0pt

Sample Reference Manual

Generated by Doxygen 1.2.18

Mon Sep 15 11:32:13 2003

Contents

Chapter 1

Sample documentation example

1.0.1 Introduction

This tutorial is a simple example of most of the features you will be using in doxygen. There are many more, but these are the most useful. Next see how you can use this yourself. Your own documentation

Chapter 2

Sample Module Index

2.1 Sample Modules

Here is a list of all modules:

Chapter 3

Sample Data Structure Index

3.1 Sample Data Structures

Here are the data structures with brief descriptions:

Chapter 4

Sample File Index

4.1 Sample File List

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

Chapter 5

Sample Page Index

5.1 Sample Related Pages

Here is a list of all related documentation pages:

Chapter 6

Sample Module Documentation

6.1 Stack

Data Structures

- struct stack

  This is a stack struct.

Functions

- int push (stack s, int val)

  This function is used to push stuff.

- int pop (stack s)

  This pops.

- int top (stack s)

  Take a look at the top.

6.1.1 Function Documentation

6.1.1.1 int pop (stack *s*)

This pops.

It is used to pop stuff

Warning:

This can give you an error if there is nothing to pop

See also:
stack , push

Parameters:
s the stack to get an element from

Returns:

6.1.1.2 int push (stack *s*, int *val*)

This function is used to push stuff.

Bug:
This function causes all kinds of buffer overflows. You should watch out. You should also note that the bug command can go before anything that is documented.

Examples:
stack_example.c.

Definition at line 13 of file stack.c.

References stack::end, and stack::stack.

6.1.1.3 int top (stack *s*)

Take a look at the top.

This is another way to do documentation. It is better to put the documentation in the source, because then you can edit them together.

Returns:
The top of the stack

Chapter 7

Sample Data Structure Documentation

7.1 stack Struct Reference

This is a stack struct.

```
#include <stack.h>
```

Data Fields

- int start

  This is the start of the stack.

- int end

  There are automatically brief descriptions.

- int stack [100]

  This is the data of the stack. This is a second line of brief description.

- int id

  ID number.

- void ∗(∗ helper)(int, int, char ∗)

7.1.1 Detailed Description

This is a stack struct.

It does stack stuff

Examples:
stack_example.c.

Definition at line 45 of file stack.h.

7.1.2 Field Documentation

7.1.2.1 void∗(∗ stack::helper)(int, int, char ∗)

It handles these good too. This is a pointer to a function that takes 2 ints and a char ∗ and returns a void ∗

7.1.2.2 int stack::stack[100]

This is the data of the stack. This is a second line of brief description.

If you do it this way, they dont only have to be brief descriptions

Definition at line 58 of file stack.h.

Referenced by push().

7.1.2.3 int stack::start

This is the start of the stack.

and multiline too, with details.

Definition at line 47 of file stack.h.

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

- stack.h

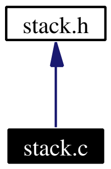
Chapter 8

Sample File Documentation

8.1 stack.c File Reference

```
#include "stack.h"
```

Include dependency graph for stack.c:



Functions

- int push (stack s, int val)

    This function is used to push stuff.

- int pop (stack s, int val)

8.1.1 Detailed Description

Definition in file stack.c.

8.1.2 Function Documentation

8.1.2.1 int pop (stack *s*, int *val*)

Todo:
Impliment this function

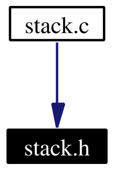
Examples:
stack_example.c.

Definition at line 24 of file stack.c.

8.2 stack.h File Reference

This is brief description of this file.

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



Data Structures

- struct stack

  This is a stack struct.

Defines

- #define ENOMEM -1
- #define EXAMPLE -2

  you may also only have one line of comments

- #define EXAMPLE2 -3

  This is how you put documentation after the fact.

Functions

- int push (stack s, int val)

  This function is used to push stuff.

- int pop (stack s)

  This pops.

- int get_elem (stack s, int val)

  This gets an element.

- int top (stack s)

  Take a look at the top.

8.2.1 Detailed Description

This is brief description of this file.

This is a slightly longer description of the file. Also note that the 'backslash'file command is needed or doxygen wont pull any declarations from the file

Version:
a million

Author:
someone without much to do , some other guy too

Date:
1896-2430

Definition in file stack.h.

8.2.2 Define Documentation

8.2.2.1 #define ENOMEM -1

This describes this define and what it is used for. In this case it is used to say there is no memory. These are automatically brief descriptions

Definition at line 26 of file stack.h.

8.2.3 Function Documentation

8.2.3.1 int get_elem (stack *s*, int *val*)

This gets an element.

It is used to get stuff

Deprecated:
Dont use this anymore, for various reasons

Parameters:
s stack to get item from

i This is the index

Returns:

8.2.3.2 int push (stack *s*, int *val*)

This function is used to push stuff.

Bug:

This function causes all kinds of buffer overflows. You should watch out. You should also note that the bug command can go before anything that is documented.

Definition at line 13 of file stack.c.

References stack::end, and stack::stack.

Chapter 9

Sample Example Documentation

9.1 stack_example.c

This is an example of how to use the stack struct

```
void main()
{
    stack s;
    int y;
    push(s, 3);
    y = pop(s);  //y will be 3
}
```

Chapter 10

Sample Page Documentation

10.1 Your own documentation

You just need to start putting these structured comments in your .h files and things will be good.

10.1.1 Step 1: making .h files

10.1.1.1 Type stuff

First, type some stuff

10.1.1.2 Add documentation

Then add documentation.

10.1.1.3 Compile.

Then compile

10.2 Todo List

Global pop(stack s, int val) Impliment this function

10.3 Deprecated List

Global get_elem(stack s, int val) Dont use this anymore, for various reasons

10.4 Bug List

Global push(stack s, int val) This function causes all kinds of buffer overflows. You should watch out. You should also note that the bug command can go before anything that is documented.