myfind

1.0.0

Generated by Doxygen 1.6.1

Sat Mar 31 11:20:10 2018

Contents

1 Class Index **1**

1.1 Class List 1

2 File Index **3**

2.1 File List 3

3 Class Documentation **5**

3.1 Args Struct Reference 5

3.1.1 Detailed Description 6

3.1.2 Member Data Documentation 6

3.1.2.1 fileTypes 6

3.1.2.2 filterByFileType 6

3.1.2.3 filterByUserID 6

3.1.2.4 filterForNamePattern 6

3.1.2.5 filterForNoUser 6

3.1.2.6 filterForPathPattern 7

3.1.2.7 namePattern 7

3.1.2.8 pathPattern 7

3.1.2.9 printInExtendedFormat 7

3.1.2.10 searchPath 7

3.1.2.11 userID 7

3.2 FileNode Struct Reference 8

3.2.1 Detailed Description 8

3.2.2 Member Data Documentation 8

3.2.2.1 fileName 8

3.2.2.2 next 8

4 File Documentation **9**

4.1 myfind.c File Reference 9

4.1.1 Detailed Description 10
4.1.2 Enumeration Type Documentation 10
4.1.2.1 FileTypes 10
4.1.3 Function Documentation 11
4.1.3.1 AddListNode 11
4.1.3.2 CombinePath 11
4.1.3.3 ConvertToInteger 11
4.1.3.4 FreeList 12
4.1.3.5 main 12
4.1.3.6 ParseCommandLineArgs 12
4.1.3.7 ParseFileTypes 13
4.1.3.8 PrintFileInformation 13
4.1.3.9 PrintUsage 13
4.1.3.10 QueryUserID 14
4.1.3.11 SearchDirectory 14
4.1.3.12 SearchFile 14
4.1.3.13 ShouldPrintFileInformation 15

Chapter 1

Class Index

1.1 Class List

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

Args (The command line arguments provided to the application at startup) 5
FileNode (A single node in the linked list of file names) 8

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

myfind.c 9

Chapter 3

Class Documentation

3.1 Args Struct Reference

The command line arguments provided to the application at startup.

Public Attributes

- char $*$ searchPath

  The path of the file or directory to search in. NULL if no search path was provided.

- bool printInExtendedFormat

  Indicates whether the output should be printed in extended list format.

- bool filterByFileType

  Indicates whether only files of the types specified in `fileTypes` should be printed.

- enum FileTypes fileTypes

  Only files with the types specified in this set of flags will be printed. This member is only valid if `filterByFileType` is true.

- bool filterByUserID

  Indicates whether only files belonging to a user with the ID specified in `userID` should be printed. This member has precedence over `filterUserName` and `filterForNoUser`.

- int userID

  Only files belonging to a user with this ID will be printed. This member is only valid if `filterByUserID` is true.

- bool filterForNoUser

  Indicates whether only files not belonging to any user should be printed.

- bool filterForNamePattern

  Indicates whether only files with names that match the pattern specified in `namePattern` should be printed.

- char ∗ namePattern

  Only files whose name matches this pattern will be printed. This member is only valid if `filterForNamePattern` *is true.*

- bool filterForPathPattern

  Indicates whether only files where the whole path matches the pattern specified in `pathPattern` *should be printed.*

- char ∗ pathPattern

  Only files where the whole path matches this pattern will be printed. This member is only valid if `filterForPathPattern` *is true.*

3.1.1 Detailed Description

Definition at line 51 of file myfind.c.

3.1.2 Member Data Documentation

3.1.2.1 enum FileTypes Args::fileTypes

Definition at line 62 of file myfind.c.

Referenced by ParseCommandLineArgs(), and ShouldPrintFileInformation().

3.1.2.2 bool Args::filterByFileType

Definition at line 60 of file myfind.c.

Referenced by ParseCommandLineArgs(), and ShouldPrintFileInformation().

3.1.2.3 bool Args::filterByUserID

Definition at line 65 of file myfind.c.

Referenced by ParseCommandLineArgs(), and ShouldPrintFileInformation().

3.1.2.4 bool Args::filterForNamePattern

Definition at line 73 of file myfind.c.

Referenced by ParseCommandLineArgs(), and ShouldPrintFileInformation().

3.1.2.5 bool Args::filterForNoUser

Definition at line 70 of file myfind.c.

Referenced by ParseCommandLineArgs(), and ShouldPrintFileInformation().

3.1.2.6 bool Args::filterForPathPattern

Definition at line 78 of file myfind.c.

Referenced by ParseCommandLineArgs(), and ShouldPrintFileInformation().

3.1.2.7 char∗ Args::namePattern

Definition at line 75 of file myfind.c.

Referenced by ParseCommandLineArgs().

3.1.2.8 char∗ Args::pathPattern

Definition at line 80 of file myfind.c.

Referenced by ParseCommandLineArgs().

3.1.2.9 bool Args::printInExtendedFormat

Definition at line 57 of file myfind.c.

Referenced by ParseCommandLineArgs(), and PrintFileInformation().

3.1.2.10 char∗ Args::searchPath

Definition at line 54 of file myfind.c.

Referenced by main(), and ParseCommandLineArgs().

3.1.2.11 int Args::userID

Definition at line 67 of file myfind.c.

Referenced by ParseCommandLineArgs().

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

- myfind.c

3.2 FileNode Struct Reference

A single node in the linked list of file names. Collaboration diagram for FileNode:

Public Attributes

- char ∗ fileName

  The name of the file (or directory). This member must not be NULL.

- struct FileNode ∗ next

  A pointer to the next node in the list, or NULL if this is the last node.

3.2.1 Detailed Description

Definition at line 84 of file myfind.c.

3.2.2 Member Data Documentation

3.2.2.1 char∗ FileNode::fileName

Definition at line 87 of file myfind.c.

Referenced by AddListNode(), FreeList(), and SearchDirectory().

3.2.2.2 struct FileNode∗ FileNode::next `[read]`

Definition at line 90 of file myfind.c.

Referenced by AddListNode(), FreeList(), and SearchDirectory().

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

- myfind.c

Chapter 4

File Documentation

4.1 myfind.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <errno.h>
#include <assert.h>
#include <dirent.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <sys/queue.h>
```

Include dependency graph for myfind.c:

Classes

- struct Args

  The command line arguments provided to the application at startup.

- struct FileNode

  A single node in the linked list of file names.

Enumerations

- enum FileTypes {
  None = 0, BlockSpecialFile = 1 << 0, CharacterSpecialFile = 1 << 1, Directory = 1 << 2,
  NamedPipe = 1 << 3, RegularFile = 1 << 4, SymbolicLink = 1 << 5, Socket = 1 << 6 }

Contains flags indicating the file types to be printed in the application's output.

Functions

- void PrintUsage ()

  Prints an explanation of the application's command line arguments.

- bool ParseCommandLineArgs (char ∗argv[ ], struct Args ∗args)
- bool ConvertToInteger (char ∗s, int ∗i)
- bool QueryUserID (char ∗userName, int ∗userID)
- bool ParseFileTypes (char ∗fileTypeChars, enum FileTypes ∗fileTypes)
- void SearchFile (char ∗file_name, struct Args ∗args)
- void SearchDirectory (char ∗dir_name, struct Args ∗args)
- char ∗ CombinePath (char ∗path1, char ∗path2)
- struct FileNode ∗ AddListNode (struct FileNode ∗∗head, char ∗fileName)
- void FreeList (struct FileNode ∗∗head)
- bool ShouldPrintFileInformation (char ∗filePath, struct stat ∗fileInformation, struct Args ∗args)
- void PrintFileInformation (char ∗filePath, struct stat ∗fileInformation, struct Args ∗args)
- int main (int argc, char ∗argv[ ])

4.1.1 Detailed Description

myfind - A simplified version of the "find" utility provided by the Linux shell.

Author:

Alexander Feldinger <ic17b055@technikum-wien.at>
Thomas Haberl <ic17b021@technikum-wien.at>
Michael Zajac <ic17b088@technikum-wien.at>

Date:

2018-03-31

Definition in file myfind.c.

4.1.2 Enumeration Type Documentation

4.1.2.1 enum FileTypes

Enumerator:

None No filtering by file type.

BlockSpecialFile Block special files should be printed.

CharacterSpecialFile Character special files should be printed.

Directory Directories should be printed.

NamedPipe Named pipes should be printed.

RegularFile Regular files should be printed.

SymbolicLink Symbolic links should be printed.

Socket Sockets should be printed.

Definition at line 29 of file myfind.c.

4.1.3 Function Documentation

4.1.3.1 struct FileNode ∗ AddListNode (struct FileNode ∗∗ *head*, char ∗ *fileName*) `[read]`

Creates a new file node and adds it to the linked list.

Parameters:

head A pointer to the head of the linked list into which the new node should be inserted.

fileName The file name to store in the created node.

Returns:

The created file node.

Definition at line 602 of file myfind.c.

References FileNode::fileName, and FileNode::next.

Referenced by SearchDirectory().

4.1.3.2 char ∗ CombinePath (char ∗ *path1*, char ∗ *path2*)

Concatenates the provided path strings into a single path, adding or removing the intermediate directory separator as necessary.

Parameters:

path1 The first path to combine.

path2 The second path to combine.

Returns:

The combined path as a newly allocated string, which needs to be released with free().

Definition at line 535 of file myfind.c.

Referenced by SearchDirectory().

4.1.3.3 bool ConvertToInteger (char ∗ *s*, int ∗ *i*)

Converts the provided string to an integer.

Parameters:

s The string to convert to an integer.

i A pointer to the integer value in which to store the converted string.

Returns:

true if the string could successfully be converted to an integer value. Otherwise, false.

Definition at line 383 of file myfind.c.

Referenced by ParseCommandLineArgs().

4.1.3.4 void FreeList (struct FileNode ** *head*)

Frees all nodes in the provided linked list.

Parameters:

head A pointer to the head of the linked list to be freed.

Definition at line 644 of file myfind.c.

References FileNode::fileName, and FileNode::next.

Referenced by SearchDirectory().

4.1.3.5 int main (int *argc*, char * *argv*[ ])

The entry point of the application.

Parameters:

argc The number of command line arguments in `argv`.

argv The array of command line arguments. The last element of the array is NULL.

Returns:

Zero if execution was successful. -1 if an unrecoverable error occurred during execution.

Definition at line 119 of file myfind.c.

References ParseCommandLineArgs(), PrintUsage(), SearchFile(), and Args::searchPath.

Here is the call graph for this function:

4.1.3.6 bool ParseCommandLineArgs (char * *argv*[ ], struct Args * *args*)

Parses and validates the application's command line arguments.

Parameters:

argv The array of command line arguments. The last element in the array must be NULL.

args A pointer to the struct of processed command line arguments within which to store the parsed information.

Returns:

true if all command line arguments could be parsed successfully. Otherwise, false.

Definition at line 182 of file myfind.c.

References ConvertToInteger(), Args::fileTypes, Args::filterByFileType, Args::filterByUserID, Args::filterForNamePattern, Args::filterForNoUser, Args::filterForPathPattern, Args::namePattern, ParseFileTypes(), Args::pathPattern, Args::printInExtendedFormat, QueryUserID(), Args::searchPath, and Args::userID.

Referenced by main().

Here is the call graph for this function:

4.1.3.7 bool ParseFileTypes (char $*$*fileTypeChars*, enum FileTypes $*$*fileTypes*)

Parses the string that specifies the file types to be printed.

Parameters:

fileTypeChars The array of characters representing the file types to be printed.

fileTypes A pointer to the enumeration value in which to store the parsed information.

Returns:

true if all file type characters could be parsed successfully. Otherwise, false.

Definition at line 326 of file myfind.c.

References BlockSpecialFile, CharacterSpecialFile, Directory, NamedPipe, None, RegularFile, Socket, and SymbolicLink.

Referenced by ParseCommandLineArgs().

4.1.3.8 void PrintFileInformation (char $*$*filePath*, struct stat $*$*fileInformation*, struct Args $*$ *args*)

Prints the information of a single file or directory.

Parameters:

filePath The path of the file to be printed.

fileInformation The information of the file as returned by stat().

args The command line options that specify the format in which to print the file's information.

Returns:

The created file node.

Definition at line 725 of file myfind.c.

References Args::printInExtendedFormat.

Referenced by SearchFile().

4.1.3.9 void PrintUsage ()

Definition at line 154 of file myfind.c.

Referenced by main().

4.1.3.10 bool QueryUserID (char ∗ *userName*, int ∗ *userID*)

Queries the user ID of the user with the specified name.

Parameters:

userName The name of the user for which to get the ID.

userID A pointer to the integer value in which to store the queries user ID.

Returns:

true if the user ID could be queried successfully. Otherwise, false.

Definition at line 392 of file myfind.c.

Referenced by ParseCommandLineArgs().

4.1.3.11 void SearchDirectory (char ∗ *directoryPath*, struct Args ∗ *args*)

Enumerates the files and directories below the specified directory path and prints the information of each entry according to the actions specified in `args`.

Parameters:

directoryPath The path of the directory to process.

args The command line options representing the actions to use for printing the information of each file or directory entry.

Definition at line 437 of file myfind.c.

References AddListNode(), CombinePath(), FileNode::fileName, FreeList(), FileNode::next, and Search-File().

Referenced by SearchFile().

Here is the call graph for this function:

4.1.3.12 void SearchFile (char ∗ *filePath*, struct Args ∗ *args*)

Recursively walks through all the files and directories below the specified path and prints the information of each entry according to the actions specified in `args`.

Parameters:

filePath The path of the file or directory to process.

args The command line options representing the actions to use for printing the information of each file or directory entry.

Definition at line 401 of file myfind.c.

References PrintFileInformation(), SearchDirectory(), and ShouldPrintFileInformation().

Referenced by main(), and SearchDirectory().

Here is the call graph for this function:

4.1.3.13 bool ShouldPrintFileInformation (char ∗ *filePath*, struct stat ∗ *fileInformation*, struct Args ∗ *args*)

Determines whether the file with the provided path and information should be printed based on the application's command line arguments.

Parameters:

filePath The path of the file to be printed.

fileInformation The information of the file as returned by stat().

args The command line options that specify the criteria by which to select the files to be printed.

Definition at line 669 of file myfind.c.

References BlockSpecialFile, CharacterSpecialFile, Directory, Args::fileTypes, Args::filterByFileType, Args::filterByUserID, Args::filterForNamePattern, Args::filterForNoUser, Args::filterForPathPattern, NamedPipe, RegularFile, Socket, and SymbolicLink.

Referenced by SearchFile().

Index

AddListNode
    myfind.c, 11
Args, 5
    fileTypes, 6
    filterByFileType, 6
    filterByUserID, 6
    filterForNamePattern, 6
    filterForNoUser, 6
    filterForPathPattern, 6
    namePattern, 7
    pathPattern, 7
    printInExtendedFormat, 7
    searchPath, 7
    userID, 7

BlockSpecialFile
    myfind.c, 10

CharacterSpecialFile
    myfind.c, 10
CombinePath
    myfind.c, 11
ConvertToInteger
    myfind.c, 11

Directory
    myfind.c, 10

fileName
    FileNode, 8
FileNode, 8
    fileName, 8
    next, 8
FileTypes
    myfind.c, 10
fileTypes
    Args, 6
filterByFileType
    Args, 6
filterByUserID
    Args, 6
filterForNamePattern
    Args, 6
filterForNoUser
    Args, 6
filterForPathPattern
    Args, 6
FreeList
    myfind.c, 12

main
    myfind.c, 12
myfind.c, 9
    AddListNode, 11
    BlockSpecialFile, 10
    CharacterSpecialFile, 10
    CombinePath, 11
    ConvertToInteger, 11
    Directory, 10
    FileTypes, 10
    FreeList, 12
    main, 12
    NamedPipe, 10
    None, 10
    ParseCommandLineArgs, 12
    ParseFileTypes, 13
    PrintFileInformation, 13
    PrintUsage, 13
    QueryUserID, 13
    RegularFile, 10
    SearchDirectory, 14
    SearchFile, 14
    ShouldPrintFileInformation, 14
    Socket, 10
    SymbolicLink, 10

NamedPipe
    myfind.c, 10
namePattern
    Args, 7
next
    FileNode, 8
None
    myfind.c, 10

ParseCommandLineArgs
    myfind.c, 12
ParseFileTypes
    myfind.c, 13
pathPattern
    Args, 7

PrintFileInformation
myfind.c, 13
printInExtendedFormat
Args, 7
PrintUsage
myfind.c, 13

QueryUserID
myfind.c, 13

RegularFile
myfind.c, 10

SearchDirectory
myfind.c, 14
SearchFile
myfind.c, 14
searchPath
Args, 7
ShouldPrintFileInformation
myfind.c, 14
Socket
myfind.c, 10
SymbolicLink
myfind.c, 10

userID
Args, 7