Yet Another File System

AUTHOR

Version 1.0

2013-12-01

Table of Contents

Data Structure Index

Data Structures

Here are the data structures with brief descriptions:

inode .....Error: Reference source not found
openfile .....Error: Reference source not found
super_block .....Error: Reference source not found

File Index

File List

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

blockio.h .....Error: Reference source not found
error.h .....Error: Reference source not found
i_node.h .....Error: Reference source not found
open_file_table.h .....Error: Reference source not found
sfs_close.h .....Error: Reference source not found
sfs_create.h .....Error: Reference source not found
sfs_delete.h .....Error: Reference source not found
sfs_getsize.h .....Error: Reference source not found
sfs_gettype.h .....Error: Reference source not found
sfs_initialize.h .....Error: Reference source not found
sfs_open.h .....Error: Reference source not found
sfs_read.h .....Error: Reference source not found
sfs_readdir.h .....Error: Reference source not found
sfs_write.h .....Error: Reference source not found
super_block.h .....Error: Reference source not found

Data Structure Documentation

inode Struct Reference

Data Fields

char * **name**
char * **i_number**
int **type**
char * **parent_i_number**
char * **file_size**
int **index_blk_location**

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

i_node.h

openfile Struct Reference

Data Fields

char * **pathname**
int **opened**
int **fd**

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

open_file_table.h

super_block Struct Reference

Data Fields

int **size**
int **blocksize**
int **free_blocks**
int **root**

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

super_block.h

File Documentation

error.h File Reference

Enumerations

enum **ERROR** { **SUCCESSFULLY_CREATED_FILE**, **NO_FILE_NAME_ENTERED**, **FULL_FILE**, **PUT_BLOCK_FAIL**, **BLK_ALLOCATED**, **FAIL_ALLOCATE**, **GET_BLOCK_FAIL**, **ERROR_WRITING_INODE_TO_DISK**, **FILE_OPENED_LIMIT_HAS_BEEN_REACHED**, **FILE_NOT_FOUND_IN_OPEN_TABLE**, **FILE_NOT_FOUND**, **FILE_WITH_NAME_EXISTS**, **FILE_IS_EMPTY**, **INVALID_FILE_NAME**, **FOUND_TYPE**, **WRITING_TO_DIR**, **READING_FROM_DIR**, **READING_BEYOND_FILE_SIZE**, **WRITING_BEYOND_FILE_SIZE**, **PATHNAME_ERROR**, **FILE_NAME_TOO_LONG**, **WRITE_TOO_MANY_CHARS**, **READDIR_REG_FILE**, **ERROR_PARENT_REG_FILE**, **ERROR_DIR_SIZE**, **ERROR_DELETE_FILE_IS_OPEN**, **ERROR_DELETE_DIR_HAS_CHILDREN**, **READDIR_FILE_NOT_OPEN** }

Functions

int **error** (ERROR error)

Basic error handling system that outputs an appropriate error by request.

Detailed Description

Error management tool Responds with an error output to predefined error codes.

Function Documentation

int error (ERROR *error*)

Basic error handling system that outputs an appropriate error by request.

This function will output an appropriate error according to the ERROR code provided

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

error	The error code to handle

-1 if it's an error
0 if it's success

Copyright:

GNU Public License.

i_node.h File Reference

Data Structures
struct **inode**

Macros

#define **I_NODE_H_**

Functions

int **add_new_inode** (**inode** *new_inode)
: Adds an inode to the inode_table and writes it to disk.

int **put_inode_table** (void)
Creates the initial inode_table and writes it to disk.

int **get_next_i_number** (char *i_number)
Gets the smallest available i_number.

int **get_inode_table_from_disk** (void)
Gets the inode table from the disk.

int **save_file_contents** (char *contents, char *name, char *parent)
Saves the file contents to disk.

int **get_file_contents** (char *name, char *parent, char *contents)
Retrieves the file contents from the disk.

int **get_size** (char *pathname)
Gets the size of the file.

int **get_i_number** (char *name, char *parent, char *i_number)
Returns the i_number of the file.

int **find_file** (char *name, char *parent)
Return true or false (0 or -1) of whether the file is found.

int **get_type** (char *name, char *parent, int *type)
Finds the appropriate file and saves its type to the pointer provided.

int **dir_get_children** (char *pathname, char *children)
Reads the inode_table and returns a directory's children.

int **save_inode_table** (void)
Saves the inode_table from memory into the disk.

int **delete_inode** (int i_num)
Removes the inode with the specified i_number from the inode_table.

Variables

short int **fd_table** [64]
int **i_numbers** [64]
inode inode_table [64]

Detailed Description

Connects the lower level functions with the higher level and deal with the inode table Contains most of the file functions regarding deleting, changing file information, writing, etc.

Macro Definition Documentation

#define I_NODE_H_

Include guard

Function Documentation

int add_new_inode (inode * *new_inode*)

: Adds an inode to the inode_table and writes it to disk.

: Inserts provided inode into the inode_table array and then rewrites the information back to the disk.

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

Returns:

**new_inode*	Reference to the inode to add into the table

0 indicating successful operation

Copyright:

GNU Public License.

int delete_inode (int *i_num*)

Removes the inode with the specified i_number from the inode_table.

Removes the appropriate inode from the inode_table and re-writes the disk

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Returns:

0 to indicate the operation is over.

Copyright:

GNU Public License.

int dir_get_children (char * *pathname*, char * *children*)

Reads the inode_table and returns a directory's children.

Looks up the file in the open_file_table by its fd, checks the file type. If it's a folder - outputs the file names of the files that have this directory as a parent.

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

Returns:

pathname	Pathname of the dir file to read
children	List of children to be returned

0 to indicate the operation is over.

Copyright:

GNU Public License.

int find_file (char * *name*, char * *parent*)

Return true or false (0 or -1) of whether the file is found.

Finds the correct file by using the name provided and the parent's i_number and returns success flag if the operation is successful

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

Returns:

**name*	Pointer to the name of the file to get the i_number of
**parent*	File's parent's i_number

0 indicating success
-1 indicating failure

Copyright:

GNU Public License.

int get_file_contents (char * *name*, char * *parent*, char * *contents*)

Retrieves the file contents from the disk.

This function gets the name of the file, parent's i_number, and pointer to contents to save the contents of the file to.

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

Returns:

**name*	Name of the file to get the contents of
**parent*	The file's parent's i_number
**contents*	Contents of the file to save to

0 if successful
-1 if error occurs

Copyright:
GNU Public License.

int get_i_number (char * *name*, char * *parent*, char * *i_number*)

Returns the i_number of the file.

Finds the correct file by using the name provided and the parent's i_number and returns the file's i_number

Author:
Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

Returns:

**name*	Pointer to the name of the file to get the i_number of
**parent*	File's parent's i_number
**i_number*	The reference to the file's i_number to save the information to

0 indicating success
-1 indicating failure

Copyright:
GNU Public License.

int get_inode_table_from_disk (void)

Gets the inode table from the disk.

Reads inode table from the disk and loads it into memory by creating new inode structs and putting them into the inode_table array

Author:
Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Returns:
0 indicating success

Copyright:
GNU Public License.

int get_next_i_number (char * *i_number*)

Gets the smallest available i_number.

Gets the smallest available i_number and marks is taken (1), and saves the value to the provided parameter making sure it has two digits.

Author:
Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

Returns:

**i_number*	Pointer to char to save the i_number to

0 success
-1 failure

Copyright:

GNU Public License.

int get_size (char * *pathname*)

Gets the size of the file.

Iterates through the path to get the size of the target file

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

Returns:

**pathname*	Pointer to the pathname of the file to get the size of

The file size
-1 indicating failure

Copyright:

GNU Public License.

int get_type (char * *name*, char * *parent*, int * *type*)

Finds the appropriate file and saves its type to the pointer provided.

Finds the correct file by using the name provided and the parent's i_number and saves the file's type to the provided pointer type

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

Returns:

**name*	Pointer to the name of the file to get the i_number of
**parent*	File's parent's i_number
**type*	The reference to the file's type to save the information to

0 indicating success
-1 indicating failure

Copyright:

GNU Public License.

int put_inode_table (void)

Creates the initial inode_table and writes it to disk.

Initiates the root inode and writes it to disk separating every parameter with a ":" delimiter and "_" indicating end of the inode

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Copyright:

GNU Public License.

int save_file_contents (char * *contents*, char * *name*, char * *parent*)

Saves the file contents to disk.

Saves the contents of the file by dynamically allocating new blocks on the disk if needed. Writes the contents to the disk and then write the index block table containing the blocks used up.

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

Returns:

**contents*	Contents of the file to save to disk
**name*	Name of the file to save
**parent*	File's parent's i_number

0 indicating successful operation

Copyright:

GNU Public License.

int save_inode_table (void)

Saves the inode_table from memory into the disk.

Converts the inode table into a string literal and stores it on to the disk blocks 2-10

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Returns:

0 to indicate the operation is over.

Copyright:

GNU Public License.

open_file_table.h File Reference

Data Structures
struct **openfile**

Macros

#define **OPEN_FILE_TABLE_H_**

Functions

int **add_opened_file** (char *pathname)
Adds a file to the open-file table.

int **close_file** (int fd)
Removes a file from the open-file table.

int **get_opened_file** (int fd, char *pathname)
Returns opened file's pathname.

int **get_opened_file_fd** (char *pathname)
Returns opened file's file descriptor.

Variables

openfile all_opened_files [64]
int **all_fd** [64]

Detailed Description

Open file table. Tracks of all opened files Creates and tracks of all open files in the open-file table with file descriptors

---

Macro Definition Documentation

#define OPEN_FILE_TABLE_H_

Include guard

Function Documentation

int add_opened_file (char * *pathname*)

Adds a file to the open-file table.

This function will add a new file specified by its pathname to the open-file table

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

Returns:

**pathname*	The absolute pathname of the file to add

The fd of the file added

Copyright:

GNU Public License.

int close_file (int *fd*)

Removes a file from the open-file table.

This function will remove the file specified by its fd from the open-file table

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

Returns:

fd	The fd of the file to close

0 for success -1 for failure

Copyright:

GNU Public License.

int get_opened_file (int *fd*, char * *pathname*)

Returns opened file's pathname.

This function will return the file's pathname

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

Returns:

fd	The fd of the file to search for
**pathname*	The char pointer where the files pathname will be saved to

0 for success -1 for failure

Copyright:

GNU Public License.

int get_opened_file_fd (char * *pathname*)

Returns opened file's file descriptor.

This function will return the file's file descriptor

Author:

Yasha Prikhodko
Cedric Leong

Arezou Mousavi

Parameters:

Returns:

**pathname*	The pathname of the file to get the file descriptor of

0 for success -1 for failure

Copyright:

GNU Public License.

sfs_close.h File Reference

Functions

int **sfs_close** (int fd)

File function to close files.

Detailed Description

File function to close a file Closes the file or its instance (if the file was opened multiple times)

Function Documentation

int sfs_close (int *fd*)

File function to close files.

This function will close a file (or its instance) in memory by its file descriptor

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

fd	The file descriptor of the file to close

Returns:

0 if success
-1 if failure

Copyright:

GNU Public License.

sfs_create.h File Reference

Macros
#define **SFS_CREATE_H_**

Functions

int **sfs_create** (char *pathname, int type)
File function to create files.

Detailed Description

Creates a directory or regular type file Creates a file along with its inode and write it to disk

---

Macro Definition Documentation

#define SFS_CREATE_H_

Include guard

Function Documentation

int sfs_create (char * *pathname*, int *type*)

File function to create files.

This function will create a new file along with its inode

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

Returns:

**pathname*	The absolute pathname of the file to be created
type	The type of the file to be created (0=regular file, 1=folder)

GNU Public License.

sfs_delete.h File Reference

Functions

int **sfs_delete** (char *pathname)

: Deletes the file from the disk

Detailed Description

File function to delete a file from disk Finds and deletes the file specified along with its contents and the inode.

Function Documentation

int sfs_delete (char * *pathname*)

: Deletes the file from the disk

: Makes sure the file is not opened and does not contain any children, then deletes its inode and frees up the blocks

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

Copyright:

**pathname*	Pointer pathname char type

0 if successful
-1 if failure

Copyright:

GNU Public License.

sfs_getsize.h File Reference

Macros
#define **SFS_GET_SIZE_H_**

Functions

int **sfs_getsize** (char *pathname)
: Gets the size of the file

Detailed Description

Returns the size of the file Looks up the file by goind down the hierarchy and gets its size from the inode_table

---

Macro Definition Documentation

#define SFS_GET_SIZE_H_

Include guard

Function Documentation

int sfs_getsize (char * *pathname*)

: Gets the size of the file

: Iterates through the pathname making sure each level exists, then finds the target file and returns its size.

Author:
Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

Returns:

**pathname*	Pointer pathname char type

Size of the file

Copyright:
GNU Public License.

sfs_gettype.h File Reference

Functions

int **sfs_gettype** (char *pathname)

: File function to check the type of the file

Detailed Description

Looks up the file and returns its type Gets the type of the file from the inode_table and returns 0 (REGULAR) or 1 (DIRECTORY).

Function Documentation

int sfs_gettype (char * *pathname*)

: File function to check the type of the file

: Iterates over the pathname making sure the path is existent. Then gets the type of the file and returns it

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

Returns:

**pathname*	The pathname of the file to get the type of

1 if directory
0 if regular file
-1 if failed to get the type

Copyright:

GNU Public License.

sfs_initialize.h File Reference

Macros
#define **SFS_INITIALIZE_H_**

Functions

int **sfs_initialize** (int erase)
: Initialize the free block list.
int **new_filesystem** (void)
: Create a null block of data

Detailed Description

File function to intialize the file system Initialized the file system with option 0 to load it from disk or option 1 to erase the disk

---

Macro Definition Documentation

#define SFS_INITIALIZE_H_

Include guard

Function Documentation

int new_filesystem (void)

: Create a null block of data

: Allocates 0's to the disk in order to override it

Author:
Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Returns:
1 if successful and -1 if failure

Copyright:
GNU Public License.

int sfs_initialize (int *erase*)

: Initialize the free block list.

: It creates a new file system, then creates a new super block with size 512 and initialize this free blocks list.

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

erase	Flag indicator to erase (1) or not (0)

Returns:

1 if successful and -1 if failure

Copyright:

GNU Public License.

sfs_open.h File Reference

Macros
#define **SFS_OPEN_H_**

Functions

int **sfs_open** (char *pathname)
File function to open files.

Detailed Description

File function to open a file and load its file descriptor into memory Finds and and opens the file by assigning a file descriptor in the open-file table.

Macro Definition Documentation

#define SFS_OPEN_H_

Include guard

Function Documentation

int sfs_open (char * *pathname*)

File function to open files.

This function will open a new file by iterating through the path using tokens making sure each level of hierarchy exists

Author:
Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

Returns:

**pathname*	The absolute pathname of the file to be opened

fd of the file opened

Copyright:
GNU Public License.

sfs_read.h File Reference

Macros
#define **SFS_READ_H_**

Functions

int **sfs_read** (int fd, int start, int length, char *buffer)
: Reads the contents of the file from the disk and outputs it

---

Detailed Description

Reads the contents of the file if its open and outputs it Checks if the file is open, then reads the contents of it off the disk and outputs it

---

Macro Definition Documentation

#define SFS_READ_H_

Include guard

Function Documentation

int sfs_read (int *fd*, int *start*, int *length*, char * *buffer*)

: Reads the contents of the file from the disk and outputs it

: Reads the file blocks at the given start position and an appropriate length and saves the result to buffer

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

Returns:

fd	File descriptor of the file in the open file table
start	Start location to read at in bytes
length	The length of the buffer to read (in bytes)
**buffe*	The pointer to where the content of the file will be saved to

0 if succedd
-1 if failure

Copyright:

GNU Public License.

sfs_readdir.h File Reference

Functions

int **sfs_readdir** (int fd)

: Reads and returns a directory's children

Detailed Description

Reads the contents of a file DIRECTORY type and outputs the result Reads the names of the children of a DIRECTORY type file and outputs the children REGULAR and DIRECTORY type files

---

Function Documentation

int sfs_readdir (int *fd*)

: Reads and returns a directory's children

: Looks up the file in the open_file_table by its fd, checks the file type. If it's a folder - outputs the file names of the files that have this directory as a parent.

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

fd	File descriptor of the folder to read the contents of.

Returns:

0 or -1 (success or failure) of the operation

Copyright:

GNU Public License.

sfs_write.h File Reference

Macros
#define **SFS_WRITE_H_**

Functions

int **sfs_write** (int fd, int start, int length, char *mem_pointer)
: Writes the buffer to the file and save it to the disc

Detailed Description

Writes a literal string to a file Writes a literal string to a file witha start location on the disk.

Macro Definition Documentation

#define SFS_WRITE_H_

Include guard

Function Documentation

int sfs_write (int *fd*, int *start*, int *length*, char * *mem_pointer*)

: Writes the buffer to the file and save it to the disc

: It checks the size before writing then writes and saves them to the disc while increasing the size if needed. Also in order to write the file need to be opened

Author:

Yasha Prikhodko
Cedric Leong
Arezou Mousavi

Parameters:

Returns:

fd	File descriptor of the file in the open file table
start	Start location to write at (-1) if the file is empty and/or to append to the end
length	The length of the buffer to write (in bytes)
**mem_pointer*	The input the user provided to write to the file

0 if succedd
-1 if failure

Copyright:

GNU Public License.

Index

INDEX