PotatOS

Generated by Doxygen 1.8.11

Contents

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 BOOTSECTORSTRUCT Struct Reference

Data Fields

- uint16_t **bytesPerSector**
- uint16_t **sectorsPerCluster**
- uint16_t **numReservedSectors**
- uint16_t **numFATCopies**
- uint16_t **maxNumRoot**
- uint16_t **numOfSectors**
- uint16_t **numSECTORSPERFAT**
- uint16_t **sectorsPerTrack**
- uint16_t **numberHeads**
- uint16_t **sectorCountFAT32**
- uint16_t **bootSig**
- uint32_t **volumeID**
- char * **volumeLabel**
- char * **fileSystemType**

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

- file_wrangler.h

3.2 command Struct Reference

A struct to hold command information.

`#include <command_utils.h>`

Data Fields

- char * **command_name**
- CommandFunction **function**
- char * **command_usage**
- char * **command_description**
- char * **command_examples**

3.2.1 Detailed Description

A struct to hold command information.

Parameters

command_name	The command's name
function	A pointer to the command's function
command_usage	Syntax for executing the command
command_description	A description of the command
command_examples	Example uses of the command

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

- command_utils.h

3.3 ENTRY Struct Reference

A struct to hold an directory entry.

`#include <command_utils.h>`

Data Fields

- BYTE **empty**
- char **fileName** [9]
- char **extension** [4]
- BYTE **attributes**
- uint16_t **reserved**
- uint16_t **creationTime**
- uint8_t **creationHour**
- uint8_t **creationMin**
- uint8_t **creationSec**
- uint16_t **creationDate**
- uint16_t **creationYear**
- uint8_t **creationMonth**
- uint8_t **creationDay**
- uint16_t **lastAccessDate**
- uint16_t **lastAccessYear**
- uint8_t **lastAccessMonth**
- uint8_t **lastAccessDay**
- uint16_t **lastWriteTime**
- uint8_t **lastWriteHour**
- uint8_t **lastWriteMin**
- uint8_t **lastWriteSec**
- uint16_t **lastWriteDate**
- uint16_t **lastWriteYear**
- uint8_t **lastWriteMonth**
- uint8_t **lastWriteDay**
- uint16_t **firstLogicalCluster**
- uint32_t **fileSize**

3.3.1 Detailed Description

A struct to hold an directory entry.

Parameters

empty	If the entry is empty
fileName	The name of the entry
extension	The extension, type, of the entry
attributes	The attributes of the entry
reserved	If the entry is resereved
creationTime	Entry creation time
creationHour	Entry creation hour
creationMin	Entry creation min
creationSec	Entry creation sec
creationDate	Entry creation date
creationYear	Entry creation year
creationMonth	Entry creation month
creationDay	Entry creation day
lastAccessDate	Entry access date
lastAccessYear	Entry access year
lastAccessMonth	Entry access month
lastAccessDay	Entry access day
lastWriteTime	Entry last write time
lastWriteHour	Entry last write hour
lastWriteMin	Entry last write min
lastWriteSec	Entry last write sec
lastWriteDate	Entry last write date
lastWriteYear	Entry last write year
lastWriteMonth	Entry last write month
lastWriteDay	Entry last write day
firstLogicalCluster	First logical cluster
fileSize	Entry filesize

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

- command_utils.h

3.4 PREVDIR Struct Reference

Data Fields

- char * **dirName**
- uint32_t **sectorStart**

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

- command_utils.h

Chapter 4

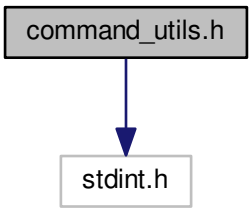
File Documentation

4.1 command_utils.h File Reference

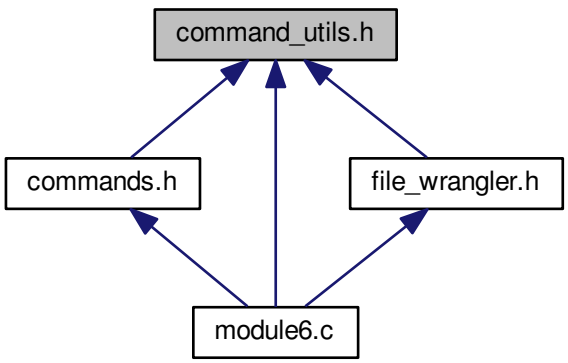
Holds standard typedefs and utility function headers for certain commands.

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

Include dependency graph for command_utils.h:



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



Data Structures

- struct command

  A struct to hold command information.
- struct ENTRY

  A struct to hold an directory entry.
- struct PREVDIR

Macros

- #define **SECTORSIZE** 512
- #define **SECTORSPERFAT** 9
- #define **FATTABLESIZE** ((SECTORSPERFAT ∗ SECTORSIZE ∗ 8) / 12)
- #define **MAXDIRENTRY** 16
- #define **MAXENTRIESPERDIR** 16
- #define **DEBLOC**() printf("\033[33;1m%s(%d)\033[0m\n",__FILE__,__LINE__)

Typedefs

- typedef unsigned char **BYTE**
- typedef int(∗ **CommandFunction**) (int, char ∗∗)
- typedef struct ENTRY ENTRY

  A struct to hold an directory entry.
- typedef struct PREVDIR **PREVDIR**
- typedef struct command **command_t**

Functions

- char ∗ trim_whitespace (char ∗str)

  A function to trim strings.
- uint32_t split_args (char ∗command, char ∗∗argv)

  A function to split up a input string into arguments for command functions.
- void memcpyUpper (char ∗, const char ∗, int)

  A memcpy function that takes all characters to their uppercase equivalent.
- int **strcimp** (char const ∗a, char const ∗b)
- uint32_t starsearch (const char ∗starryboi, const char ∗fileboi)

  Star search for ls.
- uint16_t **findDotPosition** (const char ∗)

4.1.1 Detailed Description

Holds standard typedefs and utility function headers for certain commands.

4.1.2 Typedef Documentation

4.1.2.1 typedef struct ENTRY ENTRY

A struct to hold an directory entry.

Parameters

empty	If the entry is empty
fileName	The name of the entry
extension	The extension, type, of the entry
attributes	The attributes of the entry
reserved	If the entry is resereved
creationTime	Entry creation time
creationHour	Entry creation hour
creationMin	Entry creation min
creationSec	Entry creation sec
creationDate	Entry creation date
creationYear	Entry creation year
creationMonth	Entry creation month
creationDay	Entry creation day
lastAccessDate	Entry access date
lastAccessYear	Entry access year
lastAccessMonth	Entry access month
lastAccessDay	Entry access day
lastWriteTime	Entry last write time
lastWriteHour	Entry last write hour
lastWriteMin	Entry last write min
lastWriteSec	Entry last write sec
lastWriteDate	Entry last write date
lastWriteYear	Entry last write year
lastWriteMonth	Entry last write month
lastWriteDay	Entry last write day
firstLogicalCluster	First logical cluster
fileSize	Entry filesize

4.1.3 Function Documentation

4.1.3.1 void memcpyUpper (char *, const char *, int)

A memcpy function that takes all characters to their uppercase equivalent.

Parameters

dest	The destination char pointer
source	The source char pointer
nchars	The number of chars to copy, and toUpper, from source to dest

Returns

void method, copies in place

4.1.3.2 uint32_t split_args (char * *command*, char ** *argv*)

A function to split up a input string into arguments for command functions.

Parameters

The	command input to split up
A	pointer to the destination of the argument slices, argv

Returns

The amount of arguments split up, argc

4.1.3.3 uint32_t starsearch (const char * *starryboi*, const char * *fileboi*)

Star search for ls.

Parameters

starryboi	The char pointer that contians a star search

4.1.3.4 char* trim_whitespace (char * *str*)

A function to trim strings.

Parameters

str	The string to trim

Returns

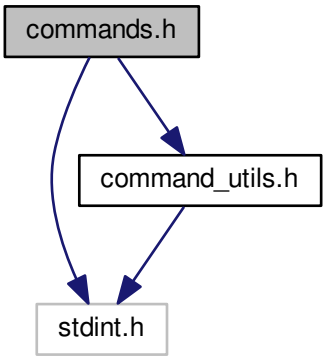
A pointer to the beginning of the trimmed string

4.2 commands.h File Reference

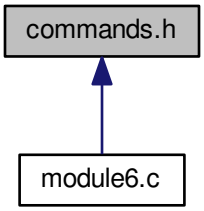
Holds all of the command prototypes.

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

Include dependency graph for commands.h:



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



Functions

- command_t * search_commands (char *command)
  Search commands for a given command.
- int help_command (int argc, char **argv)
  Command to get help about other commands.
- int pbsi_command (int argc, char **argv)
  print boot sector information
- int prd_command (int argc, char **argv)
  print root directory
- int root_command (int argc, char **argv)
  change directory to root directory
- int cd_command (int argc, char **argv)
  change directory to requested directory
- int ls_command (int argc, char **argv)

list directory

- int type_command (int argc, char **argv)

  prints out contents of file

- int rename_command (int argc, char **argv)

  renames a file or directory

- int move_command (int argc, char **argv)

  move a file to an adjacent directory

- int exit_command (int argc, char **argv)

  exit the m6 environement

4.2.1 Detailed Description

Holds all of the command prototypes.

4.2.2 Function Documentation

4.2.2.1 int cd_command (int *argc*, char ** *argv*)

change directory to requested directory

Parameters

argc	count of arguments passed in
argv	list of strings separated by space characters passed in

Returns

0

4.2.2.2 int exit_command (int *argc*, char ** *argv*)

exit the m6 environement

Parameters

argc	count of arguments passed in
argv	list of strings separated by space characters passed in

Returns

0

exit the m6 environement

Confirms user wants to exit, and saves the state of the image.

Parameters

argc	Number of arguments
argv	List of arguments. (None are applicable).

Returns

int Success condition.

4.2.2.3 int help_command (int *argc*, char ** *argv*)

Command to get help about other commands.

Parameters

argc	Number of arguments
argv	Pointer to arguments

Returns

exit status code

4.2.2.4 int ls_command (int *argc*, char ** *argv*)

list directory

Parameters

argc	count of arguments passed in
argv	list of strings separated by space characters passed in

Returns

0

4.2.2.5 int move_command (int *argc*, char ** *argv*)

move a file to an adjacent directory

Parameters

argc	count of arguments passed in
argv	list of strings separated by space characters passed in

Returns

0

move a file to an adjacent directory

Takes the name of a file and directory within the same directory, and moves the file entry into the specified directory.

Parameters

argc	Number of arguments.
argv	List of arguments argv[1] is the file to move, argv[2] is the destination directory.

Returns

int Success condition.

4.2.2.6 int pbsi_command (int *argc*, char ** *argv*)

print boot sector information

Parameters

argc	count of arguments passed in
argv	list of strings separated by space characters passed in

Returns

0

4.2.2.7 int prd_command (int *argc*, char ** *argv*)

print root directory

Parameters

argc	count of arguments passed in
argv	list of strings separated by space characters passed in

Returns

0

4.2.2.8 int rename_command (int *argc*, char ** *argv*)

renames a file or directory

Parameters

argc	count of arguments passed in
argv	list of strings separated by space characters passed in

Returns

0

4.2.2.9 int root_command (int *argc,* char ** *argv*)

change directory to root directory

Parameters

argc	count of arguments passed in
argv	list of strings separated by space characters passed in

Returns

0

change directory to root directory

Sends user's current working directory to the root.

Parameters

argc	Number of arguments
argv	List of string arguments.

Returns

int Success condition.

4.2.2.10 command_t* search_commands (char * *command*)

Search commands for a given command.

Parameters

command	A command name to search for

Returns

Pointer to command if found, else NULL

4.2.2.11 int type_command (int *argc*, char ** *argv*)

prints out contents of file

Parameters

argc	count of arguments passed in
argv	list of strings separated by space characters passed in

Returns

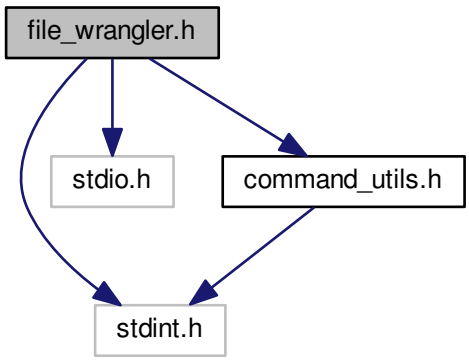
0

4.3 file_wrangler.h File Reference

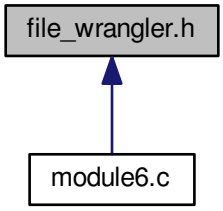
Holds all of the prototypes for file_wrangler.c and has BOOTSECTOR struct.

```
#include <stdint.h>
#include <stdio.h>
#include "command_utils.h"
```

Include dependency graph for file_wrangler.h:



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



Data Structures

- struct BOOTSECTORSTRUCT

Typedefs

- typedef struct BOOTSECTORSTRUCT **BOOTSECTORSTRUCT**

Functions

- void **loadBootSector** (FILE *fpIn)
- uint16_t * loadFAT (BYTE *fpIn, uint32_t startingSector)

  loads the FAT
- void loadDir (BYTE **dirp, BYTE *sys, uint32_t startingSec)

  loads a directory
- void loadCWD (BYTE *whole, uint32_t startingSec)

  loads the current working directory into global BYTE cwd*
- void loadROOT (BYTE *sys)

  loads the root into global BYTE root*
- void loadEntireSystem (char *filename)

  loads the entire system
- BOOTSECTORSTRUCT * getBootSectorIn ()

  gets boot sector in
- BYTE * getSystem ()

  gets entire system pointer
- uint32_t getSystemSize ()

  gets entire system size
- uint16_t * getDiabetes1 ()

  gets fat table 1
- uint16_t * getDiabetes2 ()

  gets fat table 2
- BYTE * getCWD ()

  *gets current working directory BYTE**
- BYTE * getRoot ()

gets the root directory

- char * getCwdPath ()

  gets current working directory path

- char * getImagePath ()

  gets root directory

- void setCwdPath (const char *jerry)

  sets cwd path

4.3.1 Detailed Description

Holds all of the prototypes for file_wrangler.c and has BOOTSECTOR struct.

4.3.2 Function Documentation

4.3.2.1 BOOTSECTORSTRUCT* getBootSectorIn ( )

gets boot sector in

Returns

pointer to boot sector in

4.3.2.2 BYTE* getCWD ( )

gets current working directory BYTE*

Returns

pointer to current working directory

4.3.2.3 char* getCwdPath ( )

gets current working directory path

Returns

char pointer that contains current working directory path

4.3.2.4 uint16_t* getDiabetes1 ( )

gets fat table 1

Returns

pointer to fat table 1

4.3.2.5 uint16_t* getDiabetes2 ( )

gets fat table 2

Returns

pointer to fat table 2

4.3.2.6 char* getImagePath ( )

gets root directory

Returns

pointer to root directory

4.3.2.7 BYTE* getRoot ( )

gets the root directory

Returns

pointer to the root directory

4.3.2.8 BYTE* getSystem ( )

gets entire system pointer

Returns

pointer to entire system

4.3.2.9 uint32_t getSystemSize ( )

gets entire system size

Returns

u32 int that is system size

4.3.2.10 void loadCWD (BYTE * *whole*, uint32_t *startingSec*)

loads the current working directory into global BYTE* cwd

Parameters

sys	raw byte array of entire file system
startingSector	number of the sector to start loading the cwd from

4.3.2.11 void loadDir (BYTE ∗∗ *dirp,* BYTE ∗ *sys,* uint32_t *startingSec*)

loads a directory

Parameters

sys	raw byte array of entire file system
startingSector	number of the sector to start loading the fat from

Returns

pointer to FAT

4.3.2.12 void loadEntireSystem (char ∗ *filename*)

loads the entire system

Parameters

filename	filename of the correct FAT12 Image

4.3.2.13 uint16_t∗ loadFAT (BYTE ∗ *fpln,* uint32_t *startingSector*)

loads the FAT

Parameters

fpln	raw byte array of entire file system
startingSector	number of the sector to start loading the fat from

Returns

pointer to FAT

4.3.2.14 void loadROOT (BYTE ∗ *sys*)

loads the root into global BYTE∗ root

Parameters

sys	raw byte array of entire file system

4.3.2.15 void setCwdPath (const char * *jerry*)

sets cwd path

Parameters

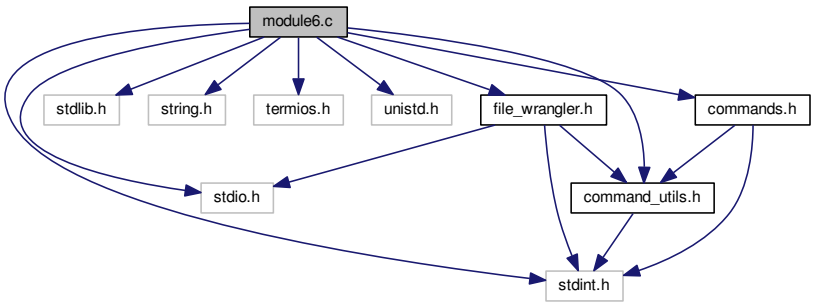
jerry	string to make cwdPath

4.4 module6.c File Reference

Entry point for file system.

```
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <termios.h>
#include <unistd.h>
#include "command_utils.h"
#include "commands.h"
#include "file_wrangler.h"
```

Include dependency graph for module6.c:



Functions

- ENTRY * file_recurse (BYTE *dir, BYTE *whole, const char *key)

  Searches through an entire file system for a file.
- int **main** (int argc, char **argv)

4.4.1 Detailed Description

Entry point for file system.

4.4.2 Function Documentation

4.4.2.1 ENTRY * file_recurse (BYTE * *dir,* BYTE * *whole,* const char * *key*)

Searches through an entire file system for a file.

Using a recursive search, this function will search through an entire image to find a specified file. This then returns an entry for this file.

Parameters

dir	Current searching directory
whole	Pointer to the system.
key	Name of the file to search for, including extension.

Returns

ENTRY* Entry of the file that was being searched for. NULL if the file is not found.

Index