "Fcb.h"
#include "Disk.h"
```

Include dependency graph for Fcb.cpp:



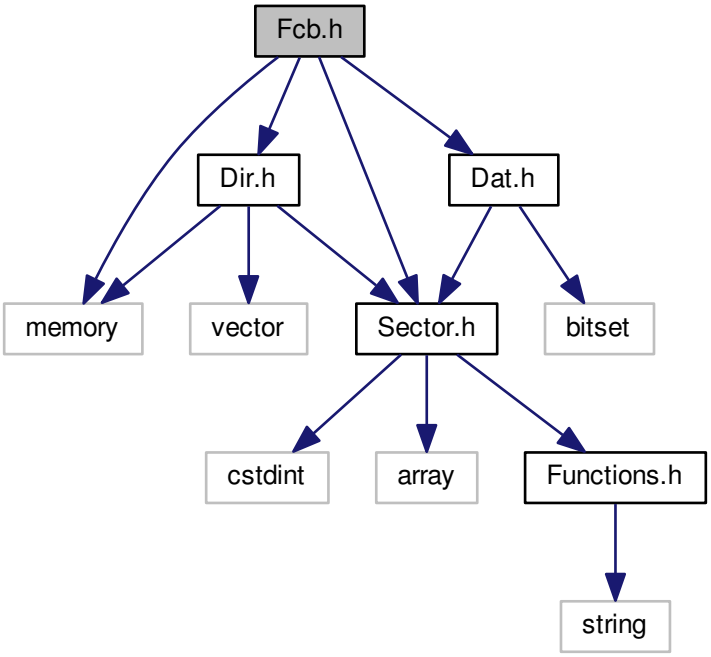
Namespaces

- Fms

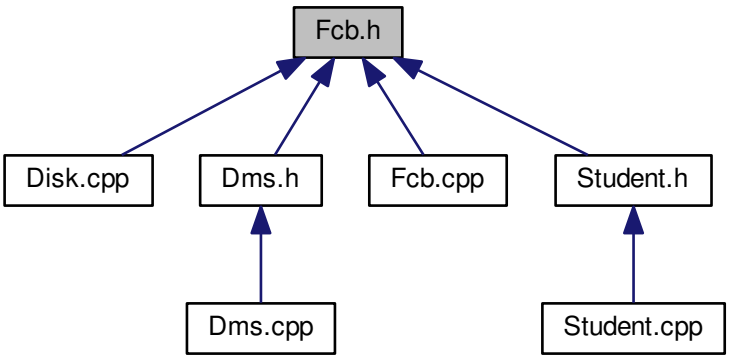
7.10 Fcb.h File Reference

```
#include "Dir.h"
#include "Dat.h"
#include "Sector.h"
#include <memory>
```

Include dependency graph for Fcb.h:



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



Classes

- class Fms::Fcb

  File control block.

Namespaces

- Fms

Macros

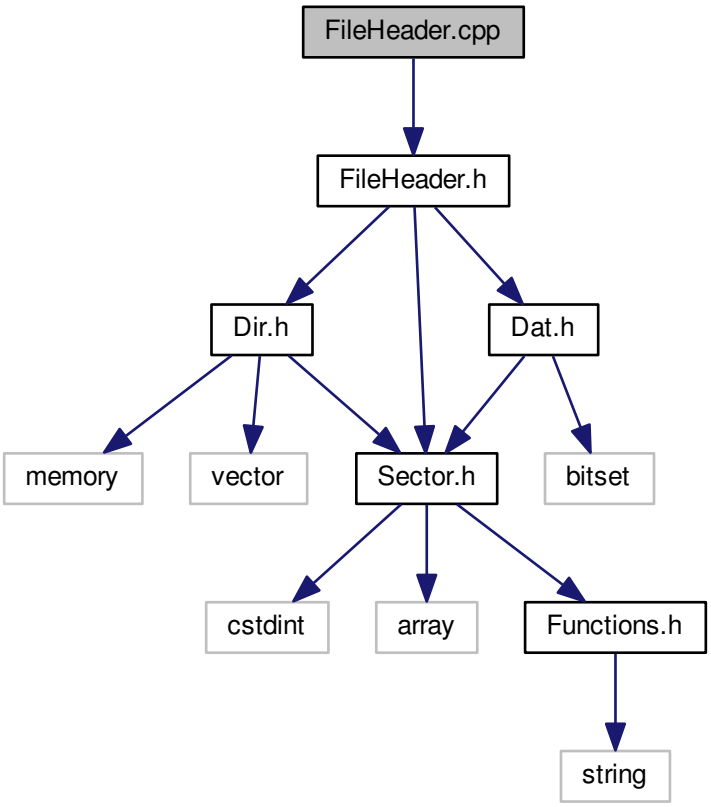
- #define FMS_API __declspec(dllimport)

7.10.1 Macro Definition Documentation

7.10.1.1 #define FMS_API __declspec(dllimport)

7.11 FileHeader.cpp File Reference

`#include "FileHeader.h"`
Include dependency graph for FileHeader.cpp:



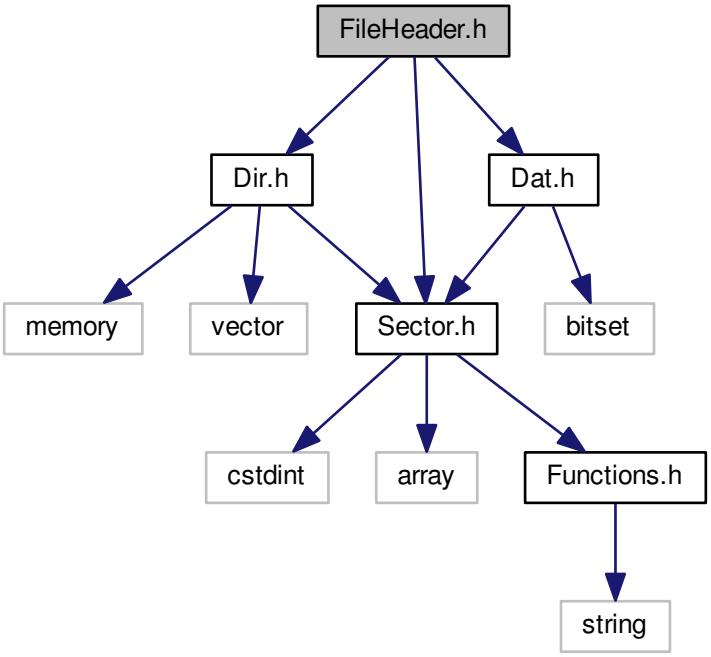
Namespaces

- Fms

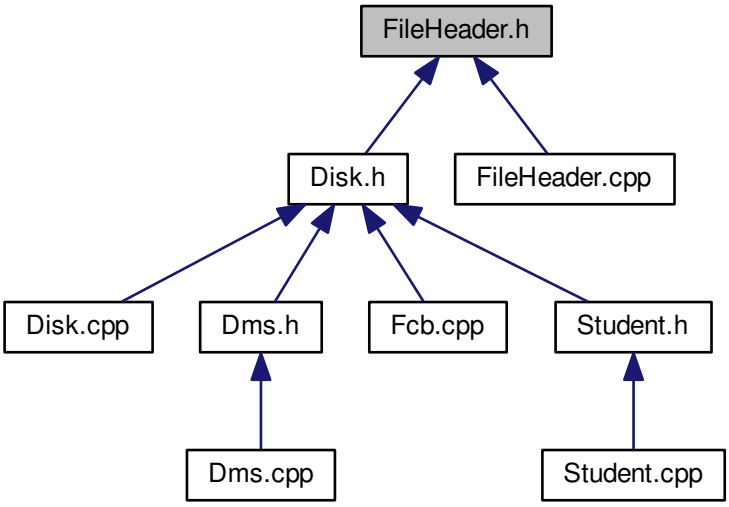
7.12 FileHeader.h File Reference

```
#include "Sector.h"
#include "Dir.h"
#include "Dat.h"
```

Include dependency graph for FileHeader.h:



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



Classes

- struct Fms::FileHeader

  Defined structure for file header.

Namespaces

- Fms

Macros

- #define FMS_API __declspec(dllimport)

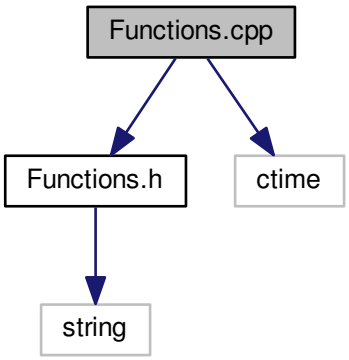
7.12.1 Macro Definition Documentation

7.12.1.1 #define FMS_API __declspec(dllimport)

7.13 Functions.cpp File Reference

```
#include "Functions.h"
#include <ctime>
```

Include dependency graph for Functions.cpp:



Namespaces

- Fms

Functions

- std::string Fms::getDate ()

  Receieve the current date in DDMMYYYY format.

- std::wstring Fms::stringToWString (const std::string &s)

  Convert string to wstring.

- std::string Fms::convertDate (const char date[])

  Convert date from DDMMYY format to something readable.

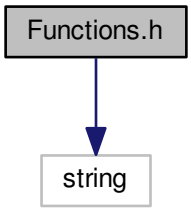
- const std::string Fms::removePath (const std::string &fileName)

  Remove path from full path.

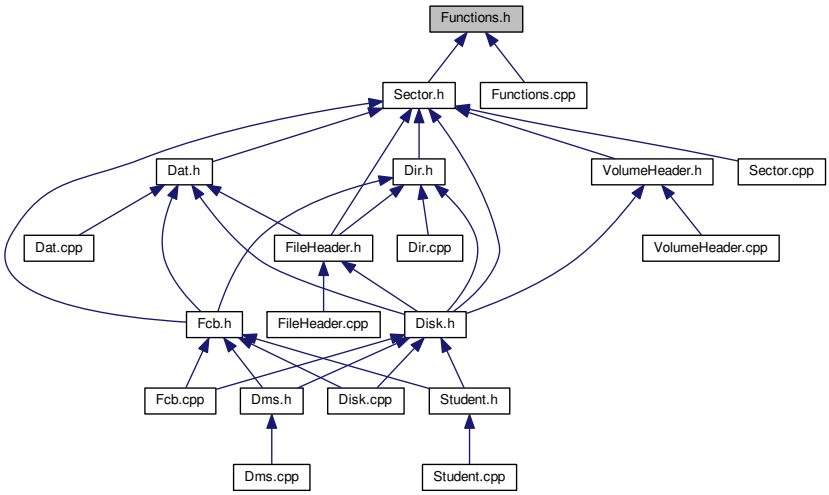
7.14 Functions.h File Reference

```
#include <string>
```

Include dependency graph for Functions.h:



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



Namespaces

- Fms

Macros

- #define FMS_API __declspec(dllimport)

Functions

- std::string Fms::getDate ()

  Receieve the current date in DDMMYYYY format.
- std::wstring Fms::stringToWString (const std::string &s)

  Convert string to wstring.
- std::string Fms::convertDate (const char date[])

Convert date from DDMMYY format to something readable.

- const std::string Fms::removePath (const std::string &fileName)

  Remove path from full path.

7.14.1 Detailed Description

This file contains helper functions for the Fms. Those functions don't really fit a class.

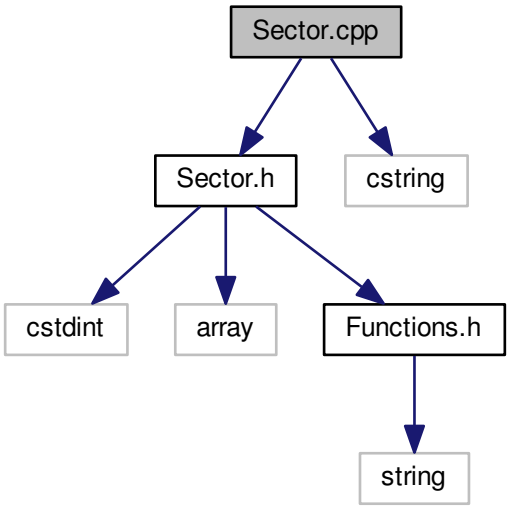
7.14.2 Macro Definition Documentation

7.14.2.1 #define FMS_API __declspec(dllimport)

7.15 Sector.cpp File Reference

```
#include "Sector.h"
#include <cstring>
```

Include dependency graph for Sector.cpp:



Namespaces

- Fms

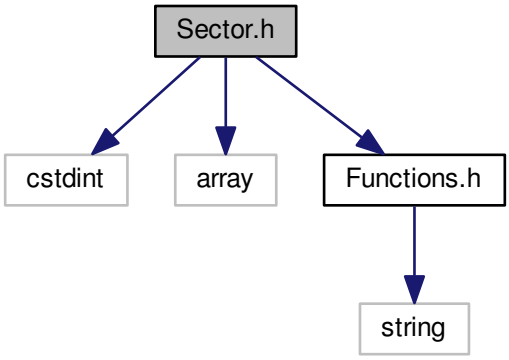
Functions

- std::ostream & Fms::operator<< (std::ostream &out, const Sector &sector)

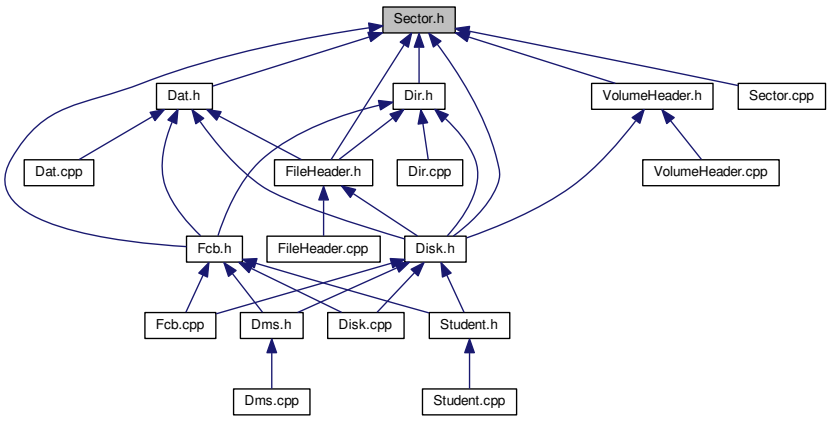
7.16 Sector.h File Reference

```
#include <cstdint>
#include <array>
#include "Functions.h"
```

Include dependency graph for Sector.h:



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



Classes

- struct Fms::Sector

  Represent a Sector on the HDD.

Namespaces

- Fms

Macros

- #define FMS_API __declspec(dllimport)

Typedefs

- typedef uint32_t Fms::SectorId

  Type for Sector ID.
- typedef uint32_t Fms::ClusterId

  Type for Cluster ID.
- typedef std::array< Sector, CLUSTERSIZE > Fms::Cluster

  Type of cluster (array of sectors)
- typedef uint32_t Fms::RecordId

  Type for Record ID.

Variables

- const SectorId Fms::SIZEOFSECTOR = 1024

  Bytes that can be stored in a single sector.
- const SectorId Fms::CLUSTERSIZE = 2

  Amount of sectors in cluster.

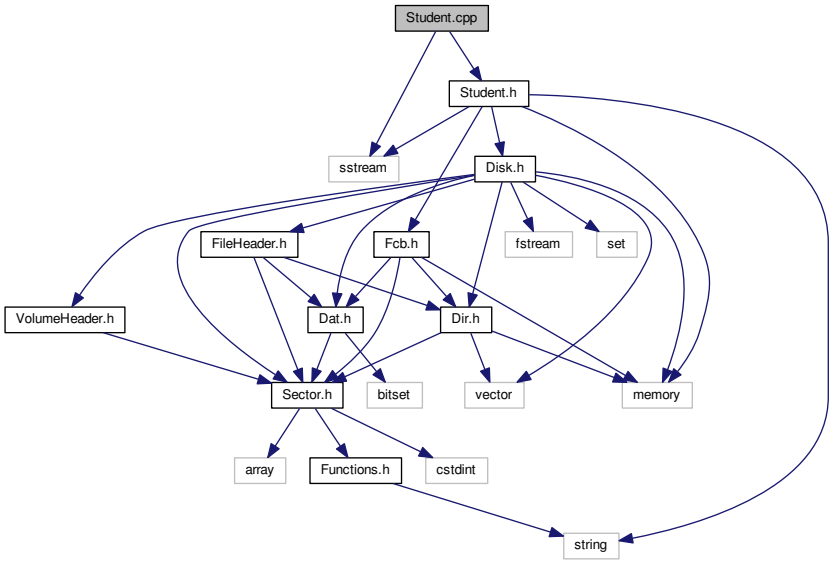
7.16.1 Macro Definition Documentation

7.16.1.1 #define FMS_API __declspec(dllimport)

7.17 Student.cpp File Reference

```
#include <sstream>
#include "Student.h"
```

Include dependency graph for Student.cpp:



Namespaces

- FmsUtils

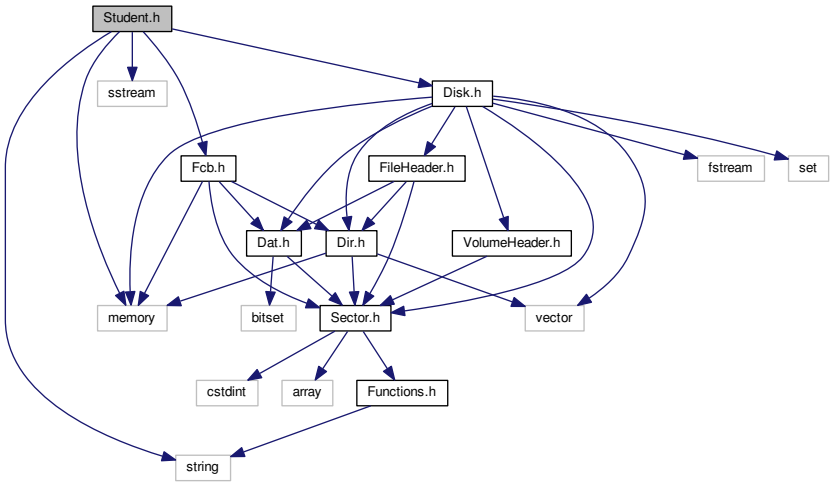
Functions

- std::ostream & FmsUtils::operator<< (std::ostream &o, const Student &s)
- std::istream & FmsUtils::operator>> (std::istream &i, Student &s)

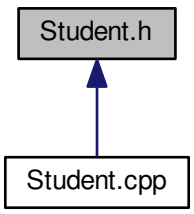
7.18 Student.h File Reference

```
#include <memory>
#include <string>
#include <sstream>
#include "Disk.h"
#include "Fcb.h"
```

Include dependency graph for Student.h:



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



Classes

- struct FmsUtils::Student

  Student item.
- struct FmsUtils::StudentRecord

  Student record for saving to disk.

Namespaces

- FmsUtils

Macros

- #define FMS_API __declspec(dllimport)

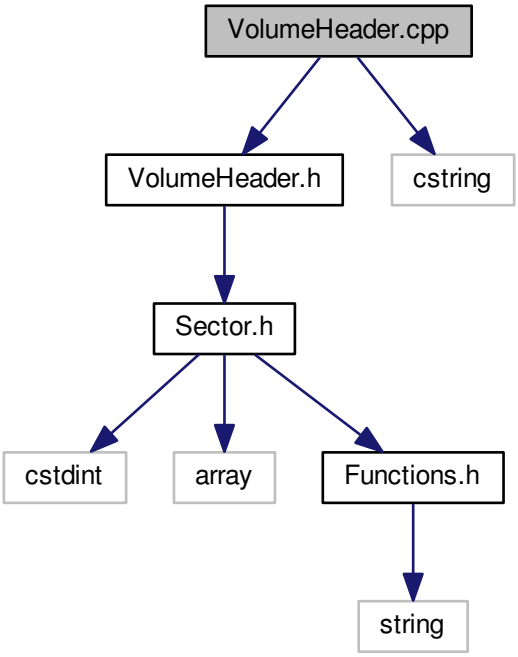
7.18.1 Macro Definition Documentation

7.18.1.1 #define FMS_API __declspec(dllimport)

7.19 VolumeHeader.cpp File Reference

```
#include "VolumeHeader.h"
#include <cstring>
```

Include dependency graph for VolumeHeader.cpp:



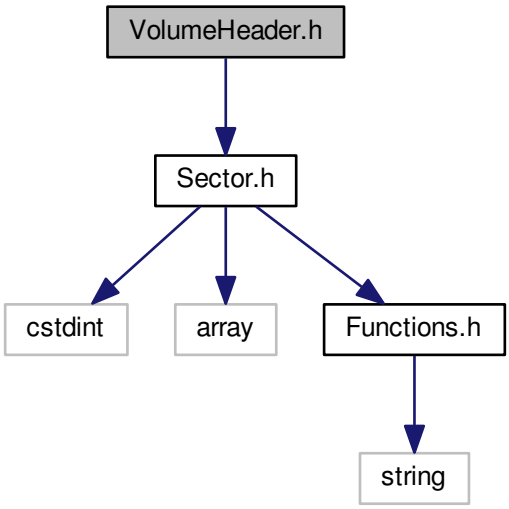
Namespaces

- Fms

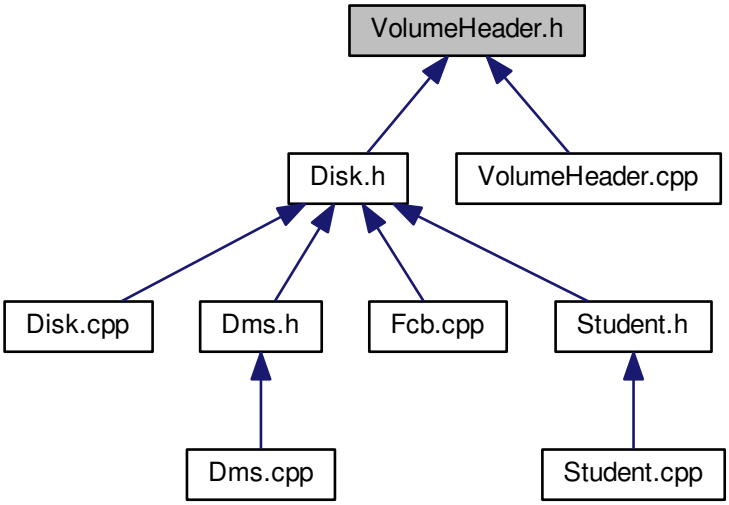
7.20 VolumeHeader.h File Reference

```
#include "Sector.h"
```

Include dependency graph for VolumeHeader.h:



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



Classes

- struct Fms::VolumeHeader

  Defined structure for Volume Header.

Namespaces

- Fms

Macros

- #define FMS_API __declspec(dllimport)

7.20.1 Macro Definition Documentation

7.20.1.1 #define FMS_API __declspec(dllimport)

Index