Mp

Generated by Doxygen 1.8.13

Contents

1 Project Mp **1**

2 Data Structure Index **3**

2.1 Data Structures 3

3 File Index **5**

3.1 File List 5

4 Data Structure Documentation **7**

4.1 Array Struct Reference 7

4.1.1 Field Documentation 7

4.1.1.1 array 7

4.1.1.2 size 7

4.1.1.3 used 7

4.2 AxesDBL Struct Reference 8

4.2.1 Field Documentation 8

4.2.1.1 ctxfactor 8

4.2.1.2 ctyfactor 8

4.2.1.3 xmax 8

4.2.1.4 xmin 8

4.2.1.5 ymax 8

4.2.1.6 ymin 9

4.3 AxesFLT Struct Reference 9

4.3.1 Field Documentation 9

4.3.1.1 ctxfactor 9

4.3.1.2 ctyfactor . . . 9
4.3.1.3 xmax . . . 9
4.3.1.4 xmin . . . 9
4.3.1.5 ymax . . . 10
4.3.1.6 ymin . . . 10
4.4 AxesFLT128 Struct Reference . . . 10
4.4.1 Field Documentation . . . 10
4.4.1.1 ctxfactor . . . 10
4.4.1.2 ctyfactor . . . 10
4.4.1.3 xmax . . . 10
4.4.1.4 xmin . . . 11
4.4.1.5 ymax . . . 11
4.4.1.6 ymin . . . 11
4.5 AxesLDBL Struct Reference . . . 11
4.5.1 Field Documentation . . . 11
4.5.1.1 ctxfactor . . . 11
4.5.1.2 ctyfactor . . . 11
4.5.1.3 xmax . . . 12
4.5.1.4 xmin . . . 12
4.5.1.5 ymax . . . 12
4.5.1.6 ymin . . . 12
4.6 cJSON Struct Reference . . . 12
4.6.1 Field Documentation . . . 12
4.6.1.1 child . . . 13
4.6.1.2 next . . . 13
4.6.1.3 prev . . . 13
4.6.1.4 string . . . 13
4.6.1.5 type . . . 13
4.6.1.6 valuedouble . . . 13
4.6.1.7 valueint . . . 13

4.6.1.8 valuestring . . . 14
4.7 cJSON_Hooks Struct Reference . . . 14
4.8 error Struct Reference . . . 14
4.8.1 Field Documentation . . . 14
4.8.1.1 json . . . 14
4.8.1.2 position . . . 14
4.9 HSV Struct Reference . . . 15
4.9.1 Field Documentation . . . 15
4.9.1.1 H . . . 15
4.9.1.2 S . . . 15
4.9.1.3 V . . . 15
4.10 internal_hooks Struct Reference . . . 15
4.11 Parameters Struct Reference . . . 16
4.11.1 Field Documentation . . . 16
4.11.1.1 aa . . . 16
4.11.1.2 centerX . . . 16
4.11.1.3 centerY . . . 16
4.11.1.4 color . . . 16
4.11.1.5 config . . . 17
4.11.1.6 cv . . . 17
4.11.1.7 diameter . . . 17
4.11.1.8 filename . . . 17
4.11.1.9 height . . . 17
4.11.1.10 magnify . . . 17
4.11.1.11 maxiter . . . 17
4.11.1.12 next . . . 17
4.11.1.13 palname . . . 18
4.11.1.14 tweak . . . 18
4.11.1.15 width . . . 18
4.12 parse_buffer Struct Reference . . . 18

4.12.1 Field Documentation . . . 18
4.12.1.1 content . . . 18
4.12.1.2 depth . . . 18
4.12.1.3 hooks . . . 19
4.12.1.4 length . . . 19
4.12.1.5 offset . . . 19
4.13 printbuffer Struct Reference . . . 19
4.13.1 Field Documentation . . . 19
4.13.1.1 buffer . . . 19
4.13.1.2 depth . . . 19
4.13.1.3 format . . . 20
4.13.1.4 hooks . . . 20
4.13.1.5 length . . . 20
4.13.1.6 noalloc . . . 20
4.13.1.7 offset . . . 20
4.14 Rgb Struct Reference . . . 20
4.14.1 Field Documentation . . . 20
4.14.1.1 b . . . 21
4.14.1.2 g . . . 21
4.14.1.3 r . . . 21

5 File Documentation **23**

5.1 dictionary.c File Reference . . . 23

5.1.1 Detailed Description . . . 24

5.1.2 Macro Definition Documentation . . . 24

5.1.2.1 DICT_INVALID_KEY . . . 24

5.1.2.2 DICTMINSZ . . . 24

5.1.2.3 MAXVALSZ . . . 24

5.1.3 Function Documentation . . . 24

5.1.3.1 dictionary_del() . . . 24

5.1.3.2 dictionary_dump() . . . 25

5.1.3.3 dictionary_get() . . . 25

5.1.3.4 dictionary_hash() . . . 26

5.1.3.5 dictionary_new() . . . 26

5.1.3.6 dictionary_set() . . . 26

5.1.3.7 dictionary_unset() . . . 27

5.2 dictionary.h File Reference . . . 27

5.2.1 Detailed Description . . . 28

5.2.2 Data Structure Documentation . . . 28

5.2.2.1 struct dictionary . . . 28

5.2.3 Function Documentation . . . 28

5.2.3.1 dictionary_del() . . . 29

5.2.3.2 dictionary_dump() . . . 29

5.2.3.3 dictionary_get() . . . 29

5.2.3.4 dictionary_hash() . . . 30

5.2.3.5 dictionary_new() . . . 30

5.2.3.6 dictionary_set() . . . 30

5.2.3.7 dictionary_unset() . . . 31

5.3 iniparser.c File Reference . . . 31

5.3.1 Detailed Description . . . 33

5.3.2 Enumeration Type Documentation . . . 33

5.3.2.1 line_status . . . 33
5.3.3 Function Documentation . . . 33
5.3.3.1 iniparser_dump() . . . 33
5.3.3.2 iniparser_dump_ini() . . . 33
5.3.3.3 iniparser_dumpsection_ini() . . . 35
5.3.3.4 iniparser_find_entry() . . . 35
5.3.3.5 iniparser_freedict() . . . 36
5.3.3.6 iniparser_getboolean() . . . 36
5.3.3.7 iniparser_getdouble() . . . 37
5.3.3.8 iniparser_getint() . . . 37
5.3.3.9 iniparser_getlongdouble() . . . 38
5.3.3.10 iniparser_getlongint() . . . 38
5.3.3.11 iniparser_getnsec() . . . 39
5.3.3.12 iniparser_getseckeys() . . . 40
5.3.3.13 iniparser_getsecname() . . . 40
5.3.3.14 iniparser_getsecnkeys() . . . 40
5.3.3.15 iniparser_getstring() . . . 41
5.3.3.16 iniparser_load() . . . 41
5.3.3.17 iniparser_set() . . . 42
5.3.3.18 iniparser_unset() . . . 42
5.4 iniparser.h File Reference . . . 43
5.4.1 Detailed Description . . . 44
5.4.2 Function Documentation . . . 44
5.4.2.1 iniparser_dump() . . . 44
5.4.2.2 iniparser_dump_ini() . . . 44
5.4.2.3 iniparser_dumpsection_ini() . . . 45
5.4.2.4 iniparser_find_entry() . . . 45
5.4.2.5 iniparser_freedict() . . . 46
5.4.2.6 iniparser_getboolean() . . . 46
5.4.2.7 iniparser_getdouble() . . . 47
5.4.2.8 iniparser_getint() . . . 47
5.4.2.9 iniparser_getlongdouble() . . . 48
5.4.2.10 iniparser_getlongint() . . . 49
5.4.2.11 iniparser_getnsec() . . . 50
5.4.2.12 iniparser_getseckeys() . . . 51
5.4.2.13 iniparser_getsecname() . . . 51
5.4.2.14 iniparser_getsecnkeys() . . . 52
5.4.2.15 iniparser_getstring() . . . 52
5.4.2.16 iniparser_load() . . . 52
5.4.2.17 iniparser_set() . . . 53
5.4.2.18 iniparser_set_error_callback() . . . 53
5.4.2.19 iniparser_unset() . . . 54

Index 55

Chapter 1

Project Mp

Intro

- Succently, this is a fractal play pen.
- It is *not* a library.
- There is no ∗∗_API_∗∗
- List its most useful/innovative/noteworthy features.
  - feature one
  - feature two
- State its goals/what problem(s) it solves.
  - Dust off my programming skills.
  - As much as possible work with what's out there. . .
  - Continue working with ∗fractals∗—probably the real point.
- Key concepts.
  - concept one
  - concept two
- This is now and always will be *alpha*.
- Does not include badges.

Core Technical Concepts/Inspiration

- Why does it exist?
- Frame your project for the potential user.
- Compare/contrast your project with other, similar projects so the user knows how it is different from those projects.
- Highlight the technical concepts that your project demonstrates or supports. Keep it very brief.

Getting Started/Requirements/Prerequisites/Dependencies

Include any essential instructions for:

- Getting it
- Installing It
- Configuring It
- Running it

Mini-Manual...

```
void help( char c, char *Program, char *Version, char *Date ) {
    printf( "%s v%s dated %s\n", Program, Version, Date );
    if ( c == 'h' || c == '?' ) {
        printf( "\n  Options:\n\n" );
        printf( "  --x_center    requires real as an argument   -x\n" );
        printf( "  --y_center    requires real as an argument   -y\n" );
        printf( "  --magnify     requires real as an argument   -m\n" );
        printf( "  --diameter    requires real as an argument   -d\n" );
        printf( "  --iteration   requires number as an argument -i\n" );
        printf( "  --width       requires number as an argument -w\n" );
        printf( "  --height      requires number as an argument -l\n" );
        printf( "  --file        requires string as an argument -f\n" );
        printf( "  --palette     requires string as an argument -p\n" );
        printf( "  --next        argument is optional number    -n\n" );
        printf( "  --config      requires string as an argument -c\n" );
        printf( "  --color       requires number as an argument -r\n" );
        printf( "  --aa          requires string as an argument -a\n" );
        printf( "  --tweak       requires number as an argument -t\n" );
        printf( "  --version     no argument                    -v\n" );
        printf( "  --help        no argument                    -h\n" );
        printf( "\n  long options ('--' prefix) are incremental till unambiguous\n" );
        printf( "  short options ('-' prefix) are exact\n" );
        printf( "  Hideously enough, options with optional arguments,\n" );
        printf( "  take the form [-[-]]option=arg, no spaces.\n" );
    }
    exit( 0 );
}
```

Contributing

- Chris Thomasson
- Greg Harley

TODO

- Next steps
- Features planned
- Known bugs (shortlist)

Contact

- hsmyers@gmail.com
- http://www.sdragons.org/

License

- see file License.txt

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Array 7
AxesDBL 8
AxesFLT 9
AxesFLT128 10
AxesLDBL 11
cJSON 12
cJSON_Hooks 14
error 14
HSV 15
internal_hooks 15
Parameters 16
parse_buffer 18
printbuffer 19
Rgb 20

Chapter 3

File Index

3.1 File List

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

cJSON.h . . . . . **??**
colors.h . . . . . **??**
cspace.h . . . . . **??**
dictionary.c
Implements a dictionary for string variables 23
dictionary.h
Implements a dictionary for string variables 27
elapsed.h . . . . . **??**
getopt.h . . . . . **??**
iniparser.c
Parser for ini files 31
iniparser.h
Parser for ini files 43
palette.h . . . . . **??**
util.h . . . . . **??**

Chapter 4

Data Structure Documentation

4.1 Array Struct Reference

Data Fields

- Rgb * **array**
- size_t **used**
- size_t **size**

4.1.1 Field Documentation

4.1.1.1 array

```
Rgb* Array::array
```

4.1.1.2 size

```
size_t Array::size
```

4.1.1.3 used

```
size_t Array::used
```

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

- palette.h

4.2 AxesDBL Struct Reference

Data Fields

- double **xmin**
- double **xmax**
- double **ymin**
- double **ymax**
- double **ctxfactor**
- double **ctyfactor**

4.2.1 Field Documentation

4.2.1.1 ctxfactor

```
double AxesDBL::ctxfactor
```

4.2.1.2 ctyfactor

```
double AxesDBL::ctyfactor
```

4.2.1.3 xmax

```
double AxesDBL::xmax
```

4.2.1.4 xmin

```
double AxesDBL::xmin
```

4.2.1.5 ymax

```
double AxesDBL::ymax
```

4.2.1.6 ymin

```
double AxesDBL::ymin
```

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

- mp.c

4.3 AxesFLT Struct Reference

Data Fields

- float **xmin**
- float **xmax**
- float **ymin**
- float **ymax**
- float **ctxfactor**
- float **ctyfactor**

4.3.1 Field Documentation

4.3.1.1 ctxfactor

```
float AxesFLT::ctxfactor
```

4.3.1.2 ctyfactor

```
float AxesFLT::ctyfactor
```

4.3.1.3 xmax

```
float AxesFLT::xmax
```

4.3.1.4 xmin

```
float AxesFLT::xmin
```

4.3.1.5 ymax

```
float AxesFLT::ymax
```

4.3.1.6 ymin

```
float AxesFLT::ymin
```

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

- mp.c

4.4 AxesFLT128 Struct Reference

Data Fields

- __float128 **xmin**
- __float128 **xmax**
- __float128 **ymin**
- __float128 **ymax**
- __float128 **ctxfactor**
- __float128 **ctyfactor**

4.4.1 Field Documentation

4.4.1.1 ctxfactor

```
__float128 AxesFLT128::ctxfactor
```

4.4.1.2 ctyfactor

```
__float128 AxesFLT128::ctyfactor
```

4.4.1.3 xmax

```
__float128 AxesFLT128::xmax
```

4.4.1.4 xmin

```
__float128 AxesFLT128::xmin
```

4.4.1.5 ymax

```
__float128 AxesFLT128::ymax
```

4.4.1.6 ymin

```
__float128 AxesFLT128::ymin
```

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

- mp.c

4.5 AxesLDBL Struct Reference

Data Fields

- long double **xmin**
- long double **xmax**
- long double **ymin**
- long double **ymax**
- long double **ctxfactor**
- long double **ctyfactor**

4.5.1 Field Documentation

4.5.1.1 ctxfactor

```
long double AxesLDBL::ctxfactor
```

4.5.1.2 ctyfactor

```
long double AxesLDBL::ctyfactor
```

4.5.1.3 xmax

```
long double AxesLDBL::xmax
```

4.5.1.4 xmin

```
long double AxesLDBL::xmin
```

4.5.1.5 ymax

```
long double AxesLDBL::ymax
```

4.5.1.6 ymin

```
long double AxesLDBL::ymin
```

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

- mp.c

4.6 cJSON Struct Reference

Data Fields

- struct cJSON * **next**
- struct cJSON * **prev**
- struct cJSON * **child**
- int **type**
- char * **valuestring**
- int **valueint**
- double **valuedouble**
- char * **string**

4.6.1 Field Documentation

4.6.1.1 child

```
struct cJSON* cJSON::child
```

4.6.1.2 next

```
struct cJSON* cJSON::next
```

4.6.1.3 prev

```
struct cJSON* cJSON::prev
```

4.6.1.4 string

```
char* cJSON::string
```

4.6.1.5 type

```
int cJSON::type
```

4.6.1.6 valuedouble

```
double cJSON::valuedouble
```

4.6.1.7 valueint

```
int cJSON::valueint
```

4.6.1.8 valuestring

```
char* cJSON::valuestring
```

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

- cJSON.h

4.7 cJSON_Hooks Struct Reference

Data Fields

- void *(* **malloc_fn**)(size_t sz)
- void(* **free_fn**)(void *ptr)

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

- cJSON.h

4.8 error Struct Reference

Data Fields

- const unsigned char * **json**
- size_t **position**

4.8.1 Field Documentation

4.8.1.1 json

```
const unsigned char* error::json
```

4.8.1.2 position

```
size_t error::position
```

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

- cJSON.c

4.9 HSV Struct Reference

Data Fields

- double **H**
- double **S**
- double **V**

4.9.1 Field Documentation

4.9.1.1 H

```
double HSV::H
```

4.9.1.2 S

```
double HSV::S
```

4.9.1.3 V

```
double HSV::V
```

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

- cspace.h
- mp.c

4.10 internal_hooks Struct Reference

Data Fields

- void ∗(∗ **allocate**)(size_t size)
- void(∗ **deallocate**)(void ∗pointer)
- void ∗(∗ **reallocate**)(void ∗pointer, size_t size)

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

- cJSON.c

4.11 Parameters Struct Reference

Data Fields

- long double **centerX**
- long double **centerY**
- long double **magnify**
- long double **diameter**
- int **maxiter**
- int **cv**
- int **width**
- int **height**
- char * **filename**
- char * **palname**
- char * **config**
- char * **aa**
- int **next**
- int **color**
- int **tweak**

4.11.1 Field Documentation

4.11.1.1 aa

```
char* Parameters::aa
```

4.11.1.2 centerX

```
long double Parameters::centerX
```

4.11.1.3 centerY

```
long double Parameters::centerY
```

4.11.1.4 color

```
int Parameters::color
```

4.11.1.5 config

```
char* Parameters::config
```

4.11.1.6 cv

```
int Parameters::cv
```

4.11.1.7 diameter

```
long double Parameters::diameter
```

4.11.1.8 filename

```
char* Parameters::filename
```

4.11.1.9 height

```
int Parameters::height
```

4.11.1.10 magnify

```
long double Parameters::magnify
```

4.11.1.11 maxiter

```
int Parameters::maxiter
```

4.11.1.12 next

```
int Parameters::next
```

4.11.1.13 palname

```
char* Parameters::palname
```

4.11.1.14 tweak

```
int Parameters::tweak
```

4.11.1.15 width

```
int Parameters::width
```

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

- getopt.h

4.12 parse_buffer Struct Reference

Data Fields

- const unsigned char * **content**
- size_t **length**
- size_t **offset**
- size_t **depth**
- internal_hooks **hooks**

4.12.1 Field Documentation

4.12.1.1 content

```
const unsigned char* parse_buffer::content
```

4.12.1.2 depth

```
size_t parse_buffer::depth
```

4.12.1.3 hooks

```
internal_hooks parse_buffer::hooks
```

4.12.1.4 length

```
size_t parse_buffer::length
```

4.12.1.5 offset

```
size_t parse_buffer::offset
```

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

- cJSON.c

4.13 printbuffer Struct Reference

Data Fields

- unsigned char * **buffer**
- size_t **length**
- size_t **offset**
- size_t **depth**
- cJSON_bool **noalloc**
- cJSON_bool **format**
- internal_hooks **hooks**

4.13.1 Field Documentation

4.13.1.1 buffer

```
unsigned char* printbuffer::buffer
```

4.13.1.2 depth

```
size_t printbuffer::depth
```

4.13.1.3 format

```
cJSON_bool printbuffer::format
```

4.13.1.4 hooks

```
internal_hooks printbuffer::hooks
```

4.13.1.5 length

```
size_t printbuffer::length
```

4.13.1.6 noalloc

```
cJSON_bool printbuffer::noalloc
```

4.13.1.7 offset

```
size_t printbuffer::offset
```

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

- cJSON.c

4.14 Rgb Struct Reference

Data Fields

- unsigned char **r**
- unsigned char **g**
- unsigned char **b**

4.14.1 Field Documentation

4.14.1.1 b

```
unsigned char Rgb::b
```

4.14.1.2 g

```
unsigned char Rgb::g
```

4.14.1.3 r

```
unsigned char Rgb::r
```

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

- palette.h

Chapter 5

File Documentation

5.1 dictionary.c File Reference

Implements a dictionary for string variables.

```
#include "dictionary.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
```

Macros

- #define MAXVALSZ 1024
- #define DICTMINSZ 128
- #define DICT_INVALID_KEY ((char*)-1)

Functions

- unsigned dictionary_hash (const char *key)

  Compute the hash key for a string.
- dictionary * dictionary_new (size_t size)

  Create a new dictionary object.
- void dictionary_del (dictionary *d)

  Delete a dictionary object.
- const char * dictionary_get (const dictionary *d, const char *key, const char *def)

  Get a value from a dictionary.
- int dictionary_set (dictionary *d, const char *key, const char *val)

  Set a value in a dictionary.
- void dictionary_unset (dictionary *d, const char *key)

  Delete a key in a dictionary.
- void dictionary_dump (const dictionary *d, FILE *out)

  Dump a dictionary to an opened file pointer.

5.1.1 Detailed Description

Implements a dictionary for string variables.

Author

N. Devillard This module implements a simple dictionary object, i.e. a list of string/string associations. This object is useful to store e.g. informations retrieved from a configuration file (ini files).

5.1.2 Macro Definition Documentation

5.1.2.1 DICT_INVALID_KEY

```
#define DICT_INVALID_KEY ((char*)-1)
```

Invalid key token

5.1.2.2 DICTMINSZ

```
#define DICTMINSZ 128
```

Minimal allocated number of entries in a dictionary

5.1.2.3 MAXVALSZ

```
#define MAXVALSZ 1024
```

Maximum value size for integers and doubles.

5.1.3 Function Documentation

5.1.3.1 dictionary_del()

```
void dictionary_del (
            dictionary * d )
```

Delete a dictionary object.

Parameters

d	dictionary object to deallocate.

Returns

void

Deallocate a dictionary object and all memory associated to it.

5.1.3.2 dictionary_dump()

```
void dictionary_dump (
            const dictionary * d,
            FILE * out )
```

Dump a dictionary to an opened file pointer.

Parameters

d	Dictionary to dump
f	Opened file pointer.

Returns

void

Dumps a dictionary onto an opened file pointer. Key pairs are printed out as [Key]=[Value], one per line. It is Ok to provide stdout or stderr as output file pointers.

5.1.3.3 dictionary_get()

```
const char* dictionary_get (
            const dictionary * d,
            const char * key,
            const char * def )
```

Get a value from a dictionary.

Parameters

d	dictionary object to search.
key	Key to look for in the dictionary.
def	Default value to return if key not found.

Returns

1 pointer to internally allocated character string.

This function locates a key in a dictionary and returns a pointer to its value, or the passed 'def' pointer if no such key can be found in dictionary. The returned character pointer points to data internal to the dictionary object, you should not try to free it or modify it.

5.1.3.4 dictionary_hash()

```
unsigned dictionary_hash (
            const char * key )
```

Compute the hash key for a string.

Parameters

key	Character string to use for key.

Returns

1 unsigned int on at least 32 bits.

This hash function has been taken from an Article in Dr Dobbs Journal. This is normally a collision-free function, distributing keys evenly. The key is stored anyway in the struct so that collision can be avoided by comparing the key itself in last resort.

5.1.3.5 dictionary_new()

```
dictionary* dictionary_new (
            size_t size )
```

Create a new dictionary object.

Parameters

size	Optional initial size of the dictionary.

Returns

1 newly allocated dictionary objet.

This function allocates a new dictionary object of given size and returns it. If you do not know in advance (roughly) the number of entries in the dictionary, give size=0.

5.1.3.6 dictionary_set()

```
int dictionary_set (
            dictionary * d,
            const char * key,
            const char * val )
```

Set a value in a dictionary.

Parameters

d	dictionary object to modify.
key	Key to modify or add.
val	Value to add.

Returns

int 0 if Ok, anything else otherwise

If the given key is found in the dictionary, the associated value is replaced by the provided one. If the key cannot be found in the dictionary, it is added to it.

It is Ok to provide a NULL value for val, but NULL values for the dictionary or the key are considered as errors: the function will return immediately in such a case.

Notice that if you dictionary_set a variable to NULL, a call to dictionary_get will return a NULL value: the variable will be found, and its value (NULL) is returned. In other words, setting the variable content to NULL is equivalent to deleting the variable from the dictionary. It is not possible (in this implementation) to have a key in the dictionary without value.

This function returns non-zero in case of failure.

5.1.3.7 dictionary_unset()

```
void dictionary_unset (
            dictionary * d,
            const char * key )
```

Delete a key in a dictionary.

Parameters

d	dictionary object to modify.
key	Key to remove.

Returns

void

This function deletes a key in a dictionary. Nothing is done if the key cannot be found.

5.2 dictionary.h File Reference

Implements a dictionary for string variables.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
```

Data Structures

- struct dictionary

  Dictionary object. *More...*

Functions

- unsigned dictionary_hash (const char *key)

  Compute the hash key for a string.
- dictionary * dictionary_new (size_t size)

  Create a new dictionary object.
- void dictionary_del (dictionary *vd)

  Delete a dictionary object.
- const char * dictionary_get (const dictionary *d, const char *key, const char *def)

  Get a value from a dictionary.
- int dictionary_set (dictionary *vd, const char *key, const char *val)

  Set a value in a dictionary.
- void dictionary_unset (dictionary *d, const char *key)

  Delete a key in a dictionary.
- void dictionary_dump (const dictionary *d, FILE *out)

  Dump a dictionary to an opened file pointer.

5.2.1 Detailed Description

Implements a dictionary for string variables.

Author

N. Devillard This module implements a simple dictionary object, i.e. a list of string/string associations. This object is useful to store e.g. informations retrieved from a configuration file (ini files).

5.2.2 Data Structure Documentation

5.2.2.1 struct dictionary

Dictionary object.

This object contains a list of string/string associations. Each association is identified by a unique string key. Looking up values in the dictionary is speeded up by the use of a (hopefully collision-free) hash function.

Data Fields

unsigned *	hash	List of string keys
char **	key	List of string values
int	n	
ssize_t	size	Number of entries in dictionary
char **	val	Storage size

5.2.3 Function Documentation

5.2.3.1 dictionary_del()

```
void dictionary_del (
            dictionary * d )
```

Delete a dictionary object.

Parameters

d	dictionary object to deallocate.

Returns

void

Deallocate a dictionary object and all memory associated to it.

5.2.3.2 dictionary_dump()

```
void dictionary_dump (
            const dictionary * d,
            FILE * out )
```

Dump a dictionary to an opened file pointer.

Parameters

d	Dictionary to dump
f	Opened file pointer.

Returns

void

Dumps a dictionary onto an opened file pointer. Key pairs are printed out as [Key]=[Value], one per line. It is Ok to provide stdout or stderr as output file pointers.

5.2.3.3 dictionary_get()

```
const char* dictionary_get (
            const dictionary * d,
            const char * key,
            const char * def )
```

Get a value from a dictionary.

Parameters

d	dictionary object to search.
key	Key to look for in the dictionary.
def	Default value to return if key not found.

Returns

1 pointer to internally allocated character string.

This function locates a key in a dictionary and returns a pointer to its value, or the passed 'def' pointer if no such key can be found in dictionary. The returned character pointer points to data internal to the dictionary object, you should not try to free it or modify it.

5.2.3.4 dictionary_hash()

```
unsigned dictionary_hash (
            const char * key )
```

Compute the hash key for a string.

Parameters

key	Character string to use for key.

Returns

1 unsigned int on at least 32 bits.

This hash function has been taken from an Article in Dr Dobbs Journal. This is normally a collision-free function, distributing keys evenly. The key is stored anyway in the struct so that collision can be avoided by comparing the key itself in last resort.

5.2.3.5 dictionary_new()

```
dictionary* dictionary_new (
            size_t size )
```

Create a new dictionary object.

Parameters

size	Optional initial size of the dictionary.

Returns

1 newly allocated dictionary objet.

This function allocates a new dictionary object of given size and returns it. If you do not know in advance (roughly) the number of entries in the dictionary, give size=0.

5.2.3.6 dictionary_set()

```
int dictionary_set (
            dictionary * d,
```

```
            const char * key,
            const char * val )
```

Set a value in a dictionary.

Parameters

d	dictionary object to modify.
key	Key to modify or add.
val	Value to add.

Returns

int 0 if Ok, anything else otherwise

If the given key is found in the dictionary, the associated value is replaced by the provided one. If the key cannot be found in the dictionary, it is added to it.

It is Ok to provide a NULL value for val, but NULL values for the dictionary or the key are considered as errors: the function will return immediately in such a case.

Notice that if you dictionary_set a variable to NULL, a call to dictionary_get will return a NULL value: the variable will be found, and its value (NULL) is returned. In other words, setting the variable content to NULL is equivalent to deleting the variable from the dictionary. It is not possible (in this implementation) to have a key in the dictionary without value.

This function returns non-zero in case of failure.

5.2.3.7 dictionary_unset()

```
void dictionary_unset (
            dictionary * d,
            const char * key )
```

Delete a key in a dictionary.

Parameters

d	dictionary object to modify.
key	Key to remove.

Returns

void

This function deletes a key in a dictionary. Nothing is done if the key cannot be found.

5.3 iniparser.c File Reference

Parser for ini files.

```
#include <ctype.h>
#include <stdarg.h>
#include "iniparser.h"
```

Macros

- #define **__USE_MINGW_ANSI_STDIO** 1
- #define **ASCIILINESZ** (1024)
- #define **INI_INVALID_KEY** ((char∗)-1)

Enumerations

- enum line_status {
  LINE_UNPROCESSED, **LINE_ERROR**, **LINE_EMPTY**, **LINE_COMMENT**,
  LINE_SECTION, **LINE_VALUE** }

Functions

- int iniparser_getnsec (const dictionary ∗d)
  Get number of sections in a dictionary.
- const char ∗ iniparser_getsecname (const dictionary ∗d, int n)
  Get name for section n in a dictionary.
- void iniparser_dump (const dictionary ∗d, FILE ∗f)
  Dump a dictionary to an opened file pointer.
- void iniparser_dump_ini (const dictionary ∗d, FILE ∗f)
  Save a dictionary to a loadable ini file.
- void iniparser_dumpsection_ini (const dictionary ∗d, const char ∗s, FILE ∗f)
  Save a dictionary section to a loadable ini file.
- int iniparser_getsecnkeys (const dictionary ∗d, const char ∗s)
  Get the number of keys in a section of a dictionary.
- const char ∗∗ iniparser_getseckeys (const dictionary ∗d, const char ∗s, const char ∗∗keys)
  Get the number of keys in a section of a dictionary.
- const char ∗ iniparser_getstring (const dictionary ∗d, const char ∗key, const char ∗def)
  Get the string associated to a key.
- long int iniparser_getlongint (const dictionary ∗d, const char ∗key, long int notfound)
  Get the string associated to a key, convert to an long int.
- int iniparser_getint (const dictionary ∗d, const char ∗key, int notfound)
  Get the string associated to a key, convert to an int.
- double iniparser_getdouble (const dictionary ∗d, const char ∗key, double notfound)
  Get the string associated to a key, convert to a double.
- long double iniparser_getlongdouble (const dictionary ∗d, const char ∗key, long double notfound)
  Get the string associated to a key, convert to a long double.
- int iniparser_getboolean (const dictionary ∗d, const char ∗key, int notfound)
  Get the string associated to a key, convert to a boolean.
- int iniparser_find_entry (const dictionary ∗ini, const char ∗entry)
  Finds out if a given entry exists in a dictionary.
- int iniparser_set (dictionary ∗ini, const char ∗entry, const char ∗val)
  Set an entry in a dictionary.
- void iniparser_unset (dictionary ∗ini, const char ∗entry)
  Delete an entry in a dictionary.
- dictionary ∗ iniparser_load (const char ∗ininame)
  Parse an ini file and return an allocated dictionary object.
- void iniparser_freedict (dictionary ∗d)
  Free all memory associated to an ini dictionary.

5.3.1 Detailed Description

Parser for ini files.

Author

N. Devillard

5.3.2 Enumeration Type Documentation

5.3.2.1 line_status

```
enum line_status
```

This enum stores the status for each parsed line (internal use only).

5.3.3 Function Documentation

5.3.3.1 iniparser_dump()

```
void iniparser_dump (
            const dictionary * d,
            FILE * f )
```

Dump a dictionary to an opened file pointer.

Parameters

d	Dictionary to dump.
f	Opened file pointer to dump to.

Returns

void

This function prints out the contents of a dictionary, one element by line, onto the provided file pointer. It is OK to specify `stderr` or `stdout` as output files. This function is meant for debugging purposes mostly.

5.3.3.2 iniparser_dump_ini()

```
void iniparser_dump_ini (
            const dictionary * d,
            FILE * f )
```

Save a dictionary to a loadable ini file.

Parameters

d	Dictionary to dump
f	Opened file pointer to dump to

Returns

void

This function dumps a given dictionary into a loadable ini file. It is Ok to specify `stderr` or `stdout` as output files.

5.3.3.3 iniparser_dumpsection_ini()

```
void iniparser_dumpsection_ini (
            const dictionary * d,
            const char * s,
            FILE * f )
```

Save a dictionary section to a loadable ini file.

Parameters

d	Dictionary to dump
s	Section name of dictionary to dump
f	Opened file pointer to dump to

Returns

void

This function dumps a given section of a given dictionary into a loadable ini file. It is Ok to specify `stderr` or `stdout` as output files.

5.3.3.4 iniparser_find_entry()

```
int iniparser_find_entry (
            const dictionary * ini,
            const char * entry )
```

Finds out if a given entry exists in a dictionary.

Parameters

ini	Dictionary to search
entry	Name of the entry to look for

Returns

integer 1 if entry exists, 0 otherwise

Finds out if a given entry exists in the dictionary. Since sections are stored as keys with NULL associated values, this is the only way of querying for the presence of sections in a dictionary.

5.3.3.5 iniparser_freedict()

```
void iniparser_freedict (
            dictionary * d )
```

Free all memory associated to an ini dictionary.

Parameters

d	Dictionary to free

Returns

void

Free all memory associated to an ini dictionary. It is mandatory to call this function before the dictionary object gets out of the current context.

5.3.3.6 iniparser_getboolean()

```
int iniparser_getboolean (
            const dictionary * d,
            const char * key,
            int notfound )
```

Get the string associated to a key, convert to a boolean.

Parameters

d	Dictionary to search
key	Key string to look for
notfound	Value to return in case of error

Returns

integer

This function queries a dictionary for a key. A key as read from an ini file is given as "section:key". If the key cannot be found, the notfound value is returned.

A true boolean is found if one of the following is matched:

- A string starting with 'y'

- A string starting with 'Y'
- A string starting with 't'
- A string starting with 'T'
- A string starting with '1'

A false boolean is found if one of the following is matched:

- A string starting with 'n'
- A string starting with 'N'
- A string starting with 'f'
- A string starting with 'F'
- A string starting with '0'

The notfound value returned if no boolean is identified, does not necessarily have to be 0 or 1.

5.3.3.7 iniparser_getdouble()

```
double iniparser_getdouble (
            const dictionary * d,
            const char * key,
            double notfound )
```

Get the string associated to a key, convert to a double.

Parameters

d	Dictionary to search
key	Key string to look for
notfound	Value to return in case of error

Returns

double

This function queries a dictionary for a key. A key as read from an ini file is given as "section:key". If the key cannot be found, the notfound value is returned.

5.3.3.8 iniparser_getint()

```
int iniparser_getint (
            const dictionary * d,
            const char * key,
            int notfound )
```

Get the string associated to a key, convert to an int.

Parameters

d	Dictionary to search
key	Key string to look for
notfound	Value to return in case of error

Returns

integer

This function queries a dictionary for a key. A key as read from an ini file is given as "section:key". If the key cannot be found, the notfound value is returned.

Supported values for integers include the usual C notation so decimal, octal (starting with 0) and hexadecimal (starting with 0x) are supported. Examples:

"42" -> 42 "042" -> 34 (octal -> decimal) "0x42" -> 66 (hexa -> decimal)

Warning: the conversion may overflow in various ways. Conversion is totally outsourced to strtol(), see the associated man page for overflow handling.

Credits: Thanks to A. Becker for suggesting strtol()

5.3.3.9 iniparser_getlongdouble()

```
long double iniparser_getlongdouble (
             const dictionary * d,
             const char * key,
             long double notfound )
```

Get the string associated to a key, convert to a long double.

Get the string associated to a key, convert to a double.

Parameters

d	Dictionary to search
key	Key string to look for
notfound	Value to return in case of error

Returns

long double

This function queries a dictionary for a key. A key as read from an ini file is given as "section:key". If the key cannot be found, the notfound value is returned.

5.3.3.10 iniparser_getlongint()

```
long int iniparser_getlongint (
             const dictionary * d,
```

```
            const char * key,
            long int notfound )
```

Get the string associated to a key, convert to an long int.

Parameters

d	Dictionary to search
key	Key string to look for
notfound	Value to return in case of error

Returns

long integer

This function queries a dictionary for a key. A key as read from an ini file is given as "section:key". If the key cannot be found, the notfound value is returned.

Supported values for integers include the usual C notation so decimal, octal (starting with 0) and hexadecimal (starting with 0x) are supported. Examples:

"42" -> 42 "042" -> 34 (octal -> decimal) "0x42" -> 66 (hexa -> decimal)

Warning: the conversion may overflow in various ways. Conversion is totally outsourced to strtol(), see the associated man page for overflow handling.

Credits: Thanks to A. Becker for suggesting strtol()

5.3.3.11 iniparser_getnsec()

```
int iniparser_getnsec (
            const dictionary * d )
```

Get number of sections in a dictionary.

Parameters

d	Dictionary to examine

Returns

int Number of sections found in dictionary

This function returns the number of sections found in a dictionary. The test to recognize sections is done on the string stored in the dictionary: a section name is given as "section" whereas a key is stored as "section:key", thus the test looks for entries that do not contain a colon.

This clearly fails in the case a section name contains a colon, but this should simply be avoided.

This function returns -1 in case of error.

5.3.3.12 iniparser_getseckeys()

```
const char** iniparser_getseckeys (
             const dictionary * d,
             const char * s,
             const char ** keys )
```

Get the number of keys in a section of a dictionary.

Parameters

d	Dictionary to examine
s	Section name of dictionary to examine
keys	Already allocated array to store the keys in

Returns

The pointer passed as `keys` argument or NULL in case of error

This function queries a dictionary and finds all keys in a given section. The keys argument should be an array of pointers which size has been determined by calling `iniparser_getsecnkeys` function prior to this one.

Each pointer in the returned char pointer-to-pointer is pointing to a string allocated in the dictionary; do not free or modify them.

5.3.3.13 iniparser_getsecname()

```
const char* iniparser_getsecname (
             const dictionary * d,
             int n )
```

Get name for section n in a dictionary.

Parameters

d	Dictionary to examine
n	Section number (from 0 to nsec-1).

Returns

Pointer to char string

This function locates the n-th section in a dictionary and returns its name as a pointer to a string statically allocated inside the dictionary. Do not free or modify the returned string!

This function returns NULL in case of error.

5.3.3.14 iniparser_getsecnkeys()

```
int iniparser_getsecnkeys (
             const dictionary * d,
             const char * s )
```

Get the number of keys in a section of a dictionary.

Parameters

d	Dictionary to examine
s	Section name of dictionary to examine

Returns

Number of keys in section

5.3.3.15 iniparser_getstring()

```
const char* iniparser_getstring (
            const dictionary * d,
            const char * key,
            const char * def )
```

Get the string associated to a key.

Parameters

d	Dictionary to search
key	Key string to look for
def	Default value to return if key not found.

Returns

pointer to statically allocated character string

This function queries a dictionary for a key. A key as read from an ini file is given as "section:key". If the key cannot be found, the pointer passed as 'def' is returned. The returned char pointer is pointing to a string allocated in the dictionary, do not free or modify it.

5.3.3.16 iniparser_load()

```
dictionary* iniparser_load (
            const char * ininame )
```

Parse an ini file and return an allocated dictionary object.

Parameters

ininame	Name of the ini file to read.

Returns

Pointer to newly allocated dictionary

This is the parser for ini files. This function is called, providing the name of the file to be read. It returns a dictionary object that should not be accessed directly, but through accessor functions instead.

The returned dictionary must be freed using iniparser_freedict().

5.3.3.17 iniparser_set()

```
int iniparser_set (
            dictionary * ini,
            const char * entry,
            const char * val )
```

Set an entry in a dictionary.

Parameters

ini	Dictionary to modify.
entry	Entry to modify (entry name)
val	New value to associate to the entry.

Returns

int 0 if Ok, -1 otherwise.

If the given entry can be found in the dictionary, it is modified to contain the provided value. If it cannot be found, the entry is created. It is Ok to set val to NULL.

5.3.3.18 iniparser_unset()

```
void iniparser_unset (
            dictionary * ini,
            const char * entry )
```

Delete an entry in a dictionary.

Parameters

ini	Dictionary to modify
entry	Entry to delete (entry name)

Returns

void

If the given entry can be found, it is deleted from the dictionary.

5.4 iniparser.h File Reference

Parser for ini files.

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

Functions

- void iniparser_set_error_callback (int(*errback)(const char *,...))
  Configure a function to receive the error messages.
- int iniparser_getnsec (const dictionary *d)
  Get number of sections in a dictionary.
- const char * iniparser_getsecname (const dictionary *d, int n)
  Get name for section n in a dictionary.
- void iniparser_dump_ini (const dictionary *d, FILE *f)
  Save a dictionary to a loadable ini file.
- void iniparser_dumpsection_ini (const dictionary *d, const char *s, FILE *f)
  Save a dictionary section to a loadable ini file.
- void iniparser_dump (const dictionary *d, FILE *f)
  Dump a dictionary to an opened file pointer.
- int iniparser_getsecnkeys (const dictionary *d, const char *s)
  Get the number of keys in a section of a dictionary.
- const char ** iniparser_getseckeys (const dictionary *d, const char *s, const char **keys)
  Get the number of keys in a section of a dictionary.
- const char * iniparser_getstring (const dictionary *d, const char *key, const char *def)
  Get the string associated to a key.
- int iniparser_getint (const dictionary *d, const char *key, int notfound)
  Get the string associated to a key, convert to an int.
- long int iniparser_getlongint (const dictionary *d, const char *key, long int notfound)
  Get the string associated to a key, convert to an long int.
- double iniparser_getdouble (const dictionary *d, const char *key, double notfound)
  Get the string associated to a key, convert to a double.
- long double iniparser_getlongdouble (const dictionary *d, const char *key, long double notfound)
  Get the string associated to a key, convert to a double.
- int iniparser_getboolean (const dictionary *d, const char *key, int notfound)
  Get the string associated to a key, convert to a boolean.
- int iniparser_set (dictionary *ini, const char *entry, const char *val)
  Set an entry in a dictionary.
- void iniparser_unset (dictionary *ini, const char *entry)
  Delete an entry in a dictionary.
- int iniparser_find_entry (const dictionary *ini, const char *entry)
  Finds out if a given entry exists in a dictionary.
- dictionary * iniparser_load (const char *ininame)
  Parse an ini file and return an allocated dictionary object.
- void iniparser_freedict (dictionary *d)
  Free all memory associated to an ini dictionary.

5.4.1 Detailed Description

Parser for ini files.

Author

N. Devillard

5.4.2 Function Documentation

5.4.2.1 iniparser_dump()

```
void iniparser_dump (
            const dictionary * d,
            FILE * f )
```

Dump a dictionary to an opened file pointer.

Parameters

d	Dictionary to dump.
f	Opened file pointer to dump to.

Returns

void

This function prints out the contents of a dictionary, one element by line, onto the provided file pointer. It is OK to specify `stderr` or `stdout` as output files. This function is meant for debugging purposes mostly.

5.4.2.2 iniparser_dump_ini()

```
void iniparser_dump_ini (
            const dictionary * d,
            FILE * f )
```

Save a dictionary to a loadable ini file.

Parameters

d	Dictionary to dump
f	Opened file pointer to dump to

Returns

void

This function dumps a given dictionary into a loadable ini file. It is Ok to specify `stderr` or `stdout` as output files.

5.4.2.3 iniparser_dumpsection_ini()

```
void iniparser_dumpsection_ini (
            const dictionary * d,
            const char * s,
            FILE * f )
```

Save a dictionary section to a loadable ini file.

Parameters

d	Dictionary to dump
s	Section name of dictionary to dump
f	Opened file pointer to dump to

Returns

void

This function dumps a given section of a given dictionary into a loadable ini file. It is Ok to specify `stderr` or `stdout` as output files.

5.4.2.4 iniparser_find_entry()

```
int iniparser_find_entry (
            const dictionary * ini,
            const char * entry )
```

Finds out if a given entry exists in a dictionary.

Parameters

ini	Dictionary to search
entry	Name of the entry to look for

Returns

integer 1 if entry exists, 0 otherwise

Finds out if a given entry exists in the dictionary. Since sections are stored as keys with NULL associated values, this is the only way of querying for the presence of sections in a dictionary.

5.4.2.5 iniparser_freedict()

```
void iniparser_freedict (
            dictionary * d )
```

Free all memory associated to an ini dictionary.

Parameters

d	Dictionary to free

Returns

void

Free all memory associated to an ini dictionary. It is mandatory to call this function before the dictionary object gets out of the current context.

5.4.2.6 iniparser_getboolean()

```
int iniparser_getboolean (
            const dictionary * d,
            const char * key,
            int notfound )
```

Get the string associated to a key, convert to a boolean.

Parameters

d	Dictionary to search
key	Key string to look for
notfound	Value to return in case of error

Returns

integer

This function queries a dictionary for a key. A key as read from an ini file is given as "section:key". If the key cannot be found, the notfound value is returned.

A true boolean is found if one of the following is matched:

- A string starting with 'y'
- A string starting with 'Y'
- A string starting with 't'
- A string starting with 'T'
- A string starting with '1'

A false boolean is found if one of the following is matched:

- A string starting with 'n'
- A string starting with 'N'
- A string starting with 'f'
- A string starting with 'F'
- A string starting with '0'

The notfound value returned if no boolean is identified, does not necessarily have to be 0 or 1.

5.4.2.7 iniparser_getdouble()

```
double iniparser_getdouble (
            const dictionary * d,
            const char * key,
            double notfound )
```

Get the string associated to a key, convert to a double.

Parameters

d	Dictionary to search
key	Key string to look for
notfound	Value to return in case of error

Returns

double

This function queries a dictionary for a key. A key as read from an ini file is given as "section:key". If the key cannot be found, the notfound value is returned.

5.4.2.8 iniparser_getint()

```
int iniparser_getint (
            const dictionary * d,
            const char * key,
            int notfound )
```

Get the string associated to a key, convert to an int.

Parameters

d	Dictionary to search
key	Key string to look for
notfound	Value to return in case of error

Returns

integer

This function queries a dictionary for a key. A key as read from an ini file is given as "section:key". If the key cannot be found, the notfound value is returned.

Supported values for integers include the usual C notation so decimal, octal (starting with 0) and hexadecimal (starting with 0x) are supported. Examples:

- "42" -> 42
- "042" -> 34 (octal -> decimal)
- "0x42" -> 66 (hexa -> decimal)

Warning: the conversion may overflow in various ways. Conversion is totally outsourced to strtol(), see the associated man page for overflow handling.

Credits: Thanks to A. Becker for suggesting strtol()

Parameters

d	Dictionary to search
key	Key string to look for
notfound	Value to return in case of error

Returns

integer

This function queries a dictionary for a key. A key as read from an ini file is given as "section:key". If the key cannot be found, the notfound value is returned.

Supported values for integers include the usual C notation so decimal, octal (starting with 0) and hexadecimal (starting with 0x) are supported. Examples:

"42" -> 42 "042" -> 34 (octal -> decimal) "0x42" -> 66 (hexa -> decimal)

Warning: the conversion may overflow in various ways. Conversion is totally outsourced to strtol(), see the associated man page for overflow handling.

Credits: Thanks to A. Becker for suggesting strtol()

5.4.2.9 iniparser_getlongdouble()

```
long double iniparser_getlongdouble (
            const dictionary * d,
            const char * key,
            long double notfound )
```

Get the string associated to a key, convert to a double.

Parameters

d	Dictionary to search
key	Key string to look for
notfound	Value to return in case of error

Returns

long double

This function queries a dictionary for a key. A key as read from an ini file is given as "section:key". If the key cannot be found, the notfound value is returned.

Get the string associated to a key, convert to a double.

Parameters

d	Dictionary to search
key	Key string to look for
notfound	Value to return in case of error

Returns

long double

This function queries a dictionary for a key. A key as read from an ini file is given as "section:key". If the key cannot be found, the notfound value is returned.

5.4.2.10 iniparser_getlongint()

```
long int iniparser_getlongint (
            const dictionary * d,
            const char * key,
            long int notfound )
```

Get the string associated to a key, convert to an long int.

Parameters

d	Dictionary to search
key	Key string to look for
notfound	Value to return in case of error

Returns

integer

This function queries a dictionary for a key. A key as read from an ini file is given as "section:key". If the key cannot be found, the notfound value is returned.

Supported values for integers include the usual C notation so decimal, octal (starting with 0) and hexadecimal (starting with 0x) are supported. Examples:

- "42" -> 42
- "042" -> 34 (octal -> decimal)
- "0x42" -> 66 (hexa -> decimal)

Warning: the conversion may overflow in various ways. Conversion is totally outsourced to strtol(), see the associated man page for overflow handling.

Parameters

d	Dictionary to search
key	Key string to look for
notfound	Value to return in case of error

Returns

long integer

This function queries a dictionary for a key. A key as read from an ini file is given as "section:key". If the key cannot be found, the notfound value is returned.

Supported values for integers include the usual C notation so decimal, octal (starting with 0) and hexadecimal (starting with 0x) are supported. Examples:

"42" -> 42 "042" -> 34 (octal -> decimal) "0x42" -> 66 (hexa -> decimal)

Warning: the conversion may overflow in various ways. Conversion is totally outsourced to strtol(), see the associated man page for overflow handling.

Credits: Thanks to A. Becker for suggesting strtol()

5.4.2.11 iniparser_getnsec()

```
int iniparser_getnsec (
            const dictionary * d )
```

Get number of sections in a dictionary.

Parameters

d	Dictionary to examine

Returns

int Number of sections found in dictionary

This function returns the number of sections found in a dictionary. The test to recognize sections is done on the

string stored in the dictionary: a section name is given as "section" whereas a key is stored as "section:key", thus the test looks for entries that do not contain a colon.

This clearly fails in the case a section name contains a colon, but this should simply be avoided.

This function returns -1 in case of error.

5.4.2.12 iniparser_getseckeys()

```
const char** iniparser_getseckeys (
             const dictionary * d,
             const char * s,
             const char ** keys )
```

Get the number of keys in a section of a dictionary.

Parameters

d	Dictionary to examine
s	Section name of dictionary to examine
keys	Already allocated array to store the keys in

Returns

The pointer passed as `keys` argument or NULL in case of error

This function queries a dictionary and finds all keys in a given section. The keys argument should be an array of pointers which size has been determined by calling `iniparser_getsecnkeys` function prior to this one.

Each pointer in the returned char pointer-to-pointer is pointing to a string allocated in the dictionary; do not free or modify them.

5.4.2.13 iniparser_getsecname()

```
const char* iniparser_getsecname (
             const dictionary * d,
             int n )
```

Get name for section n in a dictionary.

Parameters

d	Dictionary to examine
n	Section number (from 0 to nsec-1).

Returns

Pointer to char string

This function locates the n-th section in a dictionary and returns its name as a pointer to a string statically allocated inside the dictionary. Do not free or modify the returned string!

This function returns NULL in case of error.

5.4.2.14 iniparser_getsecnkeys()

```
int iniparser_getsecnkeys (
            const dictionary * d,
            const char * s )
```

Get the number of keys in a section of a dictionary.

Parameters

d	Dictionary to examine
s	Section name of dictionary to examine

Returns

Number of keys in section

5.4.2.15 iniparser_getstring()

```
const char* iniparser_getstring (
            const dictionary * d,
            const char * key,
            const char * def )
```

Get the string associated to a key.

Parameters

d	Dictionary to search
key	Key string to look for
def	Default value to return if key not found.

Returns

pointer to statically allocated character string

This function queries a dictionary for a key. A key as read from an ini file is given as "section:key". If the key cannot be found, the pointer passed as 'def' is returned. The returned char pointer is pointing to a string allocated in the dictionary, do not free or modify it.

5.4.2.16 iniparser_load()

```
dictionary* iniparser_load (
            const char * ininame )
```

Parse an ini file and return an allocated dictionary object.

Parameters

ininame	Name of the ini file to read.

Returns

Pointer to newly allocated dictionary

This is the parser for ini files. This function is called, providing the name of the file to be read. It returns a dictionary object that should not be accessed directly, but through accessor functions instead.

The returned dictionary must be freed using iniparser_freedict().

5.4.2.17 iniparser_set()

```
int iniparser_set (
            dictionary * ini,
            const char * entry,
            const char * val )
```

Set an entry in a dictionary.

Parameters

ini	Dictionary to modify.
entry	Entry to modify (entry name)
val	New value to associate to the entry.

Returns

int 0 if Ok, -1 otherwise.

If the given entry can be found in the dictionary, it is modified to contain the provided value. If it cannot be found, the entry is created. It is Ok to set val to NULL.

5.4.2.18 iniparser_set_error_callback()

```
void iniparser_set_error_callback (
            int(*)(const char *,...)  errback )
```

Configure a function to receive the error messages.

Parameters

errback	Function to call.

By default, the error will be printed on stderr. If a null pointer is passed as errback the error callback will be switched back to default.

5.4.2.19 iniparser_unset()

```
void iniparser_unset (
            dictionary * ini,
            const char * entry )
```

Delete an entry in a dictionary.

Parameters

ini	Dictionary to modify
entry	Entry to delete (entry name)

Returns

void

If the given entry can be found, it is deleted from the dictionary.

Index

aa
  Parameters, 16
Array, 7
  array, 7
  size, 7
  used, 7
array
  Array, 7
AxesDBL, 8
  ctxfactor, 8
  ctyfactor, 8
  xmax, 8
  xmin, 8
  ymax, 8
  ymin, 8
AxesFLT128, 10
  ctxfactor, 10
  ctyfactor, 10
  xmax, 10
  xmin, 10
  ymax, 11
  ymin, 11
AxesFLT, 9
  ctxfactor, 9
  ctyfactor, 9
  xmax, 9
  xmin, 9
  ymax, 9
  ymin, 10
AxesLDBL, 11
  ctxfactor, 11
  ctyfactor, 11
  xmax, 11
  xmin, 12
  ymax, 12
  ymin, 12

b
  Rgb, 20
buffer
  printbuffer, 19

cJSON_Hooks, 14
cJSON, 12
  child, 12
  next, 13
  prev, 13
  string, 13
  type, 13
  valuedouble, 13
  valueint, 13
  valuestring, 13
centerX
  Parameters, 16
centerY
  Parameters, 16
child
  cJSON, 12
color
  Parameters, 16
config
  Parameters, 16
content
  parse_buffer, 18
ctxfactor
  AxesDBL, 8
  AxesFLT128, 10
  AxesFLT, 9
  AxesLDBL, 11
ctyfactor
  AxesDBL, 8
  AxesFLT128, 10
  AxesFLT, 9
  AxesLDBL, 11
cv
  Parameters, 17

DICT_INVALID_KEY
  dictionary.c, 24
DICTMINSZ
  dictionary.c, 24
depth
  parse_buffer, 18
  printbuffer, 19
diameter
  Parameters, 17
dictionary, 28
dictionary.c, 23
  DICT_INVALID_KEY, 24
  DICTMINSZ, 24
  dictionary_del, 24
  dictionary_dump, 25
  dictionary_get, 25
  dictionary_hash, 25
  dictionary_new, 26
  dictionary_set, 26
  dictionary_unset, 27
  MAXVALSZ, 24
dictionary.h, 27
  dictionary_del, 28

dictionary_dump, 29
dictionary_get, 29
dictionary_hash, 30
dictionary_new, 30
dictionary_set, 30
dictionary_unset, 31
dictionary_del
dictionary.c, 24
dictionary.h, 28
dictionary_dump
dictionary.c, 25
dictionary.h, 29
dictionary_get
dictionary.c, 25
dictionary.h, 29
dictionary_hash
dictionary.c, 25
dictionary.h, 30
dictionary_new
dictionary.c, 26
dictionary.h, 30
dictionary_set
dictionary.c, 26
dictionary.h, 30
dictionary_unset
dictionary.c, 27
dictionary.h, 31

error, 14
json, 14
position, 14

filename
Parameters, 17
format
printbuffer, 19

g
Rgb, 21

H
HSV, 15
HSV, 15
H, 15
S, 15
V, 15
height
Parameters, 17
hooks
parse_buffer, 18
printbuffer, 20

iniparser.c, 31
iniparser_dump, 33
iniparser_dump_ini, 33
iniparser_dumpsection_ini, 35
iniparser_find_entry, 35
iniparser_freedict, 36
iniparser_getboolean, 36
iniparser_getdouble, 37
iniparser_getint, 37
iniparser_getlongdouble, 38
iniparser_getlongint, 38
iniparser_getnsec, 39
iniparser_getseckeys, 39
iniparser_getsecname, 40
iniparser_getsecnkeys, 40
iniparser_getstring, 41
iniparser_load, 41
iniparser_set, 42
iniparser_unset, 42
line_status, 33
iniparser.h, 43
iniparser_dump, 44
iniparser_dump_ini, 44
iniparser_dumpsection_ini, 45
iniparser_find_entry, 45
iniparser_freedict, 45
iniparser_getboolean, 46
iniparser_getdouble, 47
iniparser_getint, 47
iniparser_getlongdouble, 48
iniparser_getlongint, 49
iniparser_getnsec, 50
iniparser_getseckeys, 51
iniparser_getsecname, 51
iniparser_getsecnkeys, 52
iniparser_getstring, 52
iniparser_load, 52
iniparser_set, 53
iniparser_set_error_callback, 53
iniparser_unset, 53
iniparser_dump
iniparser.c, 33
iniparser.h, 44
iniparser_dump_ini
iniparser.c, 33
iniparser.h, 44
iniparser_dumpsection_ini
iniparser.c, 35
iniparser.h, 45
iniparser_find_entry
iniparser.c, 35
iniparser.h, 45
iniparser_freedict
iniparser.c, 36
iniparser.h, 45
iniparser_getboolean
iniparser.c, 36
iniparser.h, 46
iniparser_getdouble
iniparser.c, 37
iniparser.h, 47
iniparser_getint
iniparser.c, 37
iniparser.h, 47
iniparser_getlongdouble

iniparser.c, 38
iniparser.h, 48
iniparser_getlongint
iniparser.c, 38
iniparser.h, 49
iniparser_getnsec
iniparser.c, 39
iniparser.h, 50
iniparser_getseckeys
iniparser.c, 39
iniparser.h, 51
iniparser_getsecname
iniparser.c, 40
iniparser.h, 51
iniparser_getsecnkeys
iniparser.c, 40
iniparser.h, 52
iniparser_getstring
iniparser.c, 41
iniparser.h, 52
iniparser_load
iniparser.c, 41
iniparser.h, 52
iniparser_set
iniparser.c, 42
iniparser.h, 53
iniparser_set_error_callback
iniparser.h, 53
iniparser_unset
iniparser.c, 42
iniparser.h, 53
internal_hooks, 15

json
error, 14

length
parse_buffer, 19
printbuffer, 20
line_status
iniparser.c, 33

MAXVALSZ
dictionary.c, 24
magnify
Parameters, 17
maxiter
Parameters, 17

next
cJSON, 13
Parameters, 17
noalloc
printbuffer, 20

offset
parse_buffer, 19
printbuffer, 20

palname
Parameters, 17
Parameters, 16
aa, 16
centerX, 16
centerY, 16
color, 16
config, 16
cv, 17
diameter, 17
filename, 17
height, 17
magnify, 17
maxiter, 17
next, 17
palname, 17
tweak, 18
width, 18
parse_buffer, 18
content, 18
depth, 18
hooks, 18
length, 19
offset, 19
position
error, 14
prev
cJSON, 13
printbuffer, 19
buffer, 19
depth, 19
format, 19
hooks, 20
length, 20
noalloc, 20
offset, 20

r
Rgb, 21
Rgb, 20
b, 20
g, 21
r, 21

S
HSV, 15
size
Array, 7
string
cJSON, 13

tweak
Parameters, 18
type
cJSON, 13

used
Array, 7

V

HSV, 15
valuedouble
cJSON, 13
valueint
cJSON, 13
valuestring
cJSON, 13

width
Parameters, 18

xmax
AxesDBL, 8
AxesFLT128, 10
AxesFLT, 9
AxesLDBL, 11
xmin
AxesDBL, 8
AxesFLT128, 10
AxesFLT, 9
AxesLDBL, 12

ymax
AxesDBL, 8
AxesFLT128, 11
AxesFLT, 9
AxesLDBL, 12
ymin
AxesDBL, 8
AxesFLT128, 11
AxesFLT, 10
AxesLDBL, 12