## FatOS

Generated by Doxygen 1.8.11

# **Contents**

1	Clas	s Index												1
	1.1	Class I	List				 	 	 	 		 		 1
2	File	Index												3
	2.1	File Lis	st				 	 	 	 		 		 3
3	Clas	s Docu	mentation	ı										5
	3.1	FileSys	stem Struc	t Reference			 	 	 	 		 		 5
		3.1.1	Detailed	Description			 	 	 	 		 		 5
		3.1.2	Member	Data Docur	nentatior	1	 	 	 	 		 		 5
			3.1.2.1	drive_id .			 	 	 	 		 		 5
			3.1.2.2	FATpointe	er		 	 	 	 		 		 5
			3.1.2.3	fileList .			 	 	 	 		 		 6
			3.1.2.4	fileListSize			 	 	 	 		 		 6
			3.1.2.5	next			 	 	 	 		 		 6
	3.2	IDT_D	escriptor S	Struct Refere	ence		 	 	 	 		 		 6
	3.3	IDT_E	ntry Struct	Reference			 	 	 	 		 		 6
	3.4	Memoi	ryHeader S	Struct Refer	ence		 	 	 	 		 		 6
	3.5	Memoi	rvmap ent	rv Struct Re	eference		 	 		 		 		 7

iv CONTENTS

4 File Documentation

4.2.3.3 filesystem_findEmptySector(unsigned short first, FileSystem *fs)
4.2.3.1 filesystem_appendBytes(FileSystem *fs, char *fileListData, int size, char *filename, int bytesToAppend)
4.2.3.1 filesystem_appendBytes(FileSystem *fs, char *fileListData, int size, char *filename, int bytesToAppend)
#filename, int bytesToAppend). 11  4.2.3.2 filesystem_appendSectors(unsigned short first, int count, int rewrite, FileSystem *fs) . 12  4.2.3.3 filesystem_findEmptySector(unsigned short first, FileSystem *fs) . 12  4.2.3.4 filesystem_findFirstSector(char *fileIname, char *filelist, int size) . 12  4.2.3.5 filesystem_getBytes(char *fileListData, int length, char *filename) . 13  4.2.3.6 filesystem_getFileSectors(unsigned short firstSector, FileSystem *fs) . 13  4.2.3.7 filesystem_getLastSector(unsigned short firstSector, FileSystem *fs) . 13  4.2.3.8 filesystem_getNextSector(unsigned short sector, FileSystem *fs) . 13  4.2.3.9 filesystem_getNextSectorRaw(unsigned short sector, FileSystem *fs) . 14  4.2.3.10 filesystem_getNthSector(unsigned short firstSector, int n, FileSystem *fs) . 14  4.2.3.11 filesystem_init(FileSystem *fs) . 14  4.2.3.12 filesystem_list(char *fileListData, int length) . 15  4.2.3.13 filesystem_LoadFileList(FileSystem *fs) . 15  4.2.3.14 filesystem_loadSubFileList(FileSystem *fs, char *fileListData, int size, char *subFileListName, char **resBuffer) . 15  4.2.3.15 filesystem_readbytesByFirstSector(unsigned short firstSector, int startByte, int length, char *buffer, FileSystem *fs) . 15  4.2.3.16 filesystem_setBytes(char *fileListData, int length, char *filename, int newSize) . 16  4.2.3.17 filesystem_setBytes(char *fileListData, int length, char *filename, int newSize) . 16  4.2.3.18 filesystem_setBytes(char *fileListData, int length, char *filename, int newSize) . 16  4.2.3.19 filesystem_setBytes(char *fileListData, int length, char *filename, int newSize) . 16  4.2.3.19 filesystem_subList(FileSystem *fs, char *fileListData, int size, char *subListName) 16  4.2.3.19 filesystem_subList(FileSystem *fs, char *fileListData, int size, char *subListName) 16  4.2.3.19 filesystem_subList(FileSystem *fs, char *fileListData, int size, char *subListName) 16  4.2.3.20 filesystem_subList(FileSystem *fs) . 17
#fs)
4.2.3.4 filesystem_findFirstSector(char *filename, char *filelist, int size)
4.2.3.5 filesystem_getBytes(char *fileListData, int length, char *filename)
4.2.3.6 filesystem_getFileSectors(unsigned short firstSector, FileSystem *fs)
4.2.3.7 filesystem_getLastSector(unsigned short firstSector, FileSystem *fs)
4.2.3.8 filesystem_getNextSector(unsigned short sector, FileSystem *fs)
4.2.3.9 filesystem_getNextSectorRaw(unsigned short sector, FileSystem *fs)
4.2.3.10 filesystem_getNthSector(unsigned short firstSector, int n, FileSystem *fs)
4.2.3.11 filesystem_init(FileSystem *fs)
4.2.3.12 filesystem_list(char *fileListData, int length)
4.2.3.13 filesystem_LoadFileList(FileSystem *fs)
4.2.3.14 filesystem_loadSubFileList(FileSystem *fs, char *fileListData, int size, char *subFileListName, char **resBuffer)
*subFileListName, char **resBuffer) 15  4.2.3.15 filesystem_readbytesByFirstSector(unsigned short firstSector, int startByte, int length, char *buffer, FileSystem *fs) 15  4.2.3.16 filesystem_setBytes(char *fileListData, int length, char *filename, int newSize) 16  4.2.3.17 filesystem_setNextSector(unsigned short start, unsigned short next, FileSystem *fs) 16  4.2.3.18 filesystem_subList(FileSystem *fs, char *fileListData, int size, char *subListName) 16  4.2.3.19 filesystem_writebytesByFirstSector(unsigned short firstSector, int startByte, int length, char *buffer, FileSystem *fs) 17  4.2.3.20 filesystems_driveList(char *fullPath) 17
length, char *buffer, FileSystem *fs)
4.2.3.17 filesystem_setNextSector(unsigned short start, unsigned short next, FileSystem *fs)
*fs)
4.2.3.19 filesystem_writebytesByFirstSector(unsigned short firstSector, int startByte, int length, char *buffer, FileSystem *fs)
length, char *buffer, FileSystem *fs)
4.2.3.21 getFileSvstemBvDriveId(int id)
3-1-1-1-1
4.2.3.22 getFirstFileSystem()
3 kernel/kernel.c File Reference
4.3.1 Detailed Description
4.3.2 Function Documentation
4.3.2.1 main()
19

9

9

# **Chapter 1**

# **Class Index**

## 1.1 Class List

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

### FileSystem

	Structu (with m					•	_							•									•		•						_
	(WILLI II	ıuιι	ibie	; u	ΠV	es	) (	anc	J I	la	HC	nec	u	as	a	Ш	IKE	<del>2</del> U	IIS	ι					•						O
IDT_Des	criptor																														6
IDT_Entr	y																														6
Memory	Header																														6
Memorvr	nap en	trv																											 		7

2 Class Index

# Chapter 2

# File Index

## 2.1 File List

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

ernel/kernel.c	
This contains the kernel's entry point	- 1
ernel/ <b>memory.h</b>	?
ernel/interruptions/ <b>interruption.h</b>	?
ernel/interruptions/ <b>keyboard.h</b>	?
ernel/interruptions/ <b>timer.h</b>	?
ernel/IO/cmos.h	?
ernel/IO/ <b>console.h</b>	?
ernel/IO/ <b>floppy.h</b>	?
ernel/IO/fs.c	
Implementation for the filesystem functions described in fs.h	
ernel/IO/fs.h	
Contains all functions for filesystem handling	
ernel/IO/ <b>io.h</b>	?
ernel/IO/ <b>memorymap.h</b>	?
ernel/IO/ <b>print.h</b>	?
ernel/memory/ <b>memorymanager.h</b>	?
ernel/stdlib/string.h	?

File Index

## **Chapter 3**

## **Class Documentation**

### 3.1 FileSystem Struct Reference

Structure describing a FatOS filesystem. This allows for multiple filesystems to be supported (with multiple drives) and handled as a linked list.

```
#include <fs.h>
```

Collaboration diagram for FileSystem:

#### **Public Attributes**

- · int drive id
- char \* FATpointer
- struct FileSystem \* next
- char \* fileList
- int fileListSize

#### 3.1.1 Detailed Description

Structure describing a FatOS filesystem. This allows for multiple filesystems to be supported (with multiple drives) and handled as a linked list.

#### 3.1.2 Member Data Documentation

3.1.2.1 int FileSystem::drive\_id

The drive ID holding the fs

3.1.2.2 char\* FileSystem::FATpointer

pointer to the FAT for that filesystem

6 Class Documentation

3.1.2.3 char\* FileSystem::fileList

a pointer to the fileList loaded into memory

3.1.2.4 int FileSystem::fileListSize

Length in bytes of the root filelist

3.1.2.5 struct FileSystem\* FileSystem::next

allows for easy chaining of filesystems

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

kernel/IO/fs.h

### 3.2 IDT\_Descriptor Struct Reference

#### **Public Attributes**

- · unsigned short limit
- · unsigned int base

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

· kernel/interruptions/interruption.h

### 3.3 IDT\_Entry Struct Reference

#### **Public Attributes**

- unsigned short base\_lower
- · unsigned short segment\_selector
- unsigned char always\_0
- · unsigned char flags
- · unsigned short base\_higher

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

• kernel/interruptions/interruption.h

### 3.4 MemoryHeader Struct Reference

Collaboration diagram for MemoryHeader:

#### **Public Attributes**

- int length
- unsigned int flags
- struct MemoryHeader \* previous
- struct MemoryHeader \* next

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

• kernel/memory/memorymanager.h

## 3.5 Memorymap\_entry Struct Reference

#### **Public Attributes**

- unsigned long long base\_address
- unsigned long long length
- · unsigned int type

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

• kernel/IO/memorymap.h

8 Class Documentation

## **Chapter 4**

## **File Documentation**

#### 4.1 kernel/IO/fs.c File Reference

contains the implementation for the filesystem functions described in fs.h

```
#include "fs.h"
Include dependency graph for fs.c:
```

#### 4.2 kernel/IO/fs.h File Reference

Contains all functions for filesystem handling.

```
#include "floppy.h"
#include "string.h"
#include "memory.h"
```

Include dependency graph for fs.h: This graph shows which files directly or indirectly include this file:

#### Classes

struct FileSystem

Structure describing a FatOS filesystem. This allows for multiple filesystems to be supported (with multiple drives) and handled as a linked list.

#### **Macros**

```
    #define FAT_SECTORS 10
        the numbers of sectors in the FAT
    #define FAT_FIRST 1
        the id of the sector at which FAT is located (in LBA)
    #define FAT_ENTRYAT(fatPtr, id) (*((unsigned short*)((int) fatPtr + 2*id)))
```

returns a pointer to a FAT entry of given FAT and entry id

#### **Typedefs**

• typedef struct FileSystem FileSystem

#### **Functions**

- unsigned short filesystem\_getNextSector (unsigned short sector, FileSystem \*fs)

Gets sector after a given sector in the FAT.

unsigned short filesystem\_getNextSectorRaw (unsigned short sector, FileSystem \*fs)

Gets sector after a given sector in the FAT, without striping the 'used' flag (highest bit)

unsigned short filesystem\_getNthSector (unsigned short firstSector, int n, FileSystem \*fs)

gets the Nth sector after a given sector in FAT. Indexing starts at 0.

unsigned short filesystem getLastSector (unsigned short firstSector, FileSystem \*fs)

gets the last sector of a chain of sectors in the FAT

• void filesystems init ()

initializes all filesystems accessible (one per detected floppy drive). SO FAR: ONLY boot filesystem

void filesystem init (FileSystem \*fs)

init file system (especially load file table at the given address)

unsigned short filesystem\_getFileSectors (unsigned short firstSector, FileSystem \*fs)

computes number of sector a file actual uses (based on FAT, not on advertised size). Based on the first sector. Reads it from FAT loaded into memory.

Reads the given amount of bytes from file given at first sector into provided buffer.

int filesystem\_writebytesByFirstSector (unsigned short firstSector, int startByte, int length, char \*buffer, File←
 System \*fs)

Writes the given amount of bytes from given buffer at sector indicated returns -1 upon error, 0 upon success.

FileSystem \* getFirstFileSystem ()

Gets the first filesystem handled by the filesystem handler (boot filesystem)

void filesystem\_LoadFileList (FileSystem \*fs)

Loads the filetable for a given filesystem.

FileSystem \* getFileSystemByDriveId (int id)

Returns a pointer to the filesystem of given drive ID.

void filesystem\_list (char \*fileListData, int length)

Prints the content of a given fileList (loaded at given position in memory, with given size)

• unsigned short filesystem\_findFirstSector (char \*filename, char \*filelist, int size)

Find first sector of entry of specified name in table loaded in buffer, of given size.

void filesystem setNextSector (unsigned short start, unsigned short next, FileSystem \*fs)

Sets the next sector in the FAT loaded in memory (does not rewrite it in drive)

unsigned short filesystem\_findEmptySector (unsigned short first, FileSystem \*fs)

This finds an empty sector to write to, after a given first one (initial value, set to 0 if all drive)

• int filesystem\_appendSectors (unsigned short first, int count, int rewrite, FileSystem \*fs)

This adds sectors at the end of chained list of sectors in FAT. Rewrites FAT into drive if specified so.

- int filesystem\_appendBytes (FileSystem \*fs, char \*fileListData, int size, char \*filename, int bytesToAppend)

  Appends bytes to the selected file.
- int filesystem getBytes (char \*fileListData, int length, char \*filename)

Returns the number of bytes of a file in a given filelist.

• int filesystem\_setBytes (char \*fileListData, int length, char \*filename, int newSize)

sets the number of bytes of file in a given filelist.

 int filesystem\_loadSubFileList (FileSystem \*fs, char \*fileListData, int size, char \*subFileListName, char \*\*resBuffer) loads a subfilelist into an allocated buffer. This buffer needs to be freed after using sys\_free.

• int filesystem\_subList (FileSystem \*fs, char \*fileListData, int size, char \*subListName)

Lists through a path ex. /subdir/ recursively until destination reached.

• int filesystems\_driveList (char \*fullPath)

Lists the contents of a given directory. End user function, supports complete path, including drive id. Supports 1-digits drives ids (0-9)

#### **Variables**

struct FileSystem \_\_attribute\_\_

#### 4.2.1 Detailed Description

Contains all functions for filesystem handling.

**Author** 

Anton CLAES

Date

2017

#### 4.2.2 **Macro Definition Documentation**

4.2.2.1 #define FAT\_ENTRYAT( fatPtr, id ) (\*((unsigned short\*)((int) fatPtr + 2\*id)))

returns a pointer to a FAT entry of given FAT and entry id

#### **Parameters**

	fatPtr	a pointer to the FAT in memory
ſ	id	the id of the entry (indexed start at 0)

#### 4.2.3 Function Documentation

4.2.3.1 int filesystem\_appendBytes ( FileSystem \* fs, char \* fileListData, int size, char \* filename, int bytesToAppend )

Appends bytes to the selected file.

#### **Parameters**

the filesystem on which to operate changes
the filelist on which to operate changes
the length in bytes of the filelist
the name of the files to add bytes to
the amount of bytes to be appened modifies FAT and filelist where the file is identified

#### **TODO HERE**

4.2.3.2 int filesystem\_appendSectors (unsigned short first, int count, int rewrite, FileSystem \* fs )

This adds sectors at the end of chained list of sectors in FAT. Rewrites FAT into drive if specified so.

#### **Parameters**

first	a sector in a chain at the end of which to add sectors
count	the number of sectors to add
rewrite	: 1 to rewrite FAT to disk, 0 to just update memory
fs	the filesystem on which to operate changes

#### Returns

0 upon success

4.2.3.3 unsigned short filesystem\_findEmptySector ( unsigned short first, FileSystem \* fs )

This finds an empty sector to write to, after a given first one (initial value, set to 0 if all drive)

#### **Parameters**

ſ	first	to first sector after which to look for an empty one
ſ	fs	the filesystem on which to perform operations

#### Returns

the sector found (0 upon failure)

4.2.3.4 unsigned short filesystem\_findFirstSector ( char \* filename, char \* filelist, int size )

Find first sector of entry of specified name in table loaded in buffer, of given size.

#### **Parameters**

filename	a string representing the filename in the filelist
filelist	the filelist loaded in memory
size	the size of the filelist

#### Returns

the first sector of the file as advertised in the filelist

4.2.3.5 int filesystem\_getBytes ( char \* fileListData, int length, char \* filename )

Returns the number of bytes of a file in a given filelist.

#### **Parameters**

fileListData	the filelist into which to look for the file						
length	the length of the filelist						
filename	the name of the file						

#### Returns

- -1 if not found, the number of bytes of the files as indicated by the filelist otherwise
- 4.2.3.6 unsigned short filesystem\_getFileSectors ( unsigned short firstSector, FileSystem \* fs )

computes number of sector a file actual uses (based on FAT, not on advertised size). Based on the first sector. Reads it from FAT loaded into memory.

#### **Parameters**

firstector	the firstsector of the file. If the sector is not the first one it will return the number of sectors until the
	end of the file
fs	the filesystem to read from

#### Returns

the id of the last sector of the file

4.2.3.7 unsigned short filesystem\_getLastSector ( unsigned short firstSector, FileSystem \* fs )

gets the last sector of a chain of sectors in the FAT

#### Parameters

first	sector any sector in the chain to start with
fs	the filesystem to get the FAT from

#### Returns

the id of the last sector in the chain

4.2.3.8 unsigned short filesystem\_getNextSector (unsigned short sector, FileSystem \* fs )

Gets sector after a given sector in the FAT.

#### **Parameters**

sector	the current sector
fs	the filesystem to get the FAT from

#### Returns

the next sector's id in LBA, 0 if no next sector

4.2.3.9 unsigned short filesystem\_getNextSectorRaw (unsigned short sector, FileSystem \* fs )

Gets sector after a given sector in the FAT, without striping the 'used' flag (highest bit)

#### **Parameters**

sector	the current sector
fs	the filesystem to get the FAT from

#### Returns

the next sector's id in LBA, 0 if no next sector, with the used flag not striped

4.2.3.10 unsigned short filesystem\_getNthSector (unsigned short firstSector, int n, FileSystem \* fs )

gets the Nth sector after a given sector in FAT. Indexing starts at 0.

#### **Parameters**

firstSector	the firstSector to seek from (not necessarely the first sector of the file)
n	the number of sectors to skip (n=0 returns the firstSector)
fs	the filesystem to get the FAT from

#### Returns

the id of the Nth sector

4.2.3.11 void filesystem\_init ( FileSystem \* fs )

init file system (especially load file table at the given address)

#### **Parameters**

fs the filesystem to initialize
---------------------------------

4.2.3.12 void filesystem\_list ( char \* fileListData, int length )

Prints the content of a given fileList (loaded at given position in memory, with given size)

#### **Parameters**

fileListData	the filelist (loaded in memory)
length	the length of the filelist

4.2.3.13 void filesystem\_LoadFileList ( FileSystem \* fs )

Loads the filetable for a given filesystem.

#### **Parameters**

4.2.3.14 int filesystem\_loadSubFileList ( FileSystem \* fs, char \* fileListData, int size, char \* subFileListName, char \*\* resBuffer )

loads a subfilelist into an allocated buffer. This buffer needs to be freed after using sys\_free.

#### **Parameters**

fs	the filesystem from which to load the sub filelist
fileListData	the current filelist into which to look for the subFileList (works recursively)
size	the size of the current filelist in bytes
subFileListName	the name of the subFileList(directory to look for)
resBuffer	that will be allocated and into which the subfilelist will be loaded into

#### Returns

the size of the loaded subfilelist in bytes, -1 if failed

4.2.3.15 int filesystem\_readbytesByFirstSector ( unsigned short firstSector, int startByte, int length, char \* buffer, FileSystem \* fs )

Reads the given amount of bytes from file given at first sector into provided buffer.

#### **Parameters**

firstSector	the firstSector of the file
startByte	the firstByte to read
length	the number of bytes to read
а	buffer to read into
fs	the filesystem to read from

#### Returns

-1 upon error, 0 upon success

4.2.3.16 int filesystem\_setBytes ( char \* fileListData, int length, char \* filename, int newSize )

sets the number of bytes of file in a given filelist.

#### **Parameters**

fileListData	the filelist into which to look for the file
length	the length of the filelist
filename	the name of the file
newSize	the size to be set to the file

#### Returns

-1 if not found

4.2.3.17 void filesystem\_setNextSector ( unsigned short start, unsigned short next, FileSystem \* fs )

Sets the next sector in the FAT loaded in memory (does not rewrite it in drive)

#### **Parameters**

start	the sector to set the next one to
next	the sector to be written as following the "start" one
fs	the filesystem on whose FAT to operate changes on

4.2.3.18 int filesystem\_subList ( FileSystem \* fs, char \* fileListData, int size, char \* subListName )

Lists through a path ex. /subdir/ recursively until destination reached.

#### **Parameters**

fs	the filesystem into which to look for the sublist
fileListData	the current filelist
size	the size of the current filelist
subListName	the name of the sublist to look for

#### Returns

-1 upon failure, 0 upon success

4.2.3.19 int filesystem\_writebytesByFirstSector ( unsigned short *firstSector*, int *startByte*, int *length*, char \* *buffer*, FileSystem \* *fs* )

Writes the given amount of bytes from given buffer at sector indicated returns -1 upon error, 0 upon success.

- · firstSector the first sector of the file
- · startByte the first byte to write too
- · length the number of bytes to write
- · fs the filesystem to write into

If the required data writes beyond file limit then the function will fail. It will not append sectors to the file.

#### **Parameters**

firstSector	the firstSector of the file	
startByte	the firstByte to write	
length	the number of bytes to write	
а	buffer to read from	
fs	the filesystem to write to	

#### Returns

-1 upon error, 0 upon success

4.2.3.20 int filesystems\_driveList ( char \* fullPath )

Lists the contents of a given directory. End user function, supports complete path, including drive id. Supports 1-digits drives ids (0-9)

#### **Parameters**

fullPath	the absolute path to the directory

#### Returns

-1 upon failure

4.2.3.21 FileSystem\* getFileSystemByDriveld ( int id )

Returns a pointer to the filesystem of given drive ID.

#### **Parameters**

id	the drive's id
iu	tile dilve 3 id

```
Returns
```

a pointer to the filesystem's structure

```
4.2.3.22 FileSystem* getFirstFileSystem()
```

Gets the first filesystem handled by the filesystem handler (boot filesystem)

Returns

a pointer a filesystem structure representing the first filesystem

#### 4.3 kernel/kernel.c File Reference

This contains the kernel's entry point.

```
#include "IO/print.h"
#include "memory.h"
#include "interruption.h"
#include "memorymanager.h"
#include "fs.h"
```

Include dependency graph for kernel.c:

#### **Functions**

```
• int main ()

OS kernel entry point.
```

#### 4.3.1 Detailed Description

This contains the kernel's entry point.

Author

Anton Claes

Date

2017

#### 4.3.2 Function Documentation

```
4.3.2.1 int main ( )
```

OS kernel entry point.

Returns

0 after console exited

# Index

drive_id	fs.h, 16
FileSystem, 5	filesystem_subList
	fs.h, 16
FAT_ENTRYAT	filesystem_writebytesByFirstSector
fs.h, 11	fs.h, 16
FATpointer	filesystems_driveList
FileSystem, 5	fs.h, 17
fileList	fs.h
FileSystem, 5	FAT_ENTRYAT, 11
fileListSize	filesystem_LoadFileList, 15
FileSystem, 6	filesystem_appendBytes, 11
FileSystem, 5	filesystem_appendSectors, 12
drive_id, 5	filesystem_findEmptySector, 12
FATpointer, 5	filesystem_findFirstSector, 12
fileList, 5	filesystem_getBytes, 12
fileListSize, 6	filesystem_getFileSectors, 13
next, 6	filesystem_getLastSector, 13
filesystem_LoadFileList	filesystem getNextSector, 13
fs.h, 15	filesystem_getNextSectorRaw, 14
filesystem_appendBytes	filesystem_getNthSector, 14
fs.h, 11	filesystem_init, 14
filesystem_appendSectors	filesystem_list, 14
fs.h, 12	filesystem_loadSubFileList, 15
filesystem_findEmptySector	filesystem_readbytesByFirstSector, 15
fs.h, 12	filesystem_setBytes, 16
filesystem_findFirstSector	filesystem_setNextSector, 16
fs.h, 12	filesystem_subList, 16
filesystem_getBytes	filesystem_writebytesByFirstSector, 16
fs.h, 12	filesystems_driveList, 17
filesystem_getFileSectors	getFileSystemByDriveId, 17
fs.h, 13	getFirstFileSystem, 18
filesystem_getLastSector	gen iisti ile System, 10
fs.h, 13	getFileSystemByDriveId
filesystem_getNextSector	fs.h, 17
fs.h, 13	getFirstFileSystem
filesystem_getNextSectorRaw	fs.h, 18
fs.h, 14	13.11, 10
filesystem_getNthSector	IDT_Descriptor, 6
fs.h, 14	IDT_Entry, 6
filesystem_init	ib i_Linuy, o
fs.h, 14	kernel.c
filesystem list	main, 18
fs.h, 14	kernel/IO/fs.c, 9
filesystem_loadSubFileList	kernel/IO/fs.h, 9
fs.h, 15	kernel/kernel.c, 18
filesystem_readbytesByFirstSector	Remembered, 10
fs.h, 15	main
filesystem_setBytes	kernel.c, 18
fs.h, 16	MemoryHeader, 6
filesystem_setNextSector	Memorymap entry, 7

20 INDEX

next

FileSystem, 6