File Manager

Generated by Doxygen 1.8.16

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

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

Chapter 3

Data Structure Documentation

3.1 DIR_ Struct Reference

Data Fields

- int **fd**
- int **size**
- int **idx**
- struct dirent * **files**

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

- include/dirent.h

3.2 dirent Struct Reference

Data Fields

- uint32_t **d_ino**
- int **d_type**
- char **d_name** [256]

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

- include/dirent.h

3.3 fs_block Union Reference

union of a block structure

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

Data Fields

- struct fs_super_block super
- struct fs_inode inodes [FS_INODES_PER_BLOCK]
- uint32_t pointers [FS_POINTERS_PER_BLOCK]
- uint8_t data [FS_BLOCK_SIZE]

3.3.1 Detailed Description

union of a block structure

a block can either be a super block, or and array of inodes or an array of pointers to other blocks, or an array of data bytes.

3.3.2 Field Documentation

3.3.2.1 data

```
uint8_t fs_block::data[FS_BLOCK_SIZE]
```

array of data bytes

3.3.2.2 inodes

```
struct fs_inode fs_block::inodes[FS_INODES_PER_BLOCK]
```

array of inodes

3.3.2.3 pointers

```
uint32_t fs_block::pointers[FS_POINTERS_PER_BLOCK]
```

array of pointers

3.3.2.4 super

```
struct fs_super_block fs_block::super
```

super block

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

- include/fs.h

3.4 fs_filesyst Struct Reference

virtual filesystem structure

`#include <disk.h>`

Data Fields

- uint32_t fd
- uint32_t tot_size
- uint32_t nblocks

3.4.1 Detailed Description

virtual filesystem structure

contains information about the file used to simulate a disk partition

3.4.2 Field Documentation

3.4.2.1 fd

`uint32_t fs_filesyst::fd`

file descriptor

3.4.2.2 nblocks

`uint32_t fs_filesyst::nblocks`

number of blocks in disk image

3.4.2.3 tot_size

`uint32_t fs_filesyst::tot_size`

total size of our file (partition)

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

- include/disk.h

3.5 fs_inode Struct Reference

inode structure

`#include <fs.h>`

Data Fields

- uint16_t mode
- uint16_t uid
- uint16_t gid
- uint32_t atime
- uint32_t mtime
- uint32_t size
- uint32_t **hcount**
- uint32_t direct [FS_DIRECT_POINTERS_PER_INODE]
- uint32_t indirect

3.5.1 Detailed Description

inode structure

the structure of inodes contains information about one file with a total size of 54 bytes.

3.5.2 Field Documentation

3.5.2.1 atime

`uint32_t fs_inode::atime`

last access time in seconds since the epoch

3.5.2.2 direct

`uint32_t fs_inode::direct[FS_DIRECT_POINTERS_PER_INODE]`

direct data blocks

3.5.2.3 gid

`uint16_t fs_inode::gid`

group id of owners

3.5.2.4 indirect

```
uint32_t fs_inode::indirect
```

indirect data blocks

3.5.2.5 mode

```
uint16_t fs_inode::mode
```

file type and permissions

3.5.2.6 mtime

```
uint32_t fs_inode::mtime
```

last modification time in seconds since the epoch

3.5.2.7 size

```
uint32_t fs_inode::size
```

size of the file in bytes

3.5.2.8 uid

```
uint16_t fs_inode::uid
```

id of owner

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

- include/fs.h

3.6 fs_super_block Struct Reference

super block structure

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

Data Fields

- uint32_t magic
- uint32_t data_bitmap_loc
- uint32_t data_bitmap_size
- uint32_t inode_bitmap_loc
- uint32_t inode_bitmap_size
- uint32_t inode_loc
- uint32_t inode_count
- uint32_t data_loc
- uint32_t data_count
- uint32_t free_inode_count
- uint32_t free_data_count
- uint32_t nreads
- uint32_t nwrites
- uint32_t mounts
- uint32_t mtime
- uint32_t wtime

3.6.1 Detailed Description

super block structure

the structure of the super block the first block stored stored in memory contains general information about the filesystem and other useful information, with a total size of 40 bytes.

3.6.2 Field Documentation

3.6.2.1 data_bitmap_loc

```
uint32_t fs_super_block::data_bitmap_loc
```

data bitmap location in block num

3.6.2.2 data_bitmap_size

```
uint32_t fs_super_block::data_bitmap_size
```

data bitmap size in blocks

3.6.2.3 data_count

```
uint32_t fs_super_block::data_count
```

no of data blocks

3.6.2.4 data_loc

```
uint32_t fs_super_block::data_loc
```

location of the data in blocks

3.6.2.5 free_data_count

```
uint32_t fs_super_block::free_data_count
```

no of free blocks

3.6.2.6 free_inode_count

```
uint32_t fs_super_block::free_inode_count
```

no of free inodes

3.6.2.7 inode_bitmap_loc

```
uint32_t fs_super_block::inode_bitmap_loc
```

inode bitmap location in block num

3.6.2.8 inode_bitmap_size

```
uint32_t fs_super_block::inode_bitmap_size
```

inode bitmap size in blocks

3.6.2.9 inode_count

```
uint32_t fs_super_block::inode_count
```

no of inodes in blocks

3.6.2.10 inode_loc

```
uint32_t fs_super_block::inode_loc
```

location of inodes in block num

3.6.2.11 magic

```
uint32_t fs_super_block::magic
```

the filesystem magic number

3.6.2.12 mounts

```
uint32_t fs_super_block::mounts
```

number of mounts

3.6.2.13 mtime

```
uint32_t fs_super_block::mtime
```

time of mount of the filesystem

3.6.2.14 nreads

```
uint32_t fs_super_block::nreads
```

number of reads performed

3.6.2.15 nwrites

```
uint32_t fs_super_block::nwrites
```

number of writes performed

3.6.2.16 wtime

```
uint32_t fs_super_block::wtime
```

last write time

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

- include/fs.h

3.7 io_filedesc Struct Reference

Data Fields

- int **is_allocated**
- uint32_t **offset**
- uint32_t **mode**
- uint32_t **inodenum**

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

- include/io.h

3.8 io_filedesc_table Struct Reference

Data Fields

- struct io_filedesc **fds** [IO_MAX_FILEDESC]

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

- include/io.h

Chapter 4

File Documentation

4.1 include/dirent.h File Reference

dirent.h - format of directory entries

Data Structures

- struct dirent
- struct DIR_

Macros

- #define **S_DIR** 01000

Functions

- int formatdir (struct fs_filesyst fs, struct fs_super_block super, uint32_t *inodenum, uint16_t mode)
  format and empty directory
- int insertFile (struct fs_filesyst fs, struct fs_super_block super, uint32_t dirino, struct dirent file)
  insert a file into a directory
- int findFile (struct fs_filesyst fs, struct fs_super_block super, uint32_t dirino, char *filename, struct dirent *res, int *idx)
  find a filename in a directory
- int getFiles (struct fs_filesyst fs, struct fs_super_block super, uint32_t dirino, struct dirent **files, int *size)
  get the files in a directory with inode inodenum
- int delFile (struct fs_filesyst fs, struct fs_super_block super, uint32_t dirino, char *filename)
  delete a file from a directory
- int findpath (struct fs_filesyst fs, struct fs_super_block super, uint32_t *ino, char *filename)
  find the inode number of a file from its absolute path
- int opendir_creat (struct fs_filesyst fs, struct fs_super_block super, uint32_t *dirino, uint16_t mode, const char *filepath)
  creates and formats a directory
- int opendir_ino (struct fs_filesyst fs, struct fs_super_block super, uint32_t dirino, const char *filepath)
  structures the a directory using the given filepath
- int open_ino (struct fs_filesyst fs, struct fs_super_block super, uint32_t fileino, const char *filepath)
  inserts a file into a directory
- int open_creat (struct fs_filesyst fs, struct fs_super_block super, uint32_t *fileino, uint16_t mode, const char *filepath)
  creates a new file

4.1.1 Detailed Description

dirent.h - format of directory entries

Author

ABDELMOUMENE Djahid
AYAD Ishak

main filesystem structs and prototypes

4.1.2 Function Documentation

4.1.2.1 delFile()

```
int delFile (
            struct fs_filesyst fs,
            struct fs_super_block super,
            uint32_t dirino,
            char * filename )
```

delete a file from a directory

deletes the file with filename *filename* into the directory with inode number *dirino*. the deletion is also done as in a sorted list.

4.1.2.2 findFile()

```
int findFile (
            struct fs_filesyst fs,
            struct fs_super_block super,
            uint32_t dirino,
            char * filename,
            struct dirent * res,
            int * idx )
```

find a filename in a directory

gets the structure found in a directory of the corresponding file with name *filename* in directory with inode number *dirino*. this function uses a binary search because the file entries are sorted in the directory

4.1.2.3 findpath()

```
int findpath (
            struct fs_filesyst fs,
            struct fs_super_block super,
            uint32_t * ino,
            char * filename )
```

find the inode number of a file from its absolute path

searches for the inode number of file with the absolute path *filename* and puts the value found into pointer *ino*. note that the root directory "/" is a special case and always has inode number 0

4.1.2.4 formatdir()

```
int formatdir (
            struct fs_filesyst fs,
            struct fs_super_block super,
            uint32_t * inodenum,
            uint16_t mode )
```

format and empty directory

allocate the inode for the directory and initialize the size (to 0) in the first byte

4.1.2.5 getFiles()

```
int getFiles (
            struct fs_filesyst fs,
            struct fs_super_block super,
            uint32_t dirino,
            struct dirent ** files,
            int * size )
```

get the files in a directory with inode *inodenum*

allocates an array of struct dirent's and puts the files and the number of files in files and size respectively

4.1.2.6 insertFile()

```
int insertFile (
            struct fs_filesyst fs,
            struct fs_super_block super,
            uint32_t dirino,
            struct dirent file )
```

insert a file into a directory

inserts the file structure *file* into the corresponding directory with inode number *dirino*. the insertion is in a sorted list.

4.1.2.7 open_creat()

```
int open_creat (
            struct fs_filesyst fs,
            struct fs_super_block super,
            uint32_t * fileino,
            uint16_t mode,
            const char * filepath )
```

creates a new file

creates a new file with a new inode number and inserts its corresponding directory entry into the appropriate directory (meaning inserts to the parent directory)

4.1.2.8 open_ino()

```
int open_ino (
              struct fs_filesyst fs,
              struct fs_super_block super,
              uint32_t fileino,
              const char * filepath )
```

inserts a file into a directory

insert the file with inode number *fileino* with the path *filepath*, meaning it creates a new directory entry and puts it in the parent directory (we get this from the full path ex: /DIR/file the parent directory is /DIR)

Parameters

fileino	the inode of the file to be inserted
filepath	the full path of the file to be inserted

4.1.2.9 opendir_creat()

```
int opendir_creat (
              struct fs_filesyst fs,
              struct fs_super_block super,
              uint32_t * dirino,
              uint16_t perms,
              const char * filepath )
```

creates and formats a directory

creates a new directory with a new inode number and then calls *opendir_ino*

4.1.2.10 opendir_ino()

```
int opendir_ino (
              struct fs_filesyst fs,
              struct fs_super_block super,
              uint32_t dirino,
              const char * filepath )
```

structures the a directory using the given filepath

formats the main components of the directory. Meaning it creates the subdirectories . and .. and also create an instance of its directory entry in the parent directory. ex: with /DIR it puts . and .. in the /DIR directory and DIR in the / (root) directory

Parameters

dirino	the inode number of the directory to be inserted
filepath	the full path (absolute) of the directory to be inserted

4.2 include/disk.h File Reference

main functions for interacting with the os

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

Data Structures

- struct fs_filesyst
  virtual filesystem structure

Macros

- #define **FS_BLOCK_SIZE** 4096 /∗ block size in bytes ∗/

Functions

- int creatfile (const char ∗filename, size_t size, struct fs_filesyst ∗fs)
  creat a disk image for storing the disk data
- int disk_size (struct fs_filesyst fs)
  discover the number of blocks on the disk
- void disk_close (struct fs_filesyst ∗fs)
  release the file
- int fs_write_block (struct fs_filesyst fs, int blocknum, const void ∗blk, size_t blksize)
  write a chunk of data into a block
- int fs_read_block (struct fs_filesyst fs, int blocknum, void ∗blk)
  read a chunk of data from a filesystem

4.2.1 Detailed Description

main functions for interacting with the os

Author

ABDELMOUMENE Djahid
AYAD Ishak

constains structs and prototypes used to manipulate the virtual filesystem

4.2.2 Function Documentation

4.2.2.1 creatfile()

```
int creatfile (
            const char * filename,
            size_t size,
            struct fs_filesyst * fs )
```

creat a disk image for storing the disk data

if this function is called on a disk image that already exists, the function will retun -1, otherwise it will initialize the fs_filesyst struct

See also

fs_filesyst

Parameters

filename	partition name
n	the number of blocks
fs	virtual filesystem structure

4.2.2.2 disk_close()

```
void disk_close (
            struct fs_filesyst * fs )
```

release the file

this function close the file descriptor using the virtual filesystem structure

Parameters

fs	virtual filesystem structure

4.2.2.3 disk_size()

```
int disk_size (
            struct fs_filesyst fs )
```

discover the number of blocks on the disk

Parameters

fs	virtual filesystem structure

Returns

the virtual filesystem number of blocks

4.2.2.4 fs_read_block()

```
int fs_read_block (
            struct fs_filesyst fs,
            int blocknum,
            void * blk )
```

read a chunk of data from a filesystem

read a block of data blk from the filesystem fs from block number blocknum

4.2.2.5 fs_write_block()

```
int fs_write_block (
            struct fs_filesyst fs,
            int blocknum,
            const void * blk,
            size_t blksize )
```

write a chunk of data into a block

write a block of data blk of size blksize into the filesystem fs in block number blocknum

4.3 include/fs.h File Reference

filesystem function header

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

Data Structures

- struct fs_super_block

  super block structure
- struct fs_inode

  inode structure
- union fs_block

  union of a block structure

Macros

- #define **FS_MAGIC** 0xF0F03410 /∗ magic number for our filesystem ∗/
- #define **FS_POINTERS_PER_BLOCK** 1024 /∗ no of pointers (used by inodes) per block in bytes∗/
- #define **FS_INODES_PER_BLOCK** 64 /∗ no of inodes per block ∗/
- #define **FS_DIRECT_POINTERS_PER_INODE** 8 /∗ no of direct data pointers in each inode ∗/
- #define **FS_INODE_RATIO** 0.01 /∗ total ratio of inodes in the fs ∗/
- #define **FS_MAX_INODE_COUNT** (NO_BYTES_32 / (FS_BLOCK_SIZE ∗ FS_INODES_PER_BLOCK))

Functions

- int fs_format_super (struct fs_filesyst fs)

  format the superblock into the virtual filesystem
- int fs_dump_super (struct fs_filesyst fs)

  dump formatted content of the superblock
- int fs_format (struct fs_filesyst fs)

  format the filesystem
- int fs_alloc_inode (struct fs_filesyst fs, struct fs_super_block ∗super, uint32_t ∗inodenum)

  allocate an inode
- int **fs_read_inode** (struct fs_filesyst fs, struct fs_super_block super, uint32_t indno, struct fs_inode ∗inode)
- int **fs_dump_inode** (struct fs_filesyst fs, struct fs_super_block super, uint32_t inodenum)
- int **fs_alloc_data** (struct fs_filesyst fs, struct fs_super_block ∗super, uint32_t data[ ], size_t size)
- int **fs_write_inode** (struct fs_filesyst fs, struct fs_super_block super, uint32_t indno, struct fs_inode ∗inode)
- int **fs_free_inode** (struct fs_filesyst fs, struct fs_super_block ∗super, uint32_t inodenum)
- int **fs_free_data** (struct fs_filesyst fs, struct fs_super_block ∗super, uint32_t datanum)
- int **fs_is_data_allocated** (struct fs_filesyst fs, struct fs_super_block super, uint32_t datanum)
- int **fs_is_inode_allocated** (struct fs_filesyst fs, struct fs_super_block super, uint32_t inodenum)
- int **fs_write_data** (struct fs_filesyst fs, struct fs_super_block super, union fs_block ∗data, uint32_t ∗blknums, size_t size)
- int **fs_read_data** (struct fs_filesyst fs, struct fs_super_block super, union fs_block ∗data, uint32_t ∗blknums, size_t size)

4.3.1 Detailed Description

filesystem function header

Author

ABDELMOUMENE Djahid
AYAD Ishak

main filesystem structs and prototypes

4.3.2 Function Documentation

4.3.2.1 fs_alloc_inode()

```
int fs_alloc_inode (
            struct fs_filesyst fs,
            struct fs_super_block * super,
            uint32_t * inodenum )
```

allocate an inode

allocates the first free inode in the inode table

- fs: the virtual filesystem
- super: the superblock
- inodenum: the inode number allocated

4.3.2.2 fs_dump_super()

```
int fs_dump_super (
            struct fs_filesyst fs )
```

dump formatted content of the superblock

prints a human readable superblock from teh filesystem fs

4.3.2.3 fs_format()

```
int fs_format (
            struct fs_filesyst fs )
```

format the filesystem

formats the superblock and sets the bitmaps to 0

4.3.2.4 fs_format_super()

```
int fs_format_super (
            struct fs_filesyst fs )
```

format the superblock into the virtual filesystem

format and calculate the positions and sizes of each section of the filesystem (eg. bitmaps and inode and data blocks)

Returns

this functions returns -1 in case of error and 0 on success

4.4 src/dirent.c File Reference

main directory managment and naming functions

```
#include <io.h>
#include <fs.h>
#include <devutils.h>
#include <disk.h>
#include <dirent.h>
#include <libgen.h>
#include <string.h>
#include <stdint.h>
#include <stdio.h>
```

Functions

- int formatdir (struct fs_filesyst fs, struct fs_super_block super, uint32_t *inodenum, uint16_t mode)

  format and empty directory
- int getFiles (struct fs_filesyst fs, struct fs_super_block super, uint32_t dirino, struct dirent **files, int *size)

  get the files in a directory with inode inodenum
- int findFile (struct fs_filesyst fs, struct fs_super_block super, uint32_t dirino, char *filename, struct dirent *res, int *idx)

  find a filename in a directory
- int insertFile (struct fs_filesyst fs, struct fs_super_block super, uint32_t dirino, struct dirent file)

  insert a file into a directory
- int delFile (struct fs_filesyst fs, struct fs_super_block super, uint32_t dirino, char *filename)

  delete a file from a directory
- int findpath (struct fs_filesyst fs, struct fs_super_block super, uint32_t *ino, char *filename)

  find the inode number of a file from its absolute path
- int opendir_ino (struct fs_filesyst fs, struct fs_super_block super, uint32_t dirino, const char *filepath)

  structures the a directory using the given filepath
- int opendir_creat (struct fs_filesyst fs, struct fs_super_block super, uint32_t *dirino, uint16_t perms, const char *filepath)

  creates and formats a directory
- int open_ino (struct fs_filesyst fs, struct fs_super_block super, uint32_t fileino, const char *filepath)

  inserts a file into a directory
- int open_creat (struct fs_filesyst fs, struct fs_super_block super, uint32_t *fileino, uint16_t mode, const char *filepath)

  creates a new file

4.4.1 Detailed Description

main directory managment and naming functions

Author

ABDELMOUMENE Djahid
AYAD Ishak

4.4.2 Function Documentation

4.4.2.1 delFile()

```
int delFile (
            struct fs_filesyst fs,
            struct fs_super_block super,
            uint32_t dirino,
            char * filename )
```

delete a file from a directory

deletes the file with filename *filename* into the directory with inode number *dirino*. the deletion is also done as in a sorted list.

4.4.2.2 findFile()

```
int findFile (
            struct fs_filesyst fs,
            struct fs_super_block super,
            uint32_t dirino,
            char * filename,
            struct dirent * res,
            int * idx )
```

find a filename in a directory

gets the structure found in a directory of the corresponding file with name *filename* in directory with inode number *dirino*. this function uses a binary search because the file entries are sorted in the directory

4.4.2.3 findpath()

```
int findpath (
            struct fs_filesyst fs,
            struct fs_super_block super,
            uint32_t * ino,
            char * filename )
```

find the inode number of a file from its absolute path

searches for the inode number of file with the absolute path *filename* and puts the value found into pointer *ino*. note that the root directory "/" is a special case and always has inode number 0

4.4.2.4 formatdir()

```
int formatdir (
            struct fs_filesyst fs,
            struct fs_super_block super,
            uint32_t * inodenum,
            uint16_t mode )
```

format and empty directory

allocate the inode for the directory and initialize the size (to 0) in the first byte

4.4.2.5 getFiles()

```
int getFiles (
            struct fs_filesyst fs,
            struct fs_super_block super,
            uint32_t dirino,
            struct dirent ** files,
            int * size )
```

get the files in a directory with inode *inodenum*

allocates an array of struct dirent's and puts the files and the number of files in files and size respectively

4.4.2.6 insertFile()

```
int insertFile (
            struct fs_filesyst fs,
            struct fs_super_block super,
            uint32_t dirino,
            struct dirent file )
```

insert a file into a directory

inserts the file structure *file* into the corresponding directory with inode number *dirino*. the insertion is in a sorted list.

4.4.2.7 open_creat()

```
int open_creat (
            struct fs_filesyst fs,
            struct fs_super_block super,
            uint32_t * fileino,
            uint16_t mode,
            const char * filepath )
```

creates a new file

creates a new file with a new inode number and inserts its corresponding directory entry into the appropriate directory (meaning inserts to the parent directory)

4.4.2.8 open_ino()

```
int open_ino (
            struct fs_filesyst fs,
            struct fs_super_block super,
            uint32_t fileino,
            const char * filepath )
```

inserts a file into a directory

insert the file with inode number *fileino* with the path *filepath*, meaning it creates a new directory entry and puts it in the parent directory (we get this from the full path ex: /DIR/file the parent directory is /DIR)

Parameters

fileino	the inode of the file to be inserted
filepath	the full path of the file to be inserted

4.4.2.9 opendir_creat()

```
int opendir_creat (
             struct fs_filesyst fs,
             struct fs_super_block super,
             uint32_t * dirino,
             uint16_t perms,
             const char * filepath )
```

creates and formats a directory

creates a new directory with a new inode number and then calls *opendir_ino*

4.4.2.10 opendir_ino()

```
int opendir_ino (
             struct fs_filesyst fs,
             struct fs_super_block super,
             uint32_t dirino,
             const char * filepath )
```

structures the a directory using the given filepath

formats the main components of the directory. Meaning it creates the subdirectories . and .. and also create an instance of its directory entry in the parent directory. ex: with /DIR it puts . and .. in the /DIR directory and DIR in the / (root) directory

Parameters

dirino	the inode number of the directory to be inserted
filepath	the full path (absolute) of the directory to be inserted

4.5 src/disk.c File Reference

initializing the partition

```
#include <devutils.h>
#include <disk.h>
#include <fs.h>
#include <sys/types.h>
#include <fcntl.h>
```

```
#include <unistd.h>
#include <stdio.h>
```

Functions

- int creatfile (const char *filename, size_t size, struct fs_filesyst *fs)
  creat a disk image for storing the disk data
- int disk_size (struct fs_filesyst fs)
  discover the number of blocks on the disk
- void disk_close (struct fs_filesyst *fs)
  release the file
- int fs_write_block (struct fs_filesyst fs, int blocknum, const void *blk, size_t blksize)
  write a chunk of data into a block
- int fs_read_block (struct fs_filesyst fs, int blocknum, void *blk)
  read a chunk of data from a filesystem

4.5.1 Detailed Description

initializing the partition

Author

ABDELMOUMENE Djahid
AYAD Ishak

initializing the partition files and utility functions to interact with the os

4.5.2 Function Documentation

4.5.2.1 creatfile()

```
int creatfile (
            const char * filename,
            size_t size,
            struct fs_filesyst * fs )
```

creat a disk image for storing the disk data

if this function is called on a disk image that already exists, the function will retun -1, otherwise it will initialize the fs_filesyst struct

See also

fs_filesyst

Parameters

filename	partition name
n	the number of blocks
fs	virtual filesystem structure

4.5.2.2 disk_close()

```
void disk_close (
            struct fs_filesyst * fs )
```

release the file

this function close the file descriptor using the virtual filesystem structure

Parameters

fs	virtual filesystem structure

4.5.2.3 disk_size()

```
int disk_size (
            struct fs_filesyst fs )
```

discover the number of blocks on the disk

Parameters

fs	virtual filesystem structure

Returns

the virtual filesystem number of blocks

4.5.2.4 fs_read_block()

```
int fs_read_block (
            struct fs_filesyst fs,
            int blocknum,
            void * blk )
```

read a chunk of data from a filesystem

read a block of data blk from the filesystem fs from block number blocknum

4.5.2.5 fs_write_block()

```
int fs_write_block (
            struct fs_filesyst fs,
            int blocknum,
            const void * blk,
            size_t blksize )
```

write a chunk of data into a block

write a block of data blk of size blksize into the filesystem fs in block number blocknum

4.6 src/fs.c File Reference

main filesystem utilities

```
#include <devutils.h>
#include <fs.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
```

Functions

- int fs_format_super (struct fs_filesyst fs)

  format the superblock into the virtual filesystem
- int fs_dump_super (struct fs_filesyst fs)

  dump formatted content of the superblock
- int fs_format (struct fs_filesyst fs)

  format the filesystem
- int **fs_is_data_allocated** (struct fs_filesyst fs, struct fs_super_block super, uint32_t datanum)
- int **fs_is_inode_allocated** (struct fs_filesyst fs, struct fs_super_block super, uint32_t inodenum)
- int fs_alloc_inode (struct fs_filesyst fs, struct fs_super_block *super, uint32_t *inodenum)

  allocate an inode
- int **fs_write_inode** (struct fs_filesyst fs, struct fs_super_block super, uint32_t indno, struct fs_inode *inode)
- int **fs_read_inode** (struct fs_filesyst fs, struct fs_super_block super, uint32_t indno, struct fs_inode *inode)
- int **fs_dump_inode** (struct fs_filesyst fs, struct fs_super_block super, uint32_t inodenum)
- int **fs_alloc_data** (struct fs_filesyst fs, struct fs_super_block *super, uint32_t data[ ], size_t size)
- int **fs_free_inode** (struct fs_filesyst fs, struct fs_super_block *super, uint32_t inodenum)
- int **fs_free_data** (struct fs_filesyst fs, struct fs_super_block *super, uint32_t datanum)
- int **fs_write_data** (struct fs_filesyst fs, struct fs_super_block super, union fs_block *data, uint32_t *blknums, size_t size)
- int **fs_read_data** (struct fs_filesyst fs, struct fs_super_block super, union fs_block *data, uint32_t *blknums, size_t size)

4.6.1 Detailed Description

main filesystem utilities

Author

ABDELMOUMENE Djahid
AYAD Ishak

initializing the partition files and utility functions to interact with the os

4.6.2 Function Documentation

4.6.2.1 fs_alloc_inode()

```
int fs_alloc_inode (
            struct fs_filesyst fs,
            struct fs_super_block * super,
            uint32_t * inodenum )
```

allocate an inode

allocates the first free inode in the inode table

- fs: the virtual filesystem
- super: the superblock
- inodenum: the inode number allocated

4.6.2.2 fs_dump_super()

```
int fs_dump_super (
            struct fs_filesyst fs )
```

dump formatted content of the superblock

prints a human readable superblock from teh filesystem fs

4.6.2.3 fs_format()

```
int fs_format (
            struct fs_filesyst fs )
```

format the filesystem

formats the superblock and sets the bitmaps to 0

4.6.2.4 fs_format_super()

```
int fs_format_super (
            struct fs_filesyst fs )
```

format the superblock into the virtual filesystem

format and calculate the positions and sizes of each section of the filesystem (eg. bitmaps and inode and data blocks)

Returns

this functions returns -1 in case of error and 0 on success

4.7 src/io.c File Reference

filesystem input/output operations

```
#include <io.h>
#include <fs.h>
#include <devutils.h>
#include <disk.h>
#include <string.h>
#include <stdint.h>
#include <stdio.h>
```

Functions

- int io_open_fd (uint32_t inodenum)

  allocates a new file descriptor with an inodenum
- int io_close_fd (int fd)

  closes an already open file descriptor
- int io_iopen (struct fs_filesyst fs, struct fs_super_block super, uint32_t inodenum)

  opens a new file without creating a new inode
- int **io_open_creat** (struct fs_filesyst fs, struct fs_super_block super, uint16_t mode, uint32_t *inodenum)
- int io_open_creat_fd (struct fs_filesyst fs, struct fs_super_block super, uint16_t mode)

  creates a new file with a new inodenum
- int io_lazy_alloc (struct fs_filesyst fs, struct fs_super_block super, uint32_t inodenum, struct fs_inode *ind, size_t off, size_t size)

  allocates blocks based on off and size
- int io_lseek (struct fs_filesyst fs, struct fs_super_block super, int fd, size_t new_off)

  changes the current offset of the file descriptor
- int io_write_ino (struct fs_filesyst fs, struct fs_super_block super, uint32_t inodenum, void *data, uint32_t off, size_t size)

  writes data to an inode number
- int io_write (struct fs_filesyst fs, struct fs_super_block super, int fd, void *data, size_t size)

  writes data to a file descriptor
- int io_read_ino (struct fs_filesyst fs, struct fs_super_block super, uint32_t inodenum, void *data, uint32_t off, size_t size)

  read data from an inode number

- int io_read (struct fs_filesyst fs, struct fs_super_block super, int fd, void *data, size_t size)

  reads data from a file descriptor
- int io_rm_ino (struct fs_filesyst fs, struct fs_super_block super, uint32_t inodenum)

  removes all from inode number inodenum
- int io_rm (struct fs_filesyst fs, struct fs_super_block super, int fd)

  removes the inodenumber corresponding to the fd
- uint32_t io_getino (int fd)

  get the corresponding inode number from the file descriptor
- size_t io_getoff (int fd)

  get the corresponding offset from the file descriptor

Variables

- struct io_filedesc_table filedesc_table = {0}

4.7.1 Detailed Description

filesystem input/output operations

Author

ABDELMOUMENE Djahid
AYAD Ishak

contains the main functions to create, destroy, read and write to files, also contains a system of file descriptors.

4.7.2 Function Documentation

4.7.2.1 io_close_fd()

```
int io_close_fd (
            int fd )
```

closes an already open file descriptor

Returns

returns 0 in case of success, -1 if the fd was never allocated or invalid.

4.7.2.2 io_iopen()

```
int io_iopen (
            struct fs_filesyst fs,
            struct fs_super_block super,
            uint32_t inodenum )
```

opens a new file without creating a new inode

tries to open the corresponding *inodenum* from the inode table.

Returns

returns the fd of the now open file in case of success, or -1 in case of failure

4.7.2.3 io_lazy_alloc()

```
int io_lazy_alloc (
            struct fs_filesyst fs,
            struct fs_super_block super,
            uint32_t inodenum,
            struct fs_inode * ind,
            size_t off,
            size_t size )
```

allocates blocks based on *off* and *size*

a utility functions used by io_write to allocate the least possible amount of blocks based on the writing offset *off* and the writing size *size*.

4.7.2.4 io_lseek()

```
int io_lseek (
            struct fs_filesyst fs,
            struct fs_super_block super,
            int fd,
            size_t new_off )
```

changes the current offset of the file descriptor

changes the offset of the file descriptor *fd* to *new_off*

4.7.2.5 io_open_creat_fd()

```
int io_open_creat_fd (
            struct fs_filesyst fs,
            struct fs_super_block super,
            uint16_t mode )
```

creates a new file with a new inodenum

allocates an inode and opens a new file descriptor,

Returns

returns an fd of the created file in case of success, else it returns -1.

4.7.2.6 io_open_fd()

```
int io_open_fd (
            uint32_t inodenum )
```

allocates a new file descriptor with an inodenum

allocates a file descriptor with the inode number given in arguments.

Returns

returns the a file descriptor (>0) in case of success, else it returns -1.

4.7.2.7 io_read()

```
int io_read (
            struct fs_filesyst fs,
            struct fs_super_block super,
            int fd,
            void * data,
            size_t size )
```

reads data from a file descriptor

does the same thing as *io_read_ino* but for file descriptors

4.7.2.8 io_read_ino()

```
int io_read_ino (
            struct fs_filesyst fs,
            struct fs_super_block super,
            uint32_t inodenum,
            void * data,
            uint32_t off,
            size_t size )
```

read data from an inode number

reads data from the inode number *inodenum* and puts it in the pointer *data* with size *size* starting from the offset *off*

4.7.2.9 io_rm()

```
int io_rm (
            struct fs_filesyst fs,
            struct fs_super_block super,
            int fd )
```

removes the inodenumber corresponding to the fd

does the same thing as io_rm_ino but for file descriptors

4.7.2.10 io_rm_ino()

```
int io_rm_ino (
            struct fs_filesyst fs,
            struct fs_super_block super,
            uint32_t inodenum )
```

removes all from inode number inodenum

frees the inode *inodenum* along with all the data blocks used by it (direct and indirect)

4.7.2.11 io_write()

```
int io_write (
            struct fs_filesyst fs,
            struct fs_super_block super,
            int fd,
            void * data,
            size_t size )
```

writes data to a file descriptor

does the same thing as *io_write_ino* but for file descriptors

4.7.2.12 io_write_ino()

```
int io_write_ino (
            struct fs_filesyst fs,
            struct fs_super_block super,
            uint32_t inodenum,
            void * data,
            uint32_t off,
            size_t size )
```

writes data to an inode number

writes the data *data* with size *size* starting from the offset *off* into the inode number *inodenum*

4.7.3 Variable Documentation

4.7.3.1 filedesc_table

```
struct io_filedesc_table filedesc_table = {0}
```

The main file descriptor table that holds information about all currently open files.

4.8 src/main.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <getopt.h>
#include <fs.h>
```

Functions

- int main (int argc, char **argv)
  main function

4.8.1 Detailed Description

Author

ABDELMOUMENE Djahid
AYAD Ishak

4.8.2 Function Documentation

4.8.2.1 main()

```
int main (
            int argc,
            char ** argv )
```

main function

Author

ABDELMOUMENE Djahid
AYAD Ishak

Index