My Project

Generated by Doxygen 1.9.4

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 cmdargs Struct Reference

```
#include <cmdargs.h>
```

Public Attributes

- int x
- int y
- int verbose

3.1.1 Detailed Description

Holds all the possible command line arguments

3.1.2 Member Data Documentation

3.1.2.1 verbose

```
int cmdargs::verbose
```

An flag that if specified requests verbose output

3.1.2.2 x

```
int cmdargs::x
```

The first summand

3.1.2.3 y

```
int cmdargs::y
```

The second summand

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

- cmdargs.h

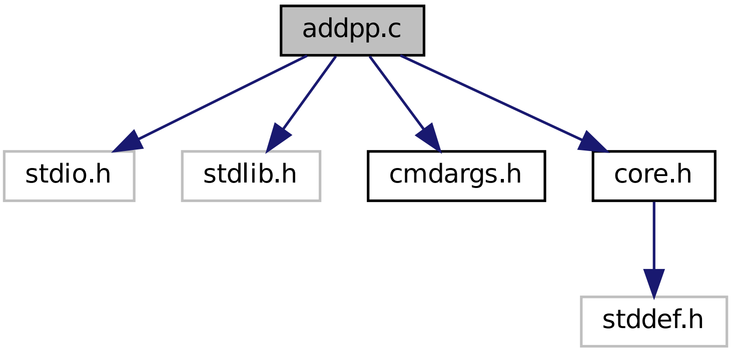
Chapter 4

File Documentation

4.1 addpp.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "cmdargs.h"
#include "core.h"
```

Include dependency graph for addpp.c:



Macros

- #define VERBOSE_LEN 100

Functions

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

4.1.1 Detailed Description

This is the main driver of the program, i.e., the program, which is then used by the user.

4.1.2 Macro Definition Documentation

4.1.2.1 VERBOSE_LEN

```
#define VERBOSE_LEN 100
```

The length of the verbose buffer.

Often, it is better not to use define like here but const variables. The define is used for demonstration purposes.

4.1.3 Function Documentation

4.1.3.1 main()

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

The progam's main entry point.

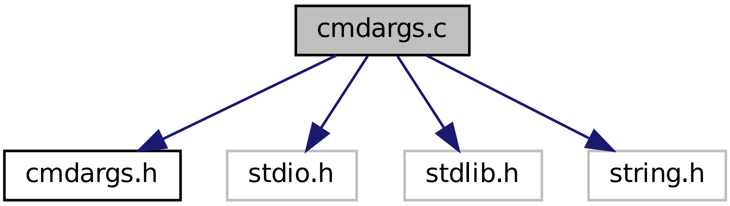
Parameters

argc	number of arguments given by the user
argv	the argument vector

4.2 cmdargs.c File Reference

```
#include "cmdargs.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for cmdargs.c:



Functions

- int cmdargs_parse (struct cmdargs ∗cmdargs, int argc, char ∗argv[ ])

4.2.1 Function Documentation

4.2.1.1 cmdargs_parse()

```
int cmdargs_parse (
            struct cmdargs * cmdargs,
            int argc,
            char * argv[ ] )
```

Parse the given command line arguments into the cmdargs struct.

Parameters

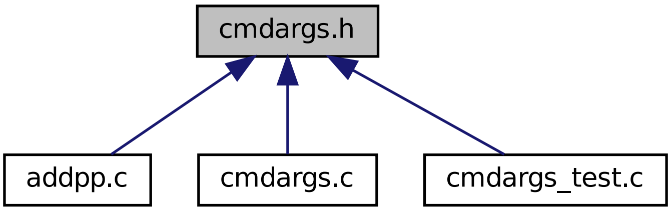
cmdargs	
argc	the size of argv as supplied in main().
argv	the the array of arguments as supplied in main().

Returns

1 on success, 0 on failure.

4.3 cmdargs.h File Reference

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



Classes

- struct cmdargs

Functions

- int cmdargs_parse (struct cmdargs ∗cmdargs, int argc, char ∗argv[ ])

4.3.1 Function Documentation

4.3.1.1 cmdargs_parse()

```
int cmdargs_parse (
            struct cmdargs * cmdargs,
            int argc,
            char * argv[ ] )
```

Parse the given command line arguments into the cmdargs struct.

Parameters

cmdargs	
argc	the size of argv as supplied in main().
argv	the the array of arguments as supplied in main().

Returns

1 on success, 0 on failure.

4.4 cmdargs.h

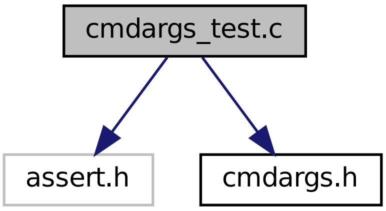
Go to the documentation of this file.

```

#ifndef _CMDARGS_H_
#define _CMDARGS_H_

struct cmdargs
{
  int x;
  int y;
  int verbose;
};

int cmdargs_parse(struct cmdargs *cmdargs, int argc, char *argv[]);

#endif
```

4.5 cmdargs_test.c File Reference

```
#include <assert.h>
#include "cmdargs.h"
```

Include dependency graph for cmdargs_test.c:



Functions

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

4.5.1 Detailed Description

Test for functions in core.c

4.5.2 Function Documentation

4.5.2.1 main()

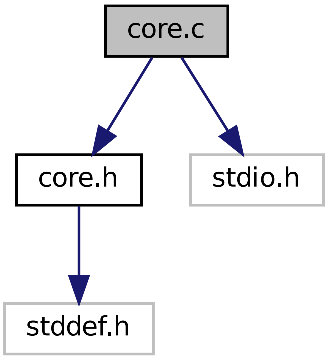
```
int main (
            int argc,
            char ** argv )
```

Main entry for the test.

4.6 core.c File Reference

```
#include "core.h"
#include <stdio.h>
```

Include dependency graph for core.c:



Functions

- int add (int x, int y, char ∗verbose, size_t verbose_len)

4.6.1 Detailed Description

Implements the core of the algorithm.

4.6.2 Function Documentation

4.6.2.1 add()

```
int add (
            int x,
            int y,
            char * verbose,
            size_t verbose_len )
```

This is a simple function to add two integer values. It main purpose is to demonstrate the testing.

Parameters

x	first summand
y	second summand
verbose	defines the pointer to a buffer where verbose output is written to
verbose_len	defines the length of the verbose buffer

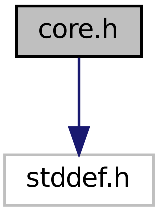
Returns

the sum of x and y

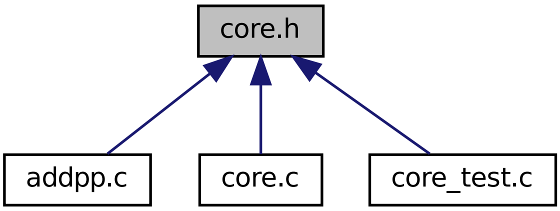
4.7 core.h File Reference

```
#include <stddef.h>
```

Include dependency graph for core.h:



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



Functions

- int add (int x, int y, char ∗verbose, size_t verbose_len)

4.7.1 Function Documentation

4.7.1.1 add()

```
int add (
             int x,
             int y,
             char * verbose,
             size_t verbose_len )
```

This is a simple function to add two integer values. It main purpose is to demonstrate the testing.

Parameters

x	first summand
y	second summand
verbose	defines the pointer to a buffer where verbose output is written to
verbose_len	defines the length of the verbose buffer

Returns

the sum of x and y

4.8 core.h

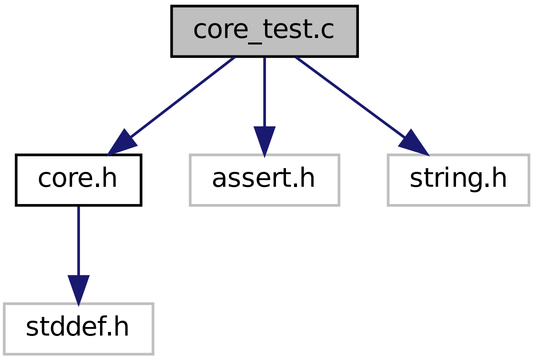
Go to the documentation of this file.

```

#ifndef _CORE_H
#define _CORE_H

#include <stddef.h>

int add(int x, int y, char *verbose, size_t verbose_len);

#endif
```

4.9 core_test.c File Reference

```
#include "core.h"
#include <assert.h>
#include <string.h>
```

Include dependency graph for core_test.c:



Functions

- int main (int argc, char ∗∗argv)

4.9.1 Detailed Description

Test for functions in core.c

4.9.2 Function Documentation

4.9.2.1 main()

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

Main entry for the test.

Index