Betriebssysteme Find

1.0.0

Generated by Doxygen 1.6.1

Thu Apr 12 09:21:37 2018

Contents

Chapter 1

Todo List

File myfind.c Review it for missing error checks.

link appent to path

fix formatting of ls

test more completely

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 opt_struct Struct Reference

opt_struct is used for storing the option information: the option itself, the number of expected arguments for verification and a mask value

Public Attributes

- char option [6+1]
- int opt_args
- int mask_val

4.1.1 Detailed Description

opt_struct is used for storing the option information: the option itself, the number of expected arguments for verification and a mask value

Definition at line 84 of file myfind.c.

4.1.2 Member Data Documentation

4.1.2.1 int opt_struct::mask_val

Definition at line 87 of file myfind.c.

Referenced by ret_arg_exp().

4.1.2.2 int opt_struct::opt_args

Definition at line 86 of file myfind.c.

4.1.2.3 char opt_struct::option[6+1]

Definition at line 85 of file myfind.c.

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

- myfind.c

4.2 type_struct Struct Reference

type_struct is used for storing chars of valid types and a mask value. This struct is used for the "-type" function.

Public Attributes

- char type [2]
- int type_val

4.2.1 Detailed Description

type_struct is used for storing chars of valid types and a mask value. This struct is used for the "-type" function.

Definition at line 91 of file myfind.c.

4.2.2 Member Data Documentation

4.2.2.1 char type_struct::type[2]

Definition at line 92 of file myfind.c.

4.2.2.2 int type_struct::type_val

Definition at line 93 of file myfind.c.

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

- myfind.c

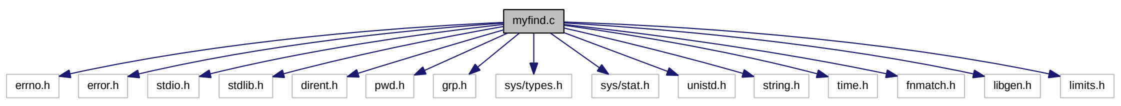
Chapter 5

File Documentation

5.1 myfind.c File Reference

```
#include <errno.h>
#include <error.h>
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <pwd.h>
#include <grp.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include <fnmatch.h>
#include <libgen.h>
#include <limits.h>
```

Include dependency graph for myfind.c:



Classes

- struct opt_struct

    opt_struct is used for storing the option information: the option itself, the number of expected arguments for verification and a mask value

- struct type_struct

  type_struct is used for storing chars of valid types and a mask value. This struct is used for the "-type" function.

Defines

- #define TRUE 1

  TRUE is a MAKRO used for boolean logic.

- #define FALSE 0

  FALSE is a MAKRO used for boolean logic.

- #define MISSING_ARG 2

  MISSING_ARG is a check value for validate_options() if an argument to an option is missing.

- #define FILE_INFO_STRING 10

  FILE_INFO_STRING is a MAKRO determining the string length for the ownership and file information of files or directories.

- #define MAX_OPTION_STRING_LENGTH 6

  MAX_OPTION_STRING_LENGTH is a MAKRO determining the maximum string length of an option string stored in struct opt_struct.

- #define TYPE_OPTIONS_CNT 7

  TYPE_OPTIONS_CNT is a MAKRO that represents the count of valid options.

- #define MAX_UID_GUID_STR_LENGTH 11

  MAX_UID_GUID_STR_LENGTH represents the maximum digit_length an integer 2^32-1 can have in base 10. ".

- #define BLOC_TYPE 100

  BLOC_TYPE is a MAKRO to be able to use switch case for types of files or directories.

- #define CHAR_TYPE 101

  CHAR_TYPE is a MAKRO to be able to use switch case for types of files or directories.

- #define DIR_TYPE 102

  DIR_TYPE is a MAKRO to be able to use switch case for types of files or directories.

- #define PIPE_TYPE 103

  PIPE_TYPE is a MAKRO to be able to use switch case for types of files or directories.

- #define FILE_TYPE 104

  FILE_TYPE is a MAKRO to be able to use switch case for types of files or directories.

- #define LINK_TYPE 105

  LINK_TYPE is a MAKRO to be able to use switch case for types of files or directories.

- #define SOCK_TYPE 106

  SOCK_TYPE is a MAKRO to be able to use switch case for types of files or directories.

- #define NAME_OPT 1001

  NAME_OPT is a MAKRO to be able to use switch case for options and to return the string of an option if needed of files or directories.

- #define TYPE_OPT 1002

  TYPE_OPT is a MAKRO to be able to use switch case for options and to return the string of an option if needed of files or directories.

- #define USER_OPT 1003

  USER_OPT is a MAKRO to be able to use switch case for options and to return the string of an option if needed of files or directories.

- #define LS_OPT 1004

  LS_OPT is a MAKRO to be able to use switch case for options and to return the string of an option if needed of files or directories.

- #define PRINT_OPT 1005

  PRINT_OPT is a MAKRO to be able to use switch case for options and to return the string of an option if needed of files or directories.

Functions

- static void do_file (const char ∗file_name, const char ∗const ∗parms)

  do_file() gathers information about a file and passes its name to action_print() if the file opened is a directory it calls do_dir() and passes the directories name

- static void do_dir (const char ∗dir_name, const char ∗const ∗parms)

  opens, reads from and closes directories

- static int action_print (const char ∗file_name)

  action_print() prints the passed string (a file_name or a directory_name) to stdout It is called either when "-print" is the only option passed to the programm or the action-input following the path name is valid and the output need to be printed if printing fails it calls the function print_error()

- static int action_ls (const char ∗file_name, const struct stat ∗file_info)

  print the file or directory name as well as the following details to stdout
  - *inode number*
  - *number of blocks*
  - *permissions*
  - *number of links*
  - *owner*
  - *group*
  - *last modification time*
  - *path*

- static int print_error (const char ∗message, const char ∗file_name, const int my_errno) *prints errors with a message*

- static void print_usage (const char ∗message)

  print_usage prints information on how to use the program to stdout if printing fails it calls the function print_error()

- static int validate_options (const char ∗const ∗parms)

  validate_options This function examines the given command line arguments for existing options. in case the pattern is missing or wrong it prints information about expected arguments and quits the program

- static int compare_name (const char ∗file_name, const char ∗compstring)

  compare_name compares string to pattern via basename and returns matches it uses fnmatch and in case of failure exits the program

- static int compare_user (const char ∗file_name, struct stat ∗file_info, const char ∗compstring)

  compare_user checks the compstring for validity and if proceeding compares the string to the file informations user. if there is no match in name compstring is taken as uid and it checks again. if the user asked for doesn't exist it exits the program after informing the user via stderr

- static int compare_type (const char ∗file_name, struct stat ∗file_info, const char ∗compstring)

  compare_type checks the given string in compstring for validity and if valid compares the file's information with the type represented by compstring

- static int ret_mask_to_opt_string (const char ∗arg)

  val_opt_string is called from the validate_options() function and checks a given string if it matches the defined options

- static int ret_arg_exp (const int mask)

  ret_arg_exp is called from validate_options() and checks if argument is expected by the option

- static char ∗ ret_opt_on_mask (const int mask)

  ret_opt_on_mask returns the optionstring to the given masked stored in opt_struct it is only called by main()

- static long long convert_str_to_uid (const char ∗string)

  convert_str_to_num converts a given string to a numerical value. This function is a helper function for compare_user() and it uses strtoll

- static char ∗ combine_paths (const char ∗src_1, const char ∗src_2)

  combine_paths combines two pathstrings to one pathstring and returns the newly created pathstring or NULL if an Error occured

- int main (int argc, const char ∗argv[ ])

  A smaller find program.

Variables

- static const char ∗ program = "<not yet set>"

  The program is set to argv[0] in order for print_error() to access it without getting it as a parameter.

- static int param_cnt = 0

param_cnt saves the argc and is used by practically all functions

- static struct opt_struct valid_options [ ]

  valid_options contains all implemented action-strings, the number of expected arguments after each string, and a mask the option can be identified by

- static struct type_struct type_chars [ ]

  type_chars contains all valid types [bcdpfls] of a file and it is used in the function compare_type()

5.1.1 Detailed Description

Betriebssysteme myfind-program File. Beispiel 1

Author:

- Magdalena Andrae <`ic17b079@technikum-wien.at`>
- Rainhardt Gabriel <`ic17b078@technikum-wien.at`>

Date:

2018/03/01

Version:

001

Todo

Review it for missing error checks.
link appent to path
fix formatting of ls
test more completely

Last modified 2018/04/07. Last modified by Rainhardt

Definition in file myfind.c.

5.1.2 Define Documentation

5.1.2.1 #define BLOC_TYPE 100

BLOC_TYPE is a MAKRO to be able to use switch case for types of files or directories.

Definition at line 56 of file myfind.c.

Referenced by compare_type().

5.1.2.2 #define CHAR_TYPE 101

CHAR_TYPE is a MAKRO to be able to use switch case for types of files or directories.

Definition at line 58 of file myfind.c.

Referenced by compare_type().

5.1.2.3 #define DIR_TYPE 102

DIR_TYPE is a MAKRO to be able to use switch case for types of files or directories.

Definition at line 60 of file myfind.c.

Referenced by compare_type().

5.1.2.4 #define FALSE 0

FALSE is a MAKRO used for boolean logic.

Definition at line 44 of file myfind.c.

Referenced by compare_name(), compare_type(), compare_user(), do_dir(), do_file(), main(), ret_arg_-exp(), ret_mask_to_opt_string(), and validate_options().

5.1.2.5 #define FILE_INFO_STRING 10

FILE_INFO_STRING is a MAKRO determining the string length for the ownership and file information of files or directories.

Definition at line 48 of file myfind.c.

5.1.2.6 #define FILE_TYPE 104

FILE_TYPE is a MAKRO to be able to use switch case for types of files or directories.

Definition at line 64 of file myfind.c.

Referenced by compare_type().

5.1.2.7 #define LINK_TYPE 105

LINK_TYPE is a MAKRO to be able to use switch case for types of files or directories.

Definition at line 66 of file myfind.c.

Referenced by compare_type().

5.1.2.8 #define LS_OPT 1004

LS_OPT is a MAKRO to be able to use switch case for options and to return the string of an option if needed of files or directories.

Definition at line 76 of file myfind.c.

Referenced by do_file(), and validate_options().

5.1.2.9 #define MAX_OPTION_STRING_LENGTH 6

MAX_OPTION_STRING_LENGTH is a MAKRO determining the maximum string length of an option string stored in struct opt_struct.

Definition at line 50 of file myfind.c.

Referenced by ret_opt_on_mask().

5.1.2.10 #define MAX_UID_GUID_STR_LENGTH 11

MAX_UID_GUID_STR_LENGTH represents the maximum digit_length an integer 2^32-1 can have in base 10. ".

Definition at line 54 of file myfind.c.

Referenced by action_ls().

5.1.2.11 #define MISSING_ARG 2

MISSING_ARG is a check value for validate_options() if an argument to an option is missing.

Definition at line 46 of file myfind.c.

Referenced by validate_options().

5.1.2.12 #define NAME_OPT 1001

NAME_OPT is a MAKRO to be able to use switch case for options and to return the string of an option if needed of files or directories.

Definition at line 70 of file myfind.c.

Referenced by validate_options().

5.1.2.13 #define PIPE_TYPE 103

PIPE_TYPE is a MAKRO to be able to use switch case for types of files or directories.

Definition at line 62 of file myfind.c.

Referenced by compare_type().

5.1.2.14 #define PRINT_OPT 1005

PRINT_OPT is a MAKRO to be able to use switch case for options and to return the string of an option if needed of files or directories.

Definition at line 78 of file myfind.c.

Referenced by do_file(), and validate_options().

5.1.2.15 #define SOCK_TYPE 106

SOCK_TYPE is a MAKRO to be able to use switch case for types of files or directories.

Definition at line 68 of file myfind.c.

Referenced by compare_type().

5.1.2.16 #define TRUE 1

TRUE is a MAKRO used for boolean logic.

Definition at line 42 of file myfind.c.

Referenced by compare_name(), compare_type(), compare_user(), do_file(), main(), ret_arg_exp(), and validate_options().

5.1.2.17 #define TYPE_OPT 1002

TYPE_OPT is a MAKRO to be able to use switch case for options and to return the string of an option if needed of files or directories.

Definition at line 72 of file myfind.c.

Referenced by validate_options().

5.1.2.18 #define TYPE_OPTIONS_CNT 7

TYPE_OPTIONS_CNT is a MAKRO that represents the count of valid options.

Definition at line 52 of file myfind.c.

Referenced by compare_type().

5.1.2.19 #define USER_OPT 1003

USER_OPT is a MAKRO to be able to use switch case for options and to return the string of an option if needed of files or directories.

Definition at line 74 of file myfind.c.

Referenced by validate_options().

5.1.3 Function Documentation

5.1.3.1 static int action_ls (const char ∗ *file_name*, const struct stat ∗ *file_info*) `[static]`

print the file or directory name as well as the following details to stdout

- inode number
- number of blocks
- permissions
- number of links
- owner
- group
- last modification time
- path

Parameters:

file_name

file_info

Return values:

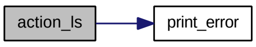
none

Definition at line 414 of file myfind.c.

References MAX_UID_GUID_STR_LENGTH, and print_error().

Referenced by do_file().

Here is the call graph for this function:



5.1.3.2 static int action_print (const char $*$ *file_name*) `[static]`

action_print() prints the passed string (a file_name or a directory_name) to stdout It is called either when "-print" is the only option passed to the programm or the action-input following the path name is valid and the output need to be printed if printing fails it calls the function print_error()

Parameters:

file_name is the filename to be printed

Return values:

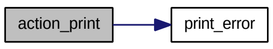
none

Definition at line 389 of file myfind.c.

References print_error().

Referenced by do_file().

Here is the call graph for this function:



5.1.3.3 static char $*$ combine_paths (const char $*$ *src_1*, const char $*$ *src_2*) `[static]`

combine_paths combines two pathstrings to one pathstring and returns the newly created pathstring or NULL if an Error occured

Parameters:

src_1 string of first path

src_2 string of to ammend path

Return values:

combined_path if everything was successful

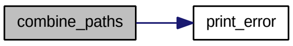
NULL if failure occured

Definition at line 367 of file myfind.c.

References print_error().

Referenced by do_dir().

Here is the call graph for this function:



5.1.3.4 static int compare_name (const char ∗ *file_name*, const char ∗ *compstring*) `[static]`

compare_name compares string to pattern via basename and returns matches it uses fnmatch and in case of failure exits the program

Parameters:

file_name String that contains the filename to match the comparestrings criteria

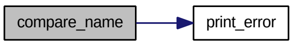
compstring String that contains to information that the file has to

Definition at line 736 of file myfind.c.

References FALSE, print_error(), and TRUE.

Referenced by do_file().

Here is the call graph for this function:



5.1.3.5 static int compare_type (const char ∗ *file_name*, struct stat ∗ *file_info*, const char ∗ *compstring*) `[static]`

compare_type checks the given string in compstring for validity and if valid compares the file's information with the type represented by compstring

Parameters:

file_name String that contains the filename to match the comparestrings criteria

file_info contains file_info

compstring String that contains to information that the file has to

Return values:

TRUE file matches criteria

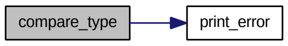
FALSE file doesnt macht criteria

Definition at line 769 of file myfind.c.

References BLOC_TYPE, CHAR_TYPE, DIR_TYPE, FALSE, FILE_TYPE, LINK_TYPE, PIPE_TYPE, print_error(), program, SOCK_TYPE, TRUE, and TYPE_OPTIONS_CNT.

Referenced by do_file().

Here is the call graph for this function:



5.1.3.6 static int compare_user (const char $*$ *file_name*, struct stat $*$ *file_info*, const char $*$ *compstring*) `[static]`

compare_user checks the compstring for validity and if proceeding compares the string to the file informations user. if there is no match in name compstring is taken as uid and it checks again. if the user asked for doesn't exist it exits the program after informing the user via stderr

Parameters:

file_name	String that contains the filename. This is needed for print_error()
file_info	contains stat info of file
compstring	String that contains username or uid that the file has to match

Return values:

TRUE	if the file belongs to the user or uid
FALSE	if the file belongs to someone else

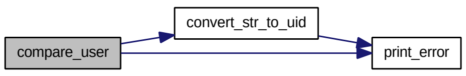
$***$init vars

Definition at line 831 of file myfind.c.

References convert_str_to_uid(), FALSE, print_error(), program, and TRUE.

Referenced by do_file().

Here is the call graph for this function:



5.1.3.7 static long long convert_str_to_uid (const char $*$ *string*) `[static]`

convert_str_to_num converts a given string to a numerical value. This function is a helper function for compare_user() and it uses strtoll

Parameters:

string	comparestring

Return values:

uid if string was a number

-1 if string was number but too long for a uid

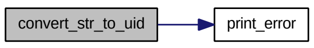
-2 if string contained chars

Definition at line 912 of file myfind.c.

References print_error().

Referenced by compare_user().

Here is the call graph for this function:



5.1.3.8 static void do_dir (const char ∗ *dir_name*, const char ∗const ∗ *parms*) `[static]`

opens, reads from and closes directories This function is called by the function do_file() and opens, reads and closes the directories. It parses the directory's entries to a string and hands it back to do_file()

Parameters:

dir_name is the directory string

parms are the program's parameters

Return values:

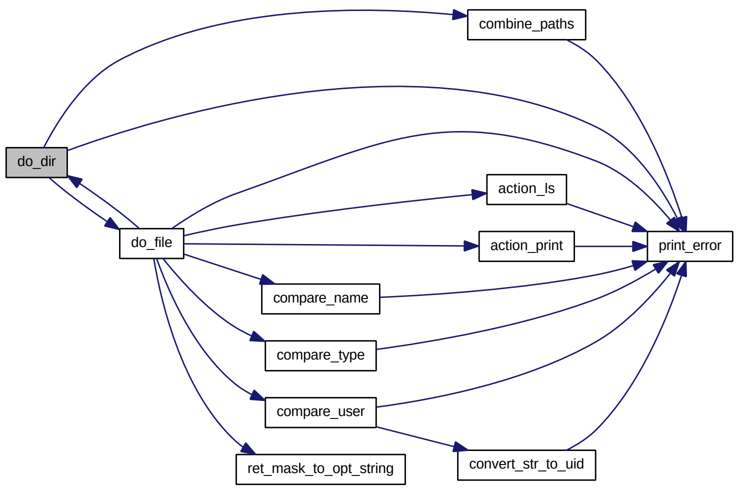
none

Definition at line 299 of file myfind.c.

References combine_paths(), do_file(), FALSE, and print_error().

Referenced by do_file().

Here is the call graph for this function:



5.1.3.9 static void do_file (const char * *file_name*, const char *const * *parms*) `[static]`

do_file() gathers information about a file and passes its name to action_print() if the file opened is a directory it calls do_dir() and passes the directories name

Parameters:

file_name is a string containing a directoryname or a filename

parms are all the arguments given to the command line including the program name itself at index 0

Return values:

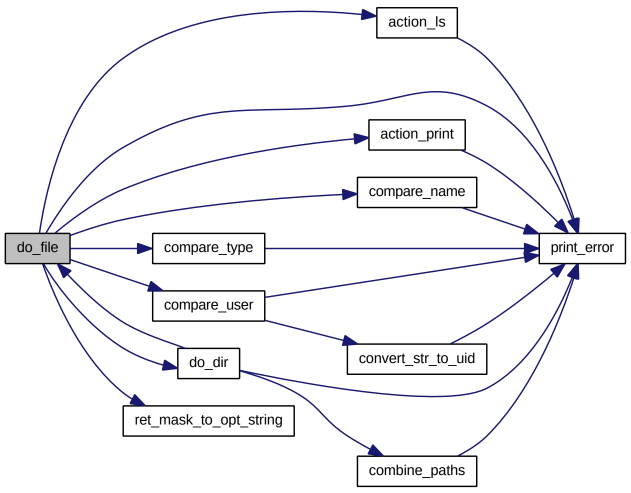
none

Definition at line 210 of file myfind.c.

References action_ls(), action_print(), compare_name(), compare_type(), compare_user(), do_dir(), FALSE, LS_OPT, param_cnt, print_error(), PRINT_OPT, ret_mask_to_opt_string(), and TRUE.

Referenced by do_dir(), and main().

Here is the call graph for this function:



5.1.3.10 int main (int *argc*, const char * *argv*[ ])

A smaller find program. This is the main entry point for any C program.

Parameters:

argc the number of arguments typed into the command line.

argv the command line input/the arguments themselves. The program name is held in argv[0].

Return values:

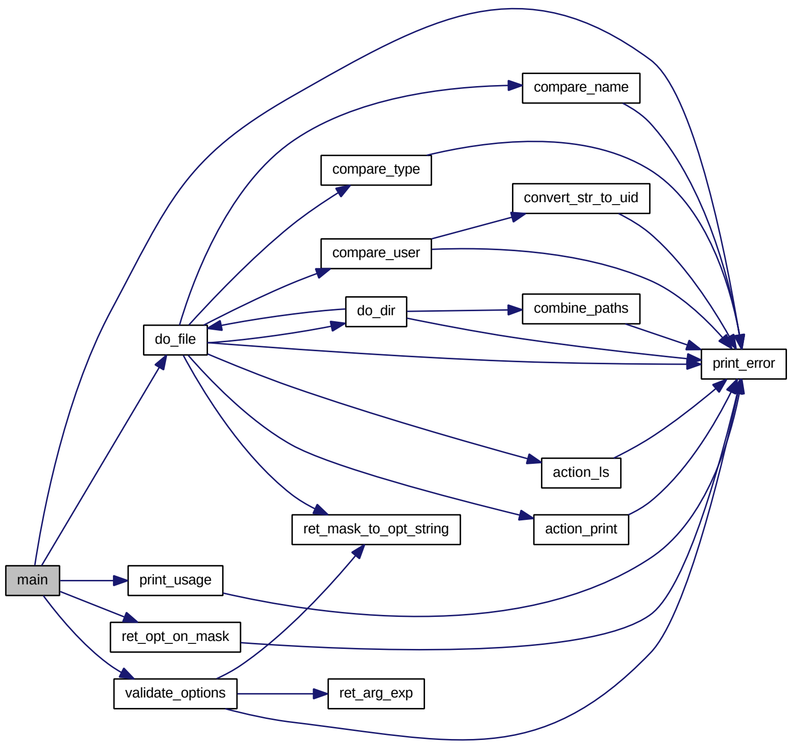
EXIT_SUCCESS

EXIT_FAILURE

Definition at line 149 of file myfind.c.

References do_file(), FALSE, param_cnt, print_error(), print_usage(), program, ret_opt_on_mask(), TRUE, and validate_options().

Here is the call graph for this function:



5.1.3.11 static int print_error (const char ∗ *message*, const char ∗ *file_name*, const int *my_errno*) `[static]`

prints errors with a message This function is called if an error occured. If the error number (errno != 0) is activated an additional error message is provided by strerror(errno) The output is "stderr" and contains the name of the program, the message and the file.

Parameters:

message	string that contains the message in which function something went wrong
file_name	string that contains the file or directory entry where myfind produced an error
my_errno	saved errno value or 0

Return values:

EXIT_SUCCESS

EXIT_FAILURE

Definition at line 194 of file myfind.c.

References program.

Referenced by action_ls(), action_print(), combine_paths(), compare_name(), compare_type(), compare_-user(), convert_str_to_uid(), do_dir(), do_file(), main(), print_usage(), ret_opt_on_mask(), and validate_-options().

5.1.3.12 static void print_usage (const char ∗ *message*) `[static]`

print_usage prints information on how to use the program to stdout if printing fails it calls the function print_error()

Parameters:

message

Return values:

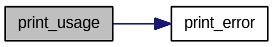
none

Definition at line 591 of file myfind.c.

References print_error(), and program.

Referenced by main().

Here is the call graph for this function:



5.1.3.13 static int ret_arg_exp (const int *mask*) `[static]`

ret_arg_exp is called from validate_options() and checks if argument is expected by the option

Return values:

TRUE	if an argument is expected by an option
FALSE	if no argument is expected by an option

Definition at line 682 of file myfind.c.

References FALSE, opt_struct::mask_val, and TRUE.

Referenced by validate_options().

5.1.3.14 static int ret_mask_to_opt_string (const char ∗ *arg*) `[static]`

val_opt_string is called from the validate_options() function and checks a given string if it matches the defined options

Return values:

FALSE	if no match
_arg_expected	MASK of matched option

Definition at line 694 of file myfind.c.

References FALSE.

Referenced by do_file(), and validate_options().

5.1.3.15 static char ∗ ret_opt_on_mask (const int *mask*) `[static]`

ret_opt_on_mask returns the optionstring to the given masked stored in opt_struct it is only called by main()

Parameters:

mask integer that has to be checked if it matches the mask of option

Return values:

nvalid in case the mask has no option stored

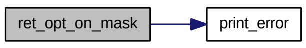
option in case the mask has an option

Definition at line 710 of file myfind.c.

References MAX_OPTION_STRING_LENGTH, and print_error().

Referenced by main().

Here is the call graph for this function:



5.1.3.16 static int validate_options (const char ∗const ∗ *parms*) `[static]`

validate_options This function examines the given command line arguments for existing options. in case the pattern is missing or wrong it prints information about expected arguments and quits the program

Parameters:

parms argument vector starting from the 3rd argument

Return values:

_arg_expected option masks if options are valid PRINT_OPT, LS_OPT, TYPE_OPT, NAME_OPT, USER_OPT

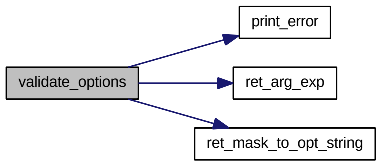
FALSE if options are not valid

Definition at line 606 of file myfind.c.

References FALSE, LS_OPT, MISSING_ARG, NAME_OPT, param_cnt, print_error(), PRINT_OPT, program, ret_arg_exp(), ret_mask_to_opt_string(), TRUE, TYPE_OPT, and USER_OPT.

Referenced by main().

Here is the call graph for this function:



5.1.4 Variable Documentation

5.1.4.1 int param_cnt = 0 `[static]`

param_cnt saves the argc and is used by practically all functions

Definition at line 101 of file myfind.c.

Referenced by do_file(), main(), and validate_options().

5.1.4.2 const char* program = "<not yet set>" `[static]`

The program is set to argv[0] in order for print_error() to access it without getting it as a parameter.

Definition at line 98 of file myfind.c.

Referenced by compare_type(), compare_user(), main(), print_error(), print_usage(), and validate_-options().

5.1.4.3 struct type_struct type_chars[ ] `[static]`

Initial value:

```
{
    { .type = "b", .type_val =  100 },
    { .type = "c", .type_val =  101 },
    { .type = "d", .type_val =  102     },
    { .type = "p", .type_val =  103 },
    { .type = "f", .type_val =  104 },
    { .type = "l", .type_val =  105 },
    { .type = "s", .type_val =  106 }
}
```

type_chars contains all valid types [bcdpfls] of a file and it is used in the function compare_type()

Definition at line 113 of file myfind.c.

5.1.4.4 struct opt_struct valid_options[ ] `[static]`

Initial value:

```
 {
    { .option = "-print", .opt_args = 0,.mask_val =  1005 },
    { .option = "-ls"   , .opt_args = 0,.mask_val =  1004 },
    { .option = "-type" , .opt_args = 1,.mask_val =  1002 },
    { .option = "-name" , .opt_args = 1,.mask_val =  1001 },
    { .option = "-user" , .opt_args = 1,.mask_val =  1003 }
}
```

valid_options contains all implemented action-strings, the number of expected arguments after each string, and a mask the option can be identified by

Definition at line 104 of file myfind.c.

Index