Project manager program

Generated by Doxygen 1.8.17

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

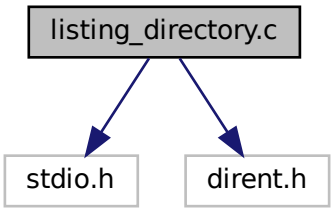
Chapter 2

File Documentation

2.1 listing_directory.c File Reference

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

Include dependency graph for listing_directory.c:



Functions

- int listing_directory (char *dirname)

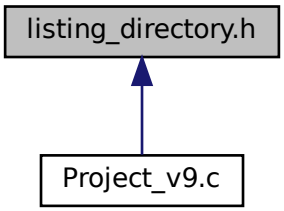
2.1.1 Function Documentation

2.1.1.1 listing_directory()

```
int listing_directory (
            char * dirname )
```

2.2 listing_directory.h File Reference

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



Functions

- int listing_directory (char ∗dirname)

2.2.1 Function Documentation

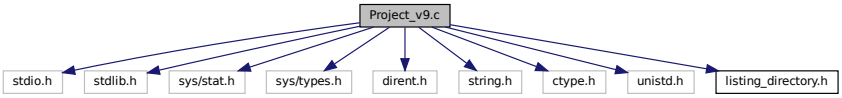
2.2.1.1 listing_directory()

```
int listing_directory (
            char * dirname )
```

2.3 Project_v9.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <dirent.h>
#include <string.h>
#include <ctype.h>
#include <unistd.h>
#include "listing_directory.h"
```

Include dependency graph for Project_v9.c:



Macros

- #define MAX_CHAR 50
- #define INPUT_CHAR 30

Functions

- void make_path (char result[ ], char path[ ], char fname[ ])
- int list_file_type (char type, char result[100][100], int index, char path[ ])
- int creating_directory (char ∗newdir)
- int main (int argc, char ∗argv[ ])

Variables

- const char ∗ slash = "/"

2.3.1 Macro Definition Documentation

2.3.1.1 INPUT_CHAR

```
#define INPUT_CHAR 30
```

2.3.1.2 MAX_CHAR

```
#define MAX_CHAR 50
```

Defining macros

2.3.2 Function Documentation

2.3.2.1 creating_directory()

```
int creating_directory (
            char * newdir )
```

Defining a function which creates a new folder structure

2.3.2.2 list_file_type()

```
int list_file_type (
             char type,
             char result[100][100],
             int index,
             char path[] )
```

Defining recursive file directory search function

2.3.2.3 main()

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

If only 1 argument supplied enter wizard version

Running program untill quit argument given

User input for options

Entering directory manager option if appropriate string supplied

Defining options for managing directories

Looping in directory manager option until appropriate string given

Listing current directory

Options for managing directories

User input for managing directory options

Removing directory option

Checking if input name exists

Creating directory option

Taking an input for a new directory

Clarifying a raw string value by discarding the "\n" in the end

If flag is activated - run in a loop

Checking for empty entries

Blocking clashing entries with current files

Renaming directory option

Storing input

Checking for existing directory

Averting empty entries

Averting empty entries

Moving directory option

Storing input

Checking for input directory existance

Checking for input directory existance

Moving up one level option

Storing input

Checking if directory exists

Moving down one level option

Composing git command

Checking for total project costs file

executing git

Calling a function for creating a directory with folders: bin, docs, lib, src, tests

Implementing time length counting system for the project

Feature 8 | If command "total_worktime" - total estimated hours to work on a project

iterating through the path's of directories inside current directory

Constructing a string

searching for a file

individual feature length in hours

Converting to type int

Putting found duration into the file

Reading total hour number

Listing current directory

Options for managing directories

User input for managing directory options

If command "add_tag" - adding a new tag

Establishing a pointer to the FILE structure

Checking for existing folders or files

Opening a file

Storing file's contents in a variable

Closing a file

Opening a file

Writing to a file

Closing a file

If command "find_tag" - finding a tag

Checking for existing folders or files

Opening a file

Setting file's position to the end of the stream

Storing current file's position in stream

If file's position more than 0 i.e. there is content

Rewinding file's position in stream

storing contents in a variable

If tag equals to input tag

marking tag's detection

Closing a file

If tag was not detected i.e. less than 1

iterating through the path's of directories inside current directory

Constructing a string

searching for a file

opening a file

Setting file's position to the end of the stream

Storing current file's position in a stream

If file's position more than 0 i.e. there is content

Rewinding file's position

Storing contents in a variable

If tag equals to input tag

Marking tag's detection

Closing a file

if tag not found

Storing input

Checking if directory exists

Installing plantuml

Creating a file

String construction

Length of the string

Copying string

Copying string

Iterating through the current directory's directories

Constructing the path

Copying string with limited length

Copying string

if input path contains part of input execute code

Iterating through the characters

If symbol / found

level of directory depth

Excluding bin, docs, lib, src and tests folders

Dividing a string by symbol "/" into tokens

Iterating through the string

Constructing the string

Constructing number of levels in the directory

Shifting 1 file's position

Constructing a final string in plantuml format

Writing string to file

Colsing the file

Processing txt file through plantuml

Feature 9 | If command "output_gantt" - creating gantt chart

Storing input

Checking for existing file

Installing plantuml

Creating a file

String construction

Length of the string

Copying string

Iterating through the current directory's directories

Constructing the path

Copying string with limited length

if input path contains part of input execute code

Excluding bin, docs, lib, src and tests folders

Dividing a string by symbol "/" into tokens

Iterating through the string

Constructing the string

Shifting 1 file's position

Constructing a final string in plantuml format

Opening a file

Writing string to file

Colsing the file

Processing txt file through plantuml

If specified directory not found

Creating new project

Creating new directory

Averting empty entries

Initializing git

CLI VERSION

Features 1,2,3,8 | If command "create_project" - creating new project

Checking for existing folders or files

Checking for unallowed number of arguments

Checking for unallowed spaces in the 3rd argument

Calling a function for creating a directory with folders: bin, docs, lib, src, tests

Initializing git

Features 1,2,3,8 | If command "add_feature" - creating new feature

Checking for existing folders or files

Checking for unallowed number of arguments

Checking for unallowed spaces in the 3rd argument

Composing git command

Checking for total project costs file

executing git

Calling a function for creating a directory with folders: bin, docs, lib, src, tests

Implementing time length counting system for the project

Feature 4 | If command "add_tag" - adding a new tag

Establishing a pointer to the FILE structure

Checking for existing folders or files

Opening a file

Storing file's contents in a variable

Closing a file

Checking for unallowed number of arguments

Checking for unallowed spaces in the 3rd argument

Opening a file

Constructing a string

Writing to a file

Closing a file

Feature 4 | If command "find_tag" - finding a tag

Checking for unallowed number of arguments

Checking for unallowed spaces in the 3rd argument

Checking for existing folders or files

Opening a file

Setting file's position to the end of the stream

Storing current file's position in stream

If file's position more than 0 i.e. there is content

Rewinding file's position in stream

storing contents in a variable

If tag equals to input tag

marking tag's detection

Closing a file

If tag was not detected i.e. less than 1

iterating through the path's of directories inside current directory

Constructing a string

searching for a file

opening a file

Setting file's position to the end of the stream

Storing current file's position in a stream

If file's position more than 0 i.e. there is content

Rewinding file's position

Storing contents in a variable

If tag equals to input tag

Marking tag's detection

Closing a file

if tag not found

Feature 5 | If command "rename_directory" - renaming existing directory

Checking for unallowed number of arguments

Checking for unallowed spaces in the 3rd and 4th arguments

Checking for existing folders or files

Renaming directory

If directory not found - Poping an error message

Feature 6 | If command "move_by_tag" - creating new project

Checking for unallowed number of arguments

Checking for unallowed spaces in the 3rd and 4th arguments

Checking for existing file

Opening a file

Setting file's position to the end of file

Storing file's position

If file's position more than zero

Rewinding file's position

Storing file's contents

If tag found in third argument

If tag found in fourth argument

Closing file

If tag was not found in current directory

iterating through the path's of directories inside current directory

Checking for tag's file

Opening and storing file's position

Checking if file has contents

Checking for tag in third argument

Constructing a string

Checking for tag in fourth argument

Constructing a string

Closing a file

if 0 tags found

if only second tag found

if only first tag found

if both tags found

dividing string by / into tokens

iterating through the tokens

Constructing the string

Transfering the directory

Feature 7 | If command "output_svg" - creating WBS tree diagram

Checking for unallowed number of arguments

Checking for unallowed spaces in the 3rd argument

Checking for existing file

Installing plantuml

Creating a file

String construction

Length of the string

Copying string

Copying string

Iterating through the current directory's directories

Constructing the path

Copying string with limited length

Copying string

if input path contains part of input execute code

Iterating through the characters

If symbol / found

level of directory depth

Excluding bin, docs, lib, src and tests folders

Dividing a string by symbol "/" into tokens

Iterating through the string

Constructing the string

Constructing number of levels in the directory

Shifting 1 file's position

Constructing a final string in plantuml format

Writing string to file

Colsing the file

Processing txt file through plantuml

If specified directory not found

Feature 8 | If command "total_worktime" - total estimated hours to work on a project

iterating through the path's of directories inside current directory

Constructing a string

searching for a file

individual feature length in hours

Converting to type int

Putting found duration into the file

Reading total hour number

Feature 9 | If command "output_gantt" - creating gantt chart

Checking for unallowed number of arguments

Checking for unallowed spaces in the 3rd argument

Checking for existing file

Installing plantuml

Creating a file

String construction

Length of the string

Copying string

Iterating through the current directory's directories

Constructing the path

Copying string with limited length

if input path contains part of input execute code

Excluding bin, docs, lib, src and tests folders

Dividing a string by symbol "/" into tokens

Iterating through the string

Constructing the string

Shifting 1 file's position

Constructing a final string in plantuml format

Opening a file

Writing string to file

Colsing the file

Processing txt file through plantuml

If specified directory not found

2.3.2.4 make_path()

```
void make_path (
             char result[],
             char path[],
             char fname[] )
```

2.3.3 Variable Documentation

2.3.3.1 slash

```
const char* slash = "/"
```

Defining a path construction function

Index