CFLIB - Flexible Configuration Library

Patchlevel 21

Generated by Doxygen 1.8.0

Wed Feb 27 2013 14:11:12

Contents

1 CFLIB Documentation

1.1 Introduction and General Issues

CFLIB is a small, simple, flexible and portable ANSI C Library to be used as configuration interface for user programs. CFLIB builds and maintains a compact database structure consisting of a list of parameters with their name, content and some additional information about each parameter.

CFLIB targets the basic needs of technical, scientific or other programmers who want to spend minimal time on coding input, output, variable parsing, report generation and the like but still have a simple to use, reliable, flexible and portable configuration interface for their programs.

Main Features:

- Commandline, environment and terminal input parsing
  - Configuration files
  - File search
  - Template driven report generation
  - Automatic time and date update

See also

- Purpose of CFLIB
- CFLIB License
- The names "CFLIB", "libcf", "∗cf∗"
- General Notes

Project Homepage:

- http://cflib.berlios.de

1.2 Basic Usage

1. Include cf.h

1. Define the Configuration Initializer, an array of CONFIG structures

1. Call cfinit() with Configuration Initializer and Commandline. The most compact initialization is done by cfstart(), a wrapper function for cfinit() that includes error reporting, usage message and (optional) debugging output. On initialization the following data sources are inspected or parsed in the order presented:

- Commandline Arguments (or compatible structure given to cfinit()) according to the description in Commandline Option for Parameter
  - Environment Variables
  - Configuration Files
  - Built-in User-defined Default from Configuration Initializer: Parameter Default Value
  - Get parameter value from Standard Input `stdin`, if required

1. Use the Retrieval Functions **cfget**∗() to access configuration parameters

1. Compile your program, linking the appropriate CFLIB library file for your platform and setup, usually the file name is *libcf.a* which means the library is referred to as **"cf"**. You can change the names to fit into your setup: See The names "CFLIB", "libcf", "∗cf∗"

1. Run your program, test CFLIB functionality and adjust the Configuration Initializer according to your needs

See also

Simple Usage Example
Configuration Parsing Levels and Source/Origin Options
Core Features

1.3 Retrieval Functions

1. Get CFLIB Version and Copyright Information: cfgetvers(), cfgetsubvers(), cfgetcpr()

1. Get Usage Message for Output: cfgetusg()

1. Get Configuration Parameter Value:

- Get parameter value: cfget(), interpretation of content and return type depend on the type setting in the parameter's Special Options Mask
  - Get string value: cfgetstr()
  - Get integer value: cfgetnum()
  - Get real/float value: cfgetreal()
  - Inquire flag/switch status: cfgetflag()
  - Get value of (next) residual command line argument: cfgetres()

Note

All of these functions except cfgetres() require the parameter name as argument

1.4 Setting Parameters and Saving the Configuration

- cfput() : Update or Add a Parameter.
- cfputstr() : cfputstr() Update or Add Parameter *name* with string *content*
- cfputtime() : cfputtime() Set all Time and/or Date entries in CFLIB DB to *now* or *today*
- cfnosave() : cfnosave() Alter or query the CF_NOSAVE Flag of Parameter *name*
- cfsave() : cfsave() Write configuration data to a Configuration File or `stdout`
- See Configuration Files for details

1.5 General Utilities

These functions and Macros are used in the library but do not depend on the configuration database or any cf∗() functions. They are (small) general tools that you can use in your program if you like.

- String Manipulation: EatWhiteSpace(), RemoveCR(), RemoveTrailSpace()
  - File Utilities: FindFile(), BackupFile()
  - Other Tools: IsATerminal(), DelFlag(), SetFlag()

1.6 Advanced Usage

1.6.1 Error Handling

CFLIB maintains a simple global Error Stack that is used by library functions like cfinit() when multiple errors can occur. Error Items consist of a numeric Error Code and (optionally) an Error Message string. Repeated calls to cfgeterr() will successively return error entries while deleting them from the stack until the list is empty. User programs may also use the CFLIB error stack by calling cfputerr() without the need to initialize a configuration.

For more details see: Error Handling - Error Codes, Functions and Structures.

1.6.2 Report Generation

- Generate Output from Template and current parameter values: cfform()

1.6.3 Configuration Debugging

These functions are thought to be used by the programmer working with CFLIB during development and testing of a program.

- Dump Configuration DB in human readable form: cfdump()
  - Test and Dump Configuration Initializer: cfdinichk()

1.6.4 Get Information About a Configuration Parameter:

- Get source/origin of the parameter's value: cfgetsrc()
- Inquire Bit from parameter's Special Options Mask by Offset: cfflaginq()

1.6.5 Other Functions and Features

- Exit Configuration: cfexit()
- Expand User Home Directory in a File Path Parameter: cfhomexp()
- General (internal) retrieval function: cfgetent()

1.7 Compilation and Development Issues

- Platforms
- Binaries and Executables
- Building the Library
- System and Compiler dependent Defines
- Minimal CFLIB Replacement

1.8 Help and Support

- **Help** is this documentation
- **Support** is the (open) source code
- The project is maintained from time to time as needed ;-)
- Comments, Bug Reports or (better) Bug Fixes are welcome!
- See CFLIB License
- ... Have Fun!

2 CFLIB License

This file is part of **CFLIB** - Flexible Configuration Library.

Author

Stefan Habermehl stefan.habermehl@mcff.de

```
  2. Conveying Modified Versions.

  If you modify a copy of the Library, and, in your modifications, a
facility refers to a function or data to be supplied by an Application
that uses the facility (other than as an argument passed when the
facility is invoked), then you may convey a copy of the modified
version:

   a) under this License, provided that you make a good faith effort to
   ensure that, in the event an Application does not supply the
   function or data, the facility still operates, and performs
   whatever part of its purpose remains meaningful, or

   b) under the GNU GPL, with none of the additional permissions of
   this License applicable to that copy.

  3. Object Code Incorporating Material from Library Header Files.

  The object code form of an Application may incorporate material from
a header file that is part of the Library.  You may convey such object
code under terms of your choice, provided that, if the incorporated
material is not limited to numerical parameters, data structure
layouts and accessors, or small macros, inline functions and templates
(ten or fewer lines in length), you do both of the following:

   a) Give prominent notice with each copy of the object code that the
   Library is used in it and that the Library and its use are
   covered by this License.

   b) Accompany the object code with a copy of the GNU GPL and this license
   document.

  4. Combined Works.

  You may convey a Combined Work under terms of your choice that,
taken together, effectively do not restrict modification of the
portions of the Library contained in the Combined Work and reverse
engineering for debugging such modifications, if you also do each of
the following:

   a) Give prominent notice with each copy of the Combined Work that
   the Library is used in it and that the Library and its use are
   covered by this License.

   b) Accompany the Combined Work with a copy of the GNU GPL and this license
   document.

   c) For a Combined Work that displays copyright notices during
   execution, include the copyright notice for the Library among
   these notices, as well as a reference directing the user to the
   copies of the GNU GPL and this license document.

   d) Do one of the following:

       0) Convey the Minimal Corresponding Source under the terms of this
       License, and the Corresponding Application Code in a form
       suitable for, and under terms that permit, the user to
       recombine or relink the Application with a modified version of
       the Linked Version to produce a modified Combined Work, in the
       manner specified by section 6 of the GNU GPL for conveying
       Corresponding Source.

       1) Use a suitable shared library mechanism for linking with the
       Library.  A suitable mechanism is one that (a) uses at run time
       a copy of the Library already present on the user's computer
       system, and (b) will operate properly with a modified version
       of the Library that is interface-compatible with the Linked
       Version.

   e) Provide Installation Information, but only if you would otherwise
   be required to provide such information under section 6 of the
   GNU GPL, and only to the extent that such information is
   necessary to install and execute a modified version of the
```

```
   Combined Work produced by recombining or relinking the
   Application with a modified version of the Linked Version. (If
   you use option 4d0, the Installation Information must accompany
   the Minimal Corresponding Source and Corresponding Application
   Code. If you use option 4d1, you must provide the Installation
   Information in the manner specified by section 6 of the GNU GPL
   for conveying Corresponding Source.)

  5. Combined Libraries.

  You may place library facilities that are a work based on the
Library side by side in a single library together with other library
facilities that are not Applications and are not covered by this
License, and convey such a combined library under terms of your
choice, if you do both of the following:

   a) Accompany the combined library with a copy of the same work based
   on the Library, uncombined with any other library facilities,
   conveyed under the terms of this License.

   b) Give prominent notice with the combined library that part of it
   is a work based on the Library, and explaining where to find the
   accompanying uncombined form of the same work.

  6. Revised Versions of the GNU Lesser General Public License.

  The Free Software Foundation may publish revised and/or new versions
of the GNU Lesser General Public License from time to time. Such new
versions will be similar in spirit to the present version, but may
differ in detail to address new problems or concerns.

  Each version is given a distinguishing version number. If the
Library as you received it specifies that a certain numbered version
of the GNU Lesser General Public License "or any later version"
applies to it, you have the option of following the terms and
conditions either of that published version or of any later version
published by the Free Software Foundation. If the Library as you
received it does not specify a version number of the GNU Lesser
General Public License, you may choose any version of the GNU Lesser
General Public License ever published by the Free Software Foundation.

  If the Library as you received it specifies that a proxy can decide
whether future versions of the GNU Lesser General Public License shall
apply, that proxy's public statement of acceptance of any version is
permanent authorization for you to choose that version for the
Library.
```

3 Purpose of CFLIB

This library started as a Beginner Project in "C Library Building" following practical needs arising from scientific/technical modeling projects.

Most of the library code has been created in 1994/95 on an Atari ST4/16 MHz under TOS 2.05/MiNT 1.12 with gcc 2.5.8 and the Mintlibs Patchlevel 46. The C coding is probably not the best possible ;-) but the source as well as the executable and allocated memory structures are very compact, simple to modify and still fine for many applications that need a stable, portable and small configuration interface.

'Mission Statement' from 1994/95 README file:

> CFLIB is meant to be a flexible, sound and easy to use tool for C programmers. It provides a set of functions for a standard method for feeding a C program with all the (external) information it needs to perform as desired with a minimum of expense for both the programmer and the user: arguments, commands, program input, configurable and/or installation dependent features, system settings and a lot more can be passed to the program through different interfaces: command line, environment, configuration files or sections within them, interactive input and last but not least a built-in default. So it should be a fast and easy task for both the programmer and the user to build and change a configuration. The library will always take more space and perform slower than code that is written and optimized for a specific requirement, but this will only be a noticeable disadvantage in some

cases. The library also provides some special features such as file search, time and date handling, generating simple, text template driven reports, etc. It is suitable for creating a comfortable and/or individual user interface for existing programs that don't have one. Some basic ideas for the library came from looking at the *termcap* library for easy and portable terminal I/O.

4 General Notes

4.1 The names "CFLIB", "libcf", "∗cf∗"

All "names" in this library source and documentation are nothing but technical acronyms just as "src", "inc", "C" and are **not** registered names, trademarks or anything like that ;-) You may change the names of the library, the include file and on compilation the function names and defines to whatever you like :-)

4.2 Input and Output

CFLIB is designed to be usable as a **Filter** program that reads from `stdin` and writes to `stdout` wherever it was desirable.

None of the library functions will produce any terminal output (to `stdout` or `stderr`) unless:

- The CF_QUERY flag is set in the Special Options Mask of an entry and the parameter value is still unassigned after parsing all other possible sources in cfinit().
- You call a function that reads from or writes to files with arguments that trigger the use of `stdin`, `stdout`, `stderr` etc. according to the function's documentation.

4.3 General (future) tasks

Todo Make proper Man Pages with function references etc.

Make Doxygen Developer Documentation

More Modularization! Still very similar code in various functions

See also

Bug List

4.4 Other general issues

- Library Versions

5 Simple Usage Example

```
/*
 *    Find a File in Path
 *
 *    Input:  Commandline or Environment
 *
 *    Output: File path, Help message or Error message
 *
 *    Compile: gcc fifi.c -o fifi -lcf -L../../lib<platform>
 *    Run:     ./fifi -h
 *    Test:    ./fifi gcc
 */
```

```
#include <stdio.h>

#include "../../include/cf.h"

int main( int argc, char **argv )
{
    int ret; char *tmp;

    CONFIG setting[] =
    {
        /* Name, Default Content, Commandline Option, Special Option Flags */
        { "PROGNAME", NULL,        '1', CF_FINDFILE|CF_SET_ARG|CF_NO_OPT_ARG, }
      ,
        { "FF_PATH",  NULL,        ' ', CF_PATH,                              }
      ,
        { "help",     CF_FLAG_OFF, 'h', CF_FLAG|CF_LAST,                      }
      ,
    };

    if ( !( ret = cfstart( setting, argc, argv, "help", CFS_USG ) ) ){

        if( ( tmp = cfgetstr("PROGNAME") ) != NULL )

            fprintf( stdout, "%s\n", tmp );
    }

    return ret;
}
```

6 Configuration Initializer

The **"Configuration Initializer"** is the major interface between the user program and CFLIB.

It is an Array of CONFIG items, just as the CFLIB Configuration Database itself.

Every parameter is characterized by:

- Parameter Name (CONFIG::name)
- Parameter Default Value (CONFIG::inhalt)
- Commandline Option for Parameter (CONFIG::option)
- Special Option Flags Mask for Parameter (CONFIG::flag)

An example initializer may look like this:

```
 const CONFIG initializer[] = {
       // name         default         option  flag
     { "SPI_PROFILE", "~/.spi.cnf",   'c',    CF_SETFILE|CF_SET_ENV,     },
     { "SPI_SYSCONF", "/etc/spi.cnf",' ',     CF_SYS_SETFILE|CF_NOSAVE,  },
     { "SPI_DEFS",    "loop-0.A",     'f',    CF_SECTION,                },
     { "SPI_ID",      "",             ' ',    CF_SET_PRIV,               },
     { "verbosity",   "1",            'v',    CF_INT|CF_CONCAT|CF_NOSAVE,},
     { "label_items", CF_FLAG_ON,     'l',    CF_FLAG,                   },
     { "label_date",  NULL,           ' ',    CF_DATE|CF_SET_PUT,        },
     { "label_logo",  NULL,           'L',    0,                         },
     { "output_file", NULL,           'o',    CF_CONCAT|CF_QUERY,        },
     { "TERM",        "0.2",          'T',    CF_STR|CF_IGN_ENV,         },
     { "SCALE",       "1",            'S',    CF_REAL|CF_IGN_ENV,        },
     { "LOOPMOD",     "random",       '1',    CF_NO_OPT_ARG,             },
     { "CYCLES",      "2009",         '2',    CF_INT|CF_NO_OPT_ARG,      },
     { "CF_DUMPVERB", CFD_COLHEAD,    ' ',    CF_INT|CF_NOSAVE|CF_LAST,  },
 };
```

The Configuration Initializer controls the behavior and actions of all **cf**∗() functions, starting with the control of Parsing Levels in the Initialization Process.

6.1 Parameter Name

Every entry in the initializer must have a name, that is a non-empty string, which is searched for in the environment and in the configuration files and which is used as an argument to the inquiry functions. Parameters that have a name beginning with **"CF_"** may have special meaning, see CFLIB Parameters.

6.2 Parameter Default Value

As the default content of an entry you can specify a `NULL` pointer or a string, which may be empty. If CF_FLAG is set for that entry you should use the CF_FLAG_ON or CF_FLAG_OFF macros.

6.3 Commandline Option for Parameter

You can control whether and how an entry's content can be set from the Commandline by setting one of the following **"Option Specifier"** Characters (and Special Option Flags where indicated):

- '' (blank)
  - No Commandline Option for this Parameter
- '*c*' (*c* = any "normal" character)
  - Looks for '-c <*string*>' on the Commandline
  - Looks for '-c<*string*>' on the Commandline, if Flag CF_CONCAT is set in the Special Options Mask
  - Looks for '-c<*char*>' on the Commandline, if Flag CF_FLAG is set in the Special Options Mask
- '*#*' (*#* = a positive number = 1, 2, 3, ...)
  - Looks for the 1st, 2nd, 3rd, ... '<*string*>' that is not part of an option, if Flag CF_NO_OPT_ARG is set in the Special Options Mask

6.4 Special Option Flags Mask for Parameter

Most of the magic happens here! ;-) The **"Special Options Mask"** is a Bitmask of type CFFLAGTYP in Configuration Entry Structure Member CONFIG::flag containing Type, Instruction and Information Flags for a Parameter. See Special Options Mask for details.

See also

Types of Parameters
Configuration Parsing Levels and Source/Origin Options

Attention

The last entry in the initializer list must have the CF_LAST Flag set!

7 Configuration Parsing Levels and Source/Origin Options

CFLIB is designed to give you a maximum of possibilities for feeding parameters and parameter values into the configuration.

7.1 Origin Window

The library remembers the source/origin of all parameter values by setting one of the **"Source Flags"** in the **"-Origin Window"** of the Special Options Mask ranging from Bit Offset 8 to 15 (see Parsing Levels). On initialization the Origin Window Flags **CF_SET_**$*$ of an entry in the Configuration Initializer are used to control Parsing Level Priorities.

7.2 Parsing Levels

"Parsing Levels" are equal to the Relative Bit Offset of the corresponding **CF_SET_**$*$ Bit within the "Origin Window":

Parsing Level : Description (Source Flag)

- 0 = CFP_PUT : Function Call or Automatic Initialization. (CF_SET_PUT)
  - 1 = CFP_ARG : Commandline. (CF_SET_ARG)
  - 2 = CFP_ENV : Environment. (CF_SET_ENV)
  - 3 = CFP_PRIV : Private Configuration File. (CF_SET_PRIV)
  - 4 = CFP_SYS : System Configuration File. (CF_SET_SYS)
  - 5 : CFP_DEF : Built-in Default. (CF_SET_DEF)
  - 6 : CFP_QRY : Standard Input Channel. (CF_SET_QRY)
  - 7 : CFP_RESERVED : Reserved for CFLIB Subprojects

7.3 Parsing Levels in the Initialization Process

On initialization the Parsing Levels 1 through 6 are processed by cfinit() in that order and priority. If a Parameter is found that has been defined in the Configuration Initializer, its content is included as String Content of that parameter in the configuration database.

Whether and how the **Commandline** Arguments are used to get a parameter value depends on the CONFIG::option setting for that parameter and the Special Option Flags:

- CF_CONCAT : Argument is concatenated to option.
  - CF_NO_OPT_ARG : Commandline argument not following an option.

See details under Commandline Option for Parameter.

The **Environment** can be excluded as a possible source of a parameter value by setting the Special Option Flag

- CF_IGN_ENV : Do not look for Environment Variable

Configuration Files are only used when corresponding entries are defined in the Configuration Initializer, see Parsing of Configuration Files for details.

Interactive queries are not performed by default. If cfinit() shall ask the user for interactive input of items that are still unresolved after parsing all lower levels, you must set Special Option Flag:

- CF_QUERY : Ask the user for unresolved item after configuration parsing.

If an entry may not be empty (`NULL` or string "") after initialization, you can have cfinit() produce an appropriate error by setting the Special Option Flag:

- CF_MUST : Entry may not be empty (`NULL` or "")

7.3.1 Parsing Level Priorities

More precise control of parsing levels priorities and origin control for a parameter is available through setting one of the Flags in the "Origin Window" of the parameter's Special Options Mask in the Configuration Initializer :

- CF_SET_PUT : Must be set by a function call / automatic processing: Set this Flag on Time or Date Types of Parameters for automatic initialization with *now* or *today*
- CF_SET_ARG : Force initialization from commandline, ignore anything else
- CF_SET_ENV : Let environment variable override earlier setting
- CF_SET_PRIV : Let variable from private configuration file override earlier setting
- CF_SET_SYS : Let variable from system configuration file override earlier setting
- CF_SET_DEF : Let default from Configuration Initializer override earlier setting
- CF_SET_QRY : Force setting from interactive Query (`stdin`)

7.4 Parsing of Configuration Files

To indicate that an entry in the initializer list given to cfinit() represents the filename of one of the Configuration Files, set one of the following Flags for that parameter:

- CF_SETFILE : Entry is Private Configuration File.
  - CF_SYS_SETFILE : Entry is System Configuration File.

Within the system configuration file you can choose a section, from which information is read as from a single file. To specify an entry in the list referring to the name of that section give this entry the flag

- CF_SECTION : Section in Configuration File.

As the filenames for the two configuration files are themselves entries in the database, these levels are revisited, if neccessary, in reverse order after parsing the built-in default.

7.5 Residual Items

Any commandline argument that did not match conditions for a parameter will be included in the configuration database as parameter without name marked with Flag CF_RESID in the Special Options Mask. These additional entries from the command line can be accessed by successive calls to cfgetres().

Residual items from one of the configuration files will also be accumulated in the database and can be accessed through inquiry by name with one of the Retrieval Functions. This is especially useful together with the report generation function cfform().

8 Configuration Files

A major task of the library is handling import and export of configuration parameters from/to files.

CFLIB knows two types of configuration files:

- **"Private Configuration File"** - User and/or program specific file in simple format to be read on initialization and optionally be updated by cfsave()

- **"System Configuration File"** - System and/or project specific file in extended format (supporting sections, see below) will only be used as a data source by cfinit() and will not be touched by cfsave() unless you explicitly specify the filename.

8.1 Configuration File Format

In a configuration file lines beginning with '#' are treated as comments and are ignored. Blank lines are ignored, too. A valid line in the file is of the form:

<*name*> = <*entry*>

Blank chars around the '=' are ignored. The name must match one of the entry's names in the initializer. In fact, any line not containing a '=' will be ignored, but it's better to indicate comments with '#' !

The optional sections in the system configuration file begin with a line like:

[<*sectionname*>]

and end with another line like this or with the file's last line. Anything after the closing bracket is ignored.

A simple example of a valid configuration file could look like this:

```
# This is my private configuration file for Project 1356 Branch C in spe

Search_Path  = /my/subproject/directory:/general/settings/directory
Section      = project_1356
Outfile      = my_subproject.cnf
ask_if_empty =
My_Flag =      ON
VERBOSITY = 1
```

The corresponding system configuration file could look like this:

```
# This is a system wide configuration file
[some_other_program]
blah = blubber
...
[project_1356]
# Settings for Project 1356 Branch B
VERBOSITY = 3
Outfile = project_1356b.cnf
X_EUR_USD=1.4562
[some_other_project]
...
```

8.2 Reading Configuration Files

A search for configuration files and import of data from these sources is only performed by the function cfinit() on initialization and only when appropriate Special Option Flags are set as described under Parsing of Configuration Files.

Configuration files are read once when the configuration database is initialized by a call to cfinit() or cfstart() using the internal function cfreadfile().

8.3 Writing a (private) Configuration File

Parameter export does not depend on any specific setting and can be performed whenever and as often as you like. For writing a configuration file call cfsave() with either the name of the file or NULL, in which case the current value of the "Private Configuration File" parameter will be used, if it exists. If an entry has the CF_NOSAVE flag set, it is excluded from saving. The "System Configuration File" may not be referred to directly.

Attention

Writing **section** marks is **not** supported!

9 Types of Parameters

The internal format of a parameter's content CONFIG::inhalt is always a String (char Pointer).

The use and interpretation of parameters is controlled by a number of Flags in the Special Options Mask :

9.1 Parameter Data Types

- CF_STR : Entry is String. (**Default**).
- CF_INT : Entry is Integer.
- CF_REAL : Entry is Float.
- CF_FLAG : FLAG Parameter Type

9.2 Special Processing Instructions

Specific processing of a String Parameter's Value/Content in various CFLIB functions can be triggered by these Flags:

You can mark a parameter as Time **or** Date. These parameters will be initialized with *now* or *today* by cfinit() when Parsing Level Priorities are controlled by a CF_SET_PUT Flag in the Special Options Mask or on call of cfputtime():

- CF_TIME : Time string.
  - CF_DATE : Date string.
- CF_FINDFILE : Entry is filename to be searched for in the path. FindFile() will be used on initialization to search the file in the **path** found in Environment Variable **PATH**. You can specify an alternative search path for this functionality in another database parameter with Flag CF_PATH set in the corresponding Configuration Initializer entry. Depending on the Operating System a list of possible of **extensions** of executables will be tried, if no extension is given with the filename:
  - Linux, Unix: "sh", "pl"
  - DOS,Windows, OS2: "exe", "com", "bat", "cmd"
  - TOS, MinT: "ttp", "tos", "prg", "app", "gtp"
- CF_EXPHOME : Expand Home Directory. See cfhomexp(). Configuration Files will have home directory expanded by default

9.3 Special Handling Instructions

A parameter can be marked as a **"volatile"** entry that is not saved by cfsave():

- CF_NOSAVE : Don't include in savefile / mark entry.

9.4 CFLIB Parameters

Some String Parameters in the configuration database are used by the library itself and are marked by one of the following Flags:

- CF_PRGNAME : Running Program's Name from commandline. This parameter **always** exists as the **first** entry in a configuration database successfully initialized by cfinit(). The default parameter name used if there was no corresponding entry in the Initializer is **"CF_PRGNAME"**.
  - CF_SETFILE : Entry is Private Configuration File. See Configuration Files
  - CF_SYS_SETFILE : Entry is System Configuration File. See Configuration Files
  - CF_SECTION : Section in Configuration File. See Configuration Files
  - CF_PATH : Search Path (for FindFile feature) See CF_FINDFILE and FindFile()
  - CF_USAGE : Usage Message format string. See cfgetusg()

Optional Parameters for fine tuning CFLIB behavior without Special Option Flag, characterized by Parameter Name:

- **"CF_DUMPVERB"** - Non-Default Verbosity Mode for cfdump()

9.5 FLAG Parameter Type

The entry is treated like a boolean variable and the string content set from any of the Configuration Parsing Levels and Source/Origin Options except the default setting is interpreted as follows:

- CF_FLAG_OFF : String Content is:
  - beginning with a `'-'`
  - `"OFF"` or `"off"`
  - `"FALSE"` or `"false"`
- CF_FLAG_ON : any other case

Example:

- Initializer entry: { "extended_message", CF_FLAG_OFF, 'x', CF_FLAG, }
- Commandline: myprog -x-
- Configuration file: extended_message = OFF

10 Advanced Usage Example

Setup Configuration Initializer, generate Report and save Configuration Files

```
/*
 *    Generate Report from Template and Configuration
 *
 *    Input:  - Commandline, Environment, Section in System Configuration File,
 *              Private Configuration File, Standard Input for Variables.
 *            - Text Template (Form).
 *
 *    Output: - Generated Report (Filled Form) to File or stdout,
 *            - Configuration to Private Configuration File, Logfile or
 *              Standard Output
 */

#include <stdio.h>
```

```
#include "../../include/cf.h"

#define DEF_SETFILE "~/fill.cnf"
#define DEF_SYS_SETFILE "/etc/cfcommon.cnf"

int main( int argc, char **argv )
{
    int ecode;
    char *save, *infile, *outfile, line[CF_MAXERRSTR+1];
    char *savemode = "w";
    FILE *error_log = stderr;

    /* Hardcoded Configuration Setup and Defaults */
    CONFIG setting[] =
    {   { "FILL_INFILE",   NULL,             'i', 0x0,                          }
      ,
        { "FILL_OUTFILE",  NULL,             'o', 0x0,                          }
      ,
        { "FILL_VARDELIM", "$()",            'l', CF_CONCAT,                    }
      ,
        { "verb",          CF_FLAG_OFF,      'v', CF_IGN_ENV|CF_FLAG,           }
      ,
        { "help",          CF_FLAG_OFF,      'h', CF_IGN_ENV|CF_FLAG|CF_NOSAVE,
      },
        { "query",         CF_FLAG_OFF,      'q', CF_IGN_ENV|CF_FLAG,           }
      ,
        { "TIME",          NULL,             't', CF_TIME|CF_SET_PUT,           }
      ,
        { "DATE",          NULL,             'd', CF_DATE|CF_SET_PUT,           }
      ,
        { "FILL_CNF",      DEF_SETFILE,      'p', CF_SETFILE,                   }
      ,
        { "FILL_SAVE",     NULL,             's', CF_IGN_ENV|CF_CONCAT,         }
      ,
        { "SAVE_DATA",     CF_FLAG_OFF,      'x', CF_FLAG|CF_IGN_ENV,           }
      ,
        { "SFILE",         DEF_SYS_SETFILE,  ' ', CF_SYS_SETFILE,               }
      ,
        { "FILL_SECTION",  "fill_default",   'c', CF_SECTION|CF_LAST,           }
      ,
    };

    /* Initialize with Error Reporting */
    if( cfinit(setting,argc,argv) < 0 ){
        while( (ecode=cfgeterr(line,0)) == TRUE )
            fprintf( error_log, "error %d: %s", ecode, line );
        fflush(error_log);
        fputs( cfgetusg(), stderr );
        return (-1);
    }

    /* User Help and Configuration Overview */
    if( cfgetflag("help") ){
        fputs( cfgetusg(), stderr );
        if( cfgetflag("verbose") ){
            fprintf( stderr, "\nConfiguration Library PL %d - %s\n\nCurrent
       Configuration:\n\n",
                              cfgetvers(), cfgetcpr() );
            cfdump(stderr);
            fputs( "\nSources: 1=update function call, 2=commandline argument,
       4=environment variable,\n", stderr );
            fputs(   "         8=private setfile, 10=system setfile,
       20=built-in default, 40=interactive terminal input\n", stderr );
         }
         return (1);
    }

    switch(
        cfform( infile = cfgetstr("FILL_INFILE"),
                outfile = cfgetstr("FILL_OUTFILE"),
                cfgetstr("FILL_VARDELIM"),
                cfgetflag("query") )
    ){
```

```
        case CFE_FNF:
            fprintf(stderr,"read access error: %s\n",infile); break;
        case CFE_WAE:
            fprintf(stderr,"write access error: %s\n",outfile); break;
    }

    if( cfgetflag("verb") ){
        if( outfile == NULL ) outfile = "<stdout>";
        fprintf( stderr, "blank form: %s, filled form: %s\n", infile, outfile )
      ;
    }

    if( cfgetflag("SAVE_DATA") ){
        cfnosave(NULL, CF_FLAG_ON);
        cfnosave("TIME", CF_FLAG_OFF);
        cfnosave("DATE", CF_FLAG_OFF);
    }
    if( (save = cfgetstr("FILL_SAVE")) != NULL ){
        if( cfgetflag("SAVE_DATA") ) savemode = "a";
        if( *save == '\0' )
            cfsave( NULL, savemode );
        else if( *save == '+' )
            cfsave( "", savemode );
        else
            cfsave( save, savemode );
    }

    return (0);
}
```

11 Compilation and Development

11.1 Library Versions

There is no strict versioning of the library. A Library **"Patchlevel"** is defined as an Integer Number counting **"Major Versions"**. See cfgetvers().

The individual source files contain more detailed version information and they all support a file version identifier `$Id$ for` automatic update by SVN and other Source Management Tools. See cfgetsubvers().

11.2 Platforms

The library source code is fairly simple ANSI C code and should compile and link without problems on most platforms. Most of the development has been done with different versions of the **"GNU C Compiler"** `gcc` and related tools. All source modules compile free of errors and warnings with:

`"gcc -pedantic -pedantic-errors -Wall -Werror -ansi ..."`

So, if you have `gcc`, use it! Any other ANSI C compiler should also work, at least after minimal adaption to the library setup, see System and Compiler dependent Defines.

until now the library has been built and used under the following setups:

- gcc, Linux, Intel PC
  - gcc, MinGW, MS Windows NT/XP, Intel PC
  - gcc, MinT/TOS Atari ST
  - cc, Unix, IBM AIX
  - cc, Unix, SGI
  - MSC, MS DOS/Windows Intel PC
  - MSC, OS/2 Intel PC

11.3 Building the Library

Compiling the source modules and building the library should be a "straight forward" task:

- Make all objects from C sources, including *cf.h* and *cflib.h*
- Link the objects with `ar` or another tools to get a library executable

See also

Makefiles under CFLIB project tree and System and Compiler dependent Defines

11.4 Binaries and Executables

- The library project should be seen as "pure" source/text code on distribution. Compile a library executable with a C compiler of your choice with appropriate setup for your platform.
- Binary and executable versions of the library found in the project tree should be seen as examples that worked under one specific setup but have not been intensively tested and may even not be up to date. If it works for you, feel free to use them. If you build a library executable which is not too ;-) dependent on a specific setup, you can include it as example in the project tree.

11.5 Minimal CFLIB Replacement

In project directory *src/examples* you may find a source file *cf_minimal.c*. You can adapt this example to your need and include it in your C source list on compilation instead of linking the library executable. This makes user programs that use CFLIB functions independent of the library at the price of very reduced functionality, which may be desirable for specific executables or if you have problems with building the library on the target platform.

11.6 System and Compiler dependent Defines

The following Defines are used in the library source to decide whether to include certain header files and use certain functions or defines:

- **_HAS_PWD** : If defined, include <**pwd.h**> and make use of function **getpwnam()** in cfhomexp() to find a user's home directory
  - **_HAS_ISATTY** : If defined, use function **isatty()** to determine whether a stream is a terminal (for interactive query/input) in function IsATerminal()
  - **_HAS_LIMITS** : If defined, include <**limits.h**> and use **PATH_MAX** defined therein in function FindFile()

The following Defines can be used to control, which features or functions shall be excluded from the library build:

- **_CF_NOFINDFILE** : Function FindFile(), component findfile.c
  - **_CF_NOSAVING** : Functions cfsave() and BackupFile(), component cfwrite.c
  - **_CF_NODEBUGGING** : Functions cfdinichk() and cfdump(), component cfdebug.c

The following Defines can be used to switch on certain Debug Features:

- **DEBUG_DINICHK** : Debug cfdinichk()
  - **DEBUG_ERROR** : Debug cfputerr()

  - **DEBUG_TIME** : Debug cfputtime()
  - **DEBUG_NOSAVE** : Debug cfsave()
  - **DEBUG_BACKUP** : Debug BackupFile()
  - **DEBUG_FINDFILE** : Debug FindFile() usage in cfinit()
  - **DEBUG_FORM** : Debug cfform()

The following Defines can be used to switch on certain other Features:

- **_PREFER_BACKSLASH** : Prefer Backslash as Directory Separator in function FindFile()
  - **_PATHSEP_SEMICOLON** : Use Semicolon as Path Separator in function FindFile(), usually on systems where a ":" can appear in a directory path
  - **_PATHSEP_COMMA** : Test Comma as Alternative Path Separator in function FindFile()
  - **_CF_RESID_FREE** : Remove residual arguments after having read them all in cfgetent()?

The following Platform dependent Defines are used in the library code:

- **unx**
  - **linux**
  - **atarist**
  - **MINT**

12 Todo List

Global cfdinichk (CONFIG *set)

Make cfdinichk() work as reliable, complete tool with much more testing and Special Options Mask validation!

Global cfform (char *infile, char *outfile, char *vardelim, int mode)

Make cfform() work with buffers instead of files

Page General Notes

Make proper Man Pages with function references etc.

Make Doxygen Developer Documentation

More Modularization! Still very similar code in various functions

13 Bug List

Group errors

There are still errors without entry in error stack

Global IsATerminal (FILE *fp)

ANSI C doesn't have function isatty(), we always return TRUE

14 Module Index

14.1 Modules

Here is a list of all modules:

15 Data Structure Index

15.1 Data Structures

Here are the data structures with brief descriptions:

16 File Index

16.1 File List

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

17 Module Documentation

17.1 Flexible Configuration Library

CFLIB is a small, simple, flexible and portable ANSI C Library to be used as configuration interface for user programs.

Modules

- Core Features
  Basic CFLIB Setup.
- Report Generation
  Process templates doing variable substitution and file inclusion.
- General Utilities
  General Utility Macros and Functions.
- Special Options Mask

*The **"Special Options Mask"** is a Bitmask of type CFFLAGTYP in Configuration Entry Structure Member CONFIG::flag containing Type, Instruction and Information Flags for a Parameter.*

- Error Handling

  Error Codes, Functions and Structures.
- Advanced Features

  Debugging and Utility Functions.
- Information Retrieval

  These functions and macros read entries from an initialized CFLIB database.
- Setting and Saving the Configuration

  Set/Update Parameter Values or Save a Configuration File.

17.1.1 Detailed Description

CFLIB is a small, simple, flexible and portable ANSI C Library to be used as configuration interface for user programs. CFLIB builds and maintains a compact database structure consisting of a list of parameters with their name, content and some additional information about each parameter.

CFLIB targets the basic needs of technical, scientific or other programmers who want to spend minimal time on coding input, output, variable parsing, report generation and the like but still have a simple to use, reliable, flexible and portable configuration interface for their programs.

Main Features

- Commandline, environment and terminal input parsing
  - Configuration files
  - File search
  - Template driven report generation
  - Automatic time and date update

Author

Stefan Habermehl

License:

http://www.gnu.org/licenses GNU General Public License v3 or later

Project Homepage:

Detailed information, source code and maybe updates are available from the library homepage:

- http://cflib.berlios.de

Local References:

- Include file: *cf.h*
  - Library file: *libcf.a*
  - Documentation: Flexible Configuration Library

17.2 Core Features

Basic CFLIB Setup.

Data Structures

- struct CONFIG

  CFLIB Configuration Database Entry.

Defines

- #define Patchlevel "21"

  CFLIB Identification.
- #define MAXCONF 4096

  Maximum number of entries in configuration database.
- #define CF_MAXERRSTR 512

  Maximum string length for error message.
- #define CF_MAXLINE 20480

  Maximum string length for setfile and form parsing.
- #define CF_MAXQLINE 512

  Maximum string length for query.
- #define CF_MAXUSAGE 1024

  Maximum string length for usage string.
- #define CF_USG_DEFCOLS 80

  Default terminal width for usage string.
- #define CF_MAXTIMEBUF 256

  Buffer size for time and day.
- #define FALSE 0

  < TRUE, if not defined
- #define NULL (void *)(0L)

  NULL, if not defined.
- #define CF_FLAG_ON "\1"

  Flag is set.
- #define CF_FLAG_OFF ""

  Flag is not set.
- #define CF_NO_OPTION ' '

  Option is not set.
- #define TABLEN 8

  TAB length.
- #define CFP_PUT 0

  Function Call or Automatic Initialization.
- #define CFP_ARG 1

  Commandline.
- #define CFP_ENV 2

  Environment.
- #define CFP_PRIV 3

  Private Configuration File.
- #define CFP_SYS 4

  System Configuration File.
- #define CFP_DEF 5

  Built-in Default.

- #define CFP_QRY 6

  Standard Input Channel.
- #define CFP_RESERVED 7

  Reserved for Subprojects.
- #define CFS_NOT 0

  Error Response Modes for cfstart()
- #define CFS_ALL 1

  Start Mode: All error messages.
- #define CFS_NEG 2

  Start Mode: Only severe errors.
- #define CFS_USG 3

  Start Mode: Usage message if error was negative, error output like CFS_NEG.
- #define CFS_DEBUG 4

  Start Mode: Output like CFS_USG plus raw dump of configuration.

Functions

- int cfinit (CONFIG *set, int argc, char **argv)

  cfinit() Initialize Configuration Database
- void cfexit (void)

  cfexit() Free allocated memory and reset configuration database and error stack
- int cfstart (CONFIG *set, int ac, char **av, char *help, int mode)

  cfstart() Start Configuration Database (with error reporting and usage message)

Variables

- CONFIG * _conf [MAXCONF+1]

  Library Internal: CFLIB Configuration Database with at most MAXCONF parameters.

17.2.1 Detailed Description

Basic CFLIB Setup.

- General Defines
- Parsing Levels
- Configuration Entries (aka Parameters)
- Configuration Database: Array of Configuration Entries
- Initialization of the Configuration Database
- Resetting the Configuration Database

17.2.2 Define Documentation

17.2.2.1 #define FALSE 0

< TRUE, if not defined

FALSE, if not defined

17.2.2.2 #define CFS_NOT 0

Error Response Modes for cfstart()

Start Mode: No action on error

17.2.3 Function Documentation

17.2.3.1 int cfinit (CONFIG * *set*, int *ac*, char ** *av*)

cfinit() Initialize Configuration Database

Initialize CFLIB Configuration Database and parse possible sources for database entries according to the settings in *set*.

Parameters

set	Configuration Initializer : Pointer to Array of CONFIG items containing parameter name, default value, Commandline Option for Parameter and Special Options Mask
ac	Argument Count from commandline or compatible
av	Argument String array from commandline or compatible structure

Returns

- 0 : configuration database initialized successfully
- >0 : Count of non fatal errors
- <0 : Fatal error(s) occurred:
  - CFE_MCF : Memory allocation for Configuration Failed.
  - CFE_MEF : Memory allocation in Error routine Failed.
  - any other absolute value is total error count

See also

Configuration Parsing Levels and Source/Origin Options

17.2.3.2 void cfexit (void)

cfexit() Free allocated memory and reset configuration database and error stack

Free allocated memory and reset the configuration database and error stack.

This function should be used before a repeated call to cfinit() or cfstart(). Furthermore, it may be desired to call cfexit() when the configuration database occupies a lot of memory and is no longer needed.

17.2.3.3 int cfstart (CONFIG * *setting*, int *ac*, char ** *av*, char * *help*, int *mode*)

cfstart() Start Configuration Database (with error reporting and usage message)

Setup Configuration Database.

Check errors and output usage message, if required

Parameters

setting	pointer to initializer
ac	argument count from commandline or compatible
av	argument string array from comandline or compatible
help	Help/Usage String • Name of a FLAG Parameter Type entry to cause a usage message, if flag is set • Omit, if NULL

mode	How to handle cfinit() return: • CFS_NOT - Error Response Modes for cfstart() • CFS_ALL - Start Mode: All error messages. • CFS_NEG - Start Mode: Only severe errors. • CFS_USG - Start Mode: Usage message if error was negative, error output like CFS_NEG. • CFS_DEBUG - Start Mode: Output like CFS_USG plus raw dump of configuration.

Returns

- `0` : configuration database has been initialized successfully, no help or usage message required, no fatal errors
- `1` : no fatal errors occurred, usage message displayed
- `-1` : fatal error(s) occurred

17.3 Report Generation

Process templates doing variable substitution and file inclusion.

Defines

- #define CF_DEF_VARDELIM "$()"

  Default variable delimiter for cfform()
- #define CF_MAXINC 8

  Maximum number of nested includes for cfform()

Functions

- int cfform (char *infile, char *outfile, char *vardelim, int mode)

  cfform() Process a Template from file or `stdin` and write generated Report to File or `stdout`

17.3.1 Detailed Description

Process templates doing variable substitution and file inclusion.

17.3.2 Function Documentation

17.3.2.1 int cfform (char * *file*, char * *outfile*, char * *vd*, int *mode*)

cfform() Process a Template from file or `stdin` and write generated Report to File or `stdout`

Process a Template from file or `stdin` and write generated Report to File or `stdout`.

The Report Template contains **Variables** like '$(*ident*)' or whatever you set in *vd*. *ident* may refer to a:

- Parameter Name like '$(*name*)' : Substitute variable by parameter's value (from DB)
  - Include File Path like '$(FILE:*/my/path/to/incfile*)' : Substitute variable by content of include file
  - File Path and Section Include like '$(FILE:*incfile#sect)'* : Include section from include file between [*sect*] and next [...]
  - Variable File Include like '$(FILE::*varname*)' : Include file from location given in parameter *varname*

See also

Advanced Usage Example

Todo Make cfform() work with buffers instead of files

Parameters

file	Name of Template File, `NULL` for `stdin`
outfile	Name of Output File, `NULL` for `stdout`
vd	Variable Delimiters: string containing the three variable delimiters in the first three chars: to have variables like '$(*name*)', let the string be '$()'. This is also the default, if *vd* is `NULL` or string length < 3
mode	Mode Mask: • `0` : Normal • `1` : Query for unresolved variables, (try to) include them in database • `2` : Unset CF_NOSAVE flag for parsed vars • `4` : Outfile Write *append*, else *write*

Returns

- 0 : Configuration has been saved to file successfully
- !=0 : An error occurred:
  - CFE_NCA : No Configuration database Available.
  - CFE_FNF : File Not Found, read access error.
  - CFE_WAE : Write Access Error.

17.4 General Utilities

General Utility Macros and Functions.

Defines

- #define CF_BACKBUFLEN 102400

  File copy buffer for BackupFile()
- #define DelFlag(a, b) a&=(~b)

  Delete bits of Mask `b` from Mask `a`.
- #define SetFlag(a, b) a|=b

  Set bits of Mask `b` in Mask `a`.

Functions

- int BackupFile (const char *file, char *modus)

  BackupFile() Copy or Rename File "`file`" to Backup File "`file~`" or "`file.bak`"
- void RemoveCR (char *string)

  Strip Carriage Return at end of string (after fgets) by introducing zero byte at CR position.
- void RemoveTrailSpace (char *string)

  Strip whitespaces at end of string by introducing zero byte after last non-whitespace character.
- char * EatWhiteSpace (char *string)

  EatWhiteSpace() Set pointer to next non-whitespace-character in string.
- int IsATerminal (FILE *fp)

  IsATerminal() Test whether stream is a terminal
- char * FindFile (const char *fname, const char *fpath, const char *const *fext)

  FindFile() Find a File in Path trying Extensions

17.4.1 Detailed Description

General Utility Macros and Functions.

17.4.2 Function Documentation

17.4.2.1 int BackupFile (const char * *file*, char * *modus*)

BackupFile() Copy or Rename File "`file`" to Backup File "`file~`" or "`file.bak`"

Copy or Rename File "`file`" to Backup File "`file~`" or "`file.bak`".

Parameters

file	Name of regular file to be opened
modus	Backup Mode: • "`r`" Rename • "`d`" Duplicate, Copy • "`@e ?`" Default "`d`" • "`@e ?d`" docs-style (.bak) • "`@e ?c`", "`??`" Un*x Style (~)

Returns

- 0 : Configuration has been saved to file successfully
- !=0 : An error occurred:
  - CFE_BOF : Invalid filename / open error
  - CFE_BMF :Backup: Memory allocation Failed.
  - CFE_BRF : Rename file failed
  - CFE_BBF : Source file open error
  - CFE_BWF : Target file write error

17.4.2.2 void RemoveCR (char * *ptr*)

Strip Carriage Return at end of string (after fgets) by introducing zero byte at CR position.

String Utility Function.

Strip Carriage Return at end of string (after fgets) by introducing zero byte at CR position. Original source was "STELM" by Kees and Lemmens.

Author

Kees and Lemmens

Parameters

ptr	pointer to beginning of string

17.4.2.3 void RemoveTrailSpace (char * *ptr*)

Strip whitespaces at end of string by introducing zero byte after last non-whitespace character.

Parameters

ptr	Pointer to beginning of string

17.4.2.4 char * EatWhiteSpace (char * *ptr*)

EatWhiteSpace() Set pointer to next non-whitespace-character in string.

Set pointer to next non-whitespace-character in string.

Original source was "STELM" by Kees and Lemmens.

Author

Kees and Lemmens

Parameters

ptr	Pointer to beginning of string

Returns

Pointer to next non-whitespace-character in string

17.4.2.5 int IsATerminal (FILE * *fp*)

IsATerminal() Test whether stream is a terminal

Test whether stream is a terminal.

Parameters

fp	File/Stream Pointer

Returns

TRUE or FALSE

Bug ANSI C doesn't have function isatty(), we always return TRUE

17.4.2.6 char * FindFile (const char * *fname,* const char * *fpath,* const char *const * *fext*)

FindFile() Find a File in Path trying Extensions

Find a File in Path trying Extensions.

Author

Eric R. Smith

License:

Public Domain

Parameters

fname	File Name
fpath	Search Path: String of Directories separated by **PATHSEP1** (':' or ';') or **PATHSEP2** (nothing or ',')
fext	Array of possible Extensions (optional, default is OS dependent)

Returns

The name by which the file was found or NULL

17.5 Special Options Mask

The **"Special Options Mask"** is a Bitmask of type CFFLAGTYP in Configuration Entry Structure Member CONFIG::flag containing Type, Instruction and Information Flags for a Parameter.

Defines

- #define CF_LAST 0x0001

  Last entry in settings array.
- #define CF_PATH 0x0002

  Search Path (for FindFile feature)
- #define CF_SETFILE 0x0004

  Entry is Private Configuration File.
- #define CF_SYS_SETFILE 0x0008

  Entry is System Configuration File.
- #define CF_SECTION 0x0010

  Section in Configuration File.
- #define CF_PRGNAME 0x0020

  Running Program's Name from commandline.
- #define CF_TIME 0x0040

  Time string.
- #define CF_DATE 0x0080

  Date string.
- #define CF_SET_PUT 0x0100

  Source: Function Call.
- #define CF_SET_ARG 0x0200

  Source: Commandline Argument.
- #define CF_SET_ENV 0x0400

  Source: Environment Variable.
- #define CF_SET_PRIV 0x0800

  Source: Private Configuration File.
- #define CF_SET_SYS 0x1000

  Source: System Configuration File.
- #define CF_SET_DEF 0x2000

  Source: Built-in Default.
- #define CF_SET_QRY 0x4000

  Source: Interactive Terminal Input.
- #define CF_NO_OPT_ARG 0x10000

  Commandline argument not following an option.
- #define CF_CONCAT 0x20000

  Argument is concatenated to option.
- #define CF_IGN_ENV 0x40000

  Do not check environment for variable.
- #define CF_QUERY 0x80000

  Ask the user for unresolved item after configuration parsing.
- #define CF_STR 0x100000

  Entry is String.
- #define CF_INT 0x200000

  Entry is Integer.
- #define CF_FLAG 0x400000

  Entry is Flag.

- #define CF_REAL 0x800000

  Entry is Float.
- #define CF_FINDFILE 0x1000000

  Entry is filename to be searched for in the path.
- #define CF_MUST 0x2000000

  Entry may not be empty (`NULL` or "")
- #define CF_RESID 0x4000000

  Residual/additional entry from commandline/setfile/cfput.
- #define CF_USAGE 0x8000000

  Usage Message format string.
- #define CF_EXPHOME 0x10000000

  Expand Home Directory.
- #define CF_MALLOC 0x20000000

  Space for entry's content was mallocated.
- #define CF_FORCED 0x40000000

  Setting has been forced (already)
- #define CF_NOSAVE 0x80000000

  Don't include in savefile / mark entry.
- #define CF_SRC (CF_INT|CF_FLAG)

  Type for source/origin inquiry.
- #define CF_FLGINQ (CF_STR|CF_FLAG)

  Type for options mask inquiry.
- #define CF_DOUBLE (CF_REAL|CF_FLAG)

  Type for double inquiry.
- #define CF_TD (CF_DATE|CF_TIME)

  Date or Time entry.

Typedefs

- typedef unsigned long CFFLAGTYP

  Special Options Mask Type.

17.5.1 Detailed Description

The **"Special Options Mask"** is a Bitmask of type CFFLAGTYP in Configuration Entry Structure Member CONFIG::flag containing Type, Instruction and Information Flags for a Parameter.

- Special CFLIB properties: CFLIB Parameters
- Source/Origin: Parsing Levels
- Initialization options: Parsing Levels in the Initialization Process
- Type/Interpretation Flags: Parameter Data Types
- Special Processing Instructions
- Special Handling Instructions
- Information/Status markers:
  - CF_LAST must appear in the Configuration Initializer
  - CF_RESID marks one of the Residual Items
  - CF_MALLOC and CF_FORCED are for library internal use

17.6 Error Handling

Error Codes, Functions and Structures.

Data Structures

- struct CONFERR

  Library Internal: Error List Item.

Defines

- #define CFE_INIT 0

  INITialize error input.
- #define CFE_OK 0

  No error / everything OKay.
- #define CFE_NEP 1

  New Entry successfully Put into DB.
- #define CFE_EXIT 1

  Finish error input.
- #define CFE_ORA 20

  Option Requires an Argument.
- #define CFE_UKO 25

  UnKnown Option.
- #define CFE_FNF 30

  File Not Found, read access error.
- #define CFE_NSE 40

  No Section specifier End found, missing "]".
- #define CFE_NSC 50

  No private Setfile Configured.
- #define CFE_WAE 60

  Write Access Error.
- #define CFE_IFP 61

  Invalid Filename entry for Private setfile.
- #define CFE_EWN 70

  Entry Without Name.
- #define CFE_ICF 80

  Invalid Combination of Flags.
- #define CFE_EWC 90

  Entry Without Content.
- #define CFE_UOS 100

  Unlikely Option Specifier.
- #define CFE_IFC 110

  Invalid Flag Combination.
- #define CFE_NLE 120

  No Last Entry flag found.
- #define CFE_TIN 130

  Error reTurn from stdIN query.
- #define CFE_EFE 140

  Empty string in content for Filename Entry.
- #define CFE_USG 200

  Entries missing: USaGge advice.

- #define CFE_URI 210

  UnResolved Item (CF_MUST was set!)
- #define CFE_FBF -500

  File Backup Failed.
- #define CFE_BMF -510

  Backup: Memory allocation Failed.
- #define CFE_BOF 520

  Backup: Open source file Failed.
- #define CFE_BBF 530

  Backup: open target Backup file Failed.
- #define CFE_BRF 540

  Backup: Rename Failed.
- #define CFE_BWF 550

  Backup: Write Failed.
- #define CFE_NCA -10

  No Configuration database Available.
- #define CFE_NEA -20

  No Entry with that name Available.
- #define CFE_NSS -30

  No Source/origin is Set.
- #define CFE_ECP -40

  Entry's Content is a NULL Pointer.
- #define CFE_MEF -100

  Memory allocation in Error routine Failed.
- #define CFE_MCF -200

  Memory allocation for Configuration Failed.
- #define CFE_INF -9999

  Integer iNquiry Failed (?!)
- #define CFE_RNF -999.999

  Real/float iNquiry Failed (?!)

Functions

- int cfgeterr (char *string, size_t len)

  cfgeterr() Error Code and Message Inquiry Function
- int cfputerr (int ecode, char *string,...)

  cfputerr() Init, exit or append to Error List
- void cfclearerr (void)

  cfclearerr() Free all entries in error list
- int cfreverr (void)

  cfreverr() Revert order of entries in error list from last->first to first->last

Variables

- CONFERR * _cferr = NULL

  Library Internal: Error List Pointer.
- int _errcnt = 0

  Library Internal: Error List Counter.

17.6.1 Detailed Description

Error Codes, Functions and Structures.

- Error Codes and their mnemonic descriptions

Bug There are still errors without entry in error stack

- Library Internal Error Variables
  - Error Functions

17.6.2 Function Documentation

17.6.2.1 int cfgeterr (char * *string*, size_t *len*)

cfgeterr() Error Code and Message Inquiry Function

Error Code and Message Inquiry Function.

Parameters

string	Pointer to a string, to which the error message should be copied. Giving it a NULL pointer will omit message return
len	Size of string, if 0 CF_MAXERRSTR will be used

Returns

- 0 : if no error is available, everything is alright
- !=0 : error code of the next error in list

17.6.2.2 int cfputerr (int *ecode*, char * *string*, ...)

cfputerr() Init, exit or append to Error List

Init, exit or append to Error List.

Parameters

ecode	Error Code of the error that occurred
string	Error message format string. Giving it a NULL pointer results in an empty error message string
...	Arguments list according to format string

Returns

- <0 : A fatal error occurred (malloc failed)
- >0 : Number of errors in error list

17.6.2.3 void cfclearerr (void)

cfclearerr() Free all entries in error list

Free all entries in error list.

17.6.2.4 int cfreverr (void)

cfreverr() Revert order of entries in error list from last->first to first->last

Revert order of entries in error list from last->first to first->last.

Returns

- >=0 : Number of errors in error list
- <0 : Inconsistency with old error count, absolute value is new error counter

17.7 Advanced Features

Debugging and Utility Functions.

Defines

- #define CFD_CFDUMP 0

  Dump Mask Minimal.
- #define CFD_LIBHEAD 1

  Dump option CFLIB header.
- #define CFD_COLHEAD 2

  Dump option Column headers.
- #define CFD_SRCFLAGS 4

  Dump option Source flag description.
- #define CFD_FLAGS 8

  Dump option All Flags description.
- #define CFD_DEFAULT CFD_COLHEAD|CFD_SRCFLAGS

  Dump Mask Default.

Functions

- char * cfhomexp (char *name)

  cfhomexp() Expand ~ or ~user in parameter content
- int cfdinichk (CONFIG *set)

  *cfdinichk() Debugging Function (**experimental**)*
- int cfdump (FILE *fout)

  cfdump() Dump CFLIB DB content to fout

17.7.1 Detailed Description

Debugging and Utility Functions. Modes for cfdump()

17.7.2 Function Documentation

17.7.2.1 char * cfhomexp (char * *name*)

cfhomexp() Expand ~ or ~user in parameter content

Expand ~ or ~user in parameter content.

This function is used by default when reading Configuration Files and on initialization of parameters with the Special Option Flag CF_EXPHOME set.

- The environment is checked for the variables LOGNAME or USER, if no user name is given ("~/....")
  - The passwd file is searched for the users home directory, if possible
  - Otherwise the environment variable HOME is checked
  - If all that fails, ~ will be omitted, ~user will expand to "./user"

Parameters

name	Entry's name

Returns

- `NULL` : an error occurred:
  - No Configuration database Available.
  - Entry's Content is a NULL Pointer.
  - Memory allocation for Configuration Failed.
- `!=NULL` : String pointer to original or expanded filename

17.7.2.2 int cfdinichk (CONFIG * *set*)

cfdinichk() Debugging Function (**experimental**)

Debugging Function (**experimental**)

Initialize CFLIB DB using the default given by `set`, checking validity and plausibility of entries

Parameters

set	Pointer to initializing CONFIG-Array

Returns

- `0` : No error occurred
- `!=0` : An error occurred

Todo Make cfdinichk() work as reliable, complete tool with much more testing and Special Options Mask validation!

17.7.2.3 int cfdump (FILE * *fout*)

cfdump() Dump CFLIB DB content to `fout`

Dump CFLIB DB content to `fout`.

Debugging Function

Configuration options:
Set integer bitmask CFLIB variable **"CF_DUMPVERB"** to

```
@arg = @ref CFD_CFDUMP  : Minimal
@arg & @ref CFD_LIBHEAD : CFLIB header
@arg & @ref CFD_COLHEAD : Column headers
@arg & @ref CFD_SRCFLAGS : Source flag description
@arg = @ref CFD_DEFAULT : Default dump verbosity
```

Parameters

fout	Pointer to File opened for writing or `stdout/stderr/...`

Returns

- `>0` : Number of entries in CFLIB DB
- `<0` : An error occurred:
  - CFE_NCA : No configuration database available

17.8 Information Retrieval

These functions and macros read entries from an initialized CFLIB database.

Defines

- #define cfget(a) cfgetent(a,0)

  Get value (content) of named entry.
- #define cfgetstr(a) ((char *)cfgetent(a,CF_STR))

  Inquire CFLIB DB for String in content of named entry.
- #define cfgetnum(a) (*(int *)cfgetent(a,CF_INT))

  Inquire CFLIB DB for Integer value in content of named entry.
- #define cfgetreal(a) (*(float *)cfgetent(a,CF_REAL))

  Inquire CFLIB DB for Float (Real) value in content of named entry.
- #define cfgetdouble(a) (*(double *)cfgetent(a,CF_DOUBLE))

  Inquire CFLIB DB for Double value in content of named entry.
- #define cfgetflag(a) (*(int *)cfgetent(a,CF_FLAG))

  Inquire CFLIB DB for Flag value in content of named entry.
- #define cfflaginq(a, b) (*(int *)cfgetent(a,CF_FLGINQ|(31&b)))

  Inquire CFLIB DB for Bit set in entry's Special Options Flag CONFIG::flag.
- #define cfgetsrc(a) (*(int *)cfgetent(a,CF_SRC))

  Inquire CFLIB DB for Source of named entry's content.
- #define cfgetres() ((char *)cfgetent("",CF_RESID))

  Get next Residual Command Line Argument from CFLIB DB.
- #define cfgetcpr() "CFLIB (c) 1994-2012 Stefan Habermehl"

  Get CFLIB Copyright Notice.

Functions

- void * cfgetent (char *name, CFFLAGTYP typ)

  cfgetent() Inquire configuration database for content of entry name
- int cfgetvers (void)

  cfgetvers() Get Library Version/Patchlevel
- char * cfgetsubvers (void)

  cfgetsubvers() Get Library Subversion Details
- char * cfgetusg (void)

  cfgetusg() Get Usage Message for (Terminal) Output

17.8.1 Detailed Description

These functions and macros read entries from an initialized CFLIB database. The exact name of the required parameter must be given as argument, where indicated.

17.8.2 Define Documentation

17.8.2.1 #define cfget(*a*) cfgetent(a,0)

Get value (content) of named entry.

Parameters

a	Entry's name

Returns

Depending on configured type, see cfgetent() and Macro Definitions!

17.8.2.2 #define cfgetstr(a) ((char *)cfgetent(a,CF_STR))

Inquire CFLIB DB for String in content of named entry.

Parameters

a	Entry's name

Returns

- NULL : An error occurred:
  - No configuration database available
  - No entry of this name available
  - Content is really NULL, Check that with cfgetflag() !!
- Any other : Pointer to string in content of entry name

17.8.2.3 #define cfgetnum(a) (*(int *)cfgetent(a,CF_INT))

Inquire CFLIB DB for Integer value in content of named entry.

Parameters

a	Entry's name

Returns

- CFE_INF : Integer inquiry failed because of
  - No configuration database available
  - No entry of this name available
  - Content doesn't begin with digit
  - Content is a NULL pointer
  - Content is really CFE_INF, Check that with cfgetstr() !!
- Any other : Integer value for named entry

17.8.2.4 #define cfgetreal(a) (*(float *)cfgetent(a,CF_REAL))

Inquire CFLIB DB for Float (Real) value in content of named entry.

Parameters

a	Entry's name

Returns

- CFE_RNF : Real/float value inquiry failed because of
  - No configuration database available
  - No entry of this name available
  - Content doesn't begin with digit or signum (+/-)
  - Content is a NULL pointer
  - Content is really CFE_RNF, Check that with cfgetstr() !!
- Any other : Float value for named entry

17.8.2.5 #define cfgetflag(a) (*(int *)cfgetent(a,CF_FLAG))

Inquire CFLIB DB for Flag value in content of named entry.

Parameters

a	Entry's name

Returns

- `TRUE` (1) : Flag is set
- `FALSE` (0) : Flag is not set
- `<0` : An error occurred:
  - CFE_NCA : No configuration database available
  - CFE_NEA : No entry of this name available
  - CFE_ECP : Entry's content is a `NULL` pointer

17.8.2.6 #define cfflaginq(a, b) (*(int *)cfgetent(a,CF_FLGINQ|(31&b)))

Inquire CFLIB DB for Bit set in entry's Special Options Flag CONFIG::flag.

Inquire CFLIB DB for Bit set in entry's Special Options Fag CONFIG::flag.

Debugging Function

Parameters

a	Entry's name
b	Bit Offset, `0<=b<=31`

Returns

- `TRUE` (`1`) : Flag is set
- `FALSE` (`0`) : Flag is not set
- `<0` : An error occurred:
  - CFE_NCA : No Configuration database Available.
  - CFE_NEA : No Entry with that name Available.

17.8.2.7 #define cfgetsrc(a) (*(int *)cfgetent(a,CF_SRC))

Inquire CFLIB DB for Source of named entry's content.

Parameters

a	Entry's name

Returns

- `0-6` : Source of entry's content:
  - `0` : cfputstr() call
  - `1` : Command line / Arguments
  - `2` : Environment
  - `3` : Private Configuration File
  - `4` : System Configuration File
  - `5` : Default setting
  - `6` : Interactive input (query)

- <0 : An error occurred:
  - CFE_NCA : No configuration database available
  - CFE_NEA : No entry of this name available
  - CFE_NSS : No source set (should not happen!)

17.8.2.8 #define cfgetres( ) ((char *)cfgetent("",CF_RESID))

Get next Residual Command Line Argument from CFLIB DB.

Returns

- NULL : An error occurred:
  - No Configuration database Available.
  - No more residual arguments available
- Any other : Pointer to string content

See also

Residual Items

17.8.2.9 #define cfgetcpr( ) "CFLIB (c) 1994-2012 Stefan Habermehl"

Get CFLIB Copyright Notice.

Get Copyright Notice.

Returns

Pointer to Copyright Message String

17.8.3 Function Documentation

17.8.3.1 void * cfgetent (char * *name*, CFFLAGTYP *typ*)

cfgetent() Inquire configuration database for content of entry name

Library internal function, use appropriate Macro functions!

Inquire configuration database for content of entry name

Parameters

name	Entry's name
typ	Expected/required type of content: • 0 : Get type from entry's flag • CF_INT : Integer • CF_REAL : Real • CF_FLAG : Flag • CF_STR : String • CF_SRC : Source • CF_FLGINQ : Bit in entry flag (bit no. in lowest bytes) • CF_RESID : Residual argument • Anything else : String

Returns

- for CF_STR, CF_RESID or default:
- `NULL` : An error occurred or nothing available:
  - No configuration database available
  - No entry of this name available
  - The entry's content is really `NULL`, Check that with cfgetflag() !!
  - No more residual argument (for CF_RESID)
- any other pointer to string in content of entry name
- for others: Pointer to return values of the corresponding macro function

17.8.3.2 int cfgetvers (void)

cfgetvers() Get Library Version/Patchlevel

Get Library Version/Patchlevel.

Returns

- `>0` : Libary Patchlevel
- `<=0` : Error

17.8.3.3 char * cfgetsubvers (void)

cfgetsubvers() Get Library Subversion Details

Get Library Subversion Details.

The Patchlevel returned by this function should match Patchlevel in the public include file cf.h

Returns

Libary Patchlevel and Subversion (Source Revision marked by library internal header file *cflib.h*.

The return value is "burned" into the library executable and looks like:

```
CFLIB PL 20 $LastChangedRevision: 65 $
```

17.8.3.4 char * cfgetusg (void)

cfgetusg() Get Usage Message for (Terminal) Output

Get Usage Message for (Terminal) Output.

There are two flavours:

1. Let CFLIB do the job: Usage message is generated based on settings for commandline parsing and Special Options Mask found in the database

1. Deliver your own Usage Message: Just set the CF_USAGE Flag in the Special Options Mask of one parameter in the database to get a custom usage message (from hardcoded default, configuration file or environment). The delivered custom message string is taken as a format string for the printf() function: Use "%s" in the message string to have the program name inserted that CFLIB got from the default parameter **"CF_PRGNAME"** which by default is set to the name of the running program from the commandline at startup.

Returns

- Pointer to usage string
- `NULL` : An error occurred:
  - No Configuration database Available.
  - malloc() for usage string failed

See also

CF_MAXUSAGE : Maximum string length for usage string.
CF_USG_DEFCOLS : Default terminal width for usage string.

17.9 Setting and Saving the Configuration

Set/Update Parameter Values or Save a Configuration File.

Defines

- #define cfput(a, b) cfputstr(a,(char ∗)b)

  Update or Add a Parameter.

Functions

- int cfnosave (char ∗name, const char ∗onoff)

  cfnosave() Alter or query the CF_NOSAVE Flag of Parameter name
- int cfputstr (char ∗name, char ∗content)

  cfputstr() Update or Add Parameter name with string content
- int cfputtime (CFFLAGTYP td)

  cfputtime() Set all Time and/or Date entries in CFLIB DB to now or today
- int cfsave (char ∗fname, const char ∗savemode)

  cfsave() Write configuration data to a Configuration File or `stdout`

17.9.1 Detailed Description

Set/Update Parameter Values or Save a Configuration File.

17.9.2 Define Documentation

17.9.2.1 #define cfput(*a, b*) cfputstr(a,(char ∗)b)

Update or Add a Parameter.

Update or Add Parameter (Utility Function Macro)

Parameters

a	Parameter Name
b	New Content (Type casted to expected Char Pointer)

Returns

int cfputstr()

17.9.3 Function Documentation

17.9.3.1 int cfnosave (char ∗ *name,* const char ∗ *onoff*)

cfnosave() Alter or query the CF_NOSAVE Flag of Parameter *name*

Alter or query the CF_NOSAVE Flag of Parameter *name*.

When the configuration database is saved to a configuration file, the function cfsave() will exclude all items with the CF_NOSAVE flag set from the output.

The CF_NOSAVE flag can be set in the Special Option Mask CONFIG::flag for every entry in the Configuration Initializer given to cfinit() or cfstart() or later be set with this function for parameters in the current configuration database _conf

Residual Items will have the CF_NOSAVE flag set by default.

Parameters

name	Entry's name • `""` : All entries • `NULL` : All hardcoded entries
onoff	• CF_FLAG_ON : Set Flag • CF_FLAG_OFF : Delete Flag • `"i"` : Inquire Flag

Returns

- `0` : Entry updated successfully / Flag is OFF (for `"i"`)
- `1` : Flag is ON (for `"i"`)
- `!=0` or `1` : An Error occurred:
  - CFE_NCA : No Configuration database Available.
  - CFE_NEA : No Entry with that name Available.

17.9.3.2 int cfputstr (char * *name*, char * *content*)

cfputstr() Update or Add Parameter *name* with string *content*

Update or Add Parameter *name* with string *content*.

Parameters

name	Parameter Name
content	New (String) Content

Returns

- CFE_NEP : New Entry successfully Put into DB.
- `0` : Entry updated successfully
- `<0` : An error occurred:
  - CFE_NCA : No Configuration database Available.
  - CFE_MCF : Memory allocation for Configuration Failed.

17.9.3.3 int cfputtime (CFFLAGTYP *td*)

cfputtime() Set all Time and/or Date entries in CFLIB DB to *now* or *today*

Set all Time and/or Date entries in CFLIB DB to *now* or *today*.

Parameters

td	Target Selection Mask: • CF_TIME : Set Time • CF_DATE : Set Date • CF_TD : Set Time and Date

Returns

- >0 : Number of entries updated successfully
- <=0 : An error occurred:
  - CFE_NCA : No Configuration database Available.
  - CFE_MCF : Memory allocation for Configuration Failed.
  - CFE_NEA : No Entry with that name Available.

17.9.3.4 int cfsave (char * *file*, const char * *savemode*)

cfsave() Write configuration data to a Configuration File or stdout

Write configuration data to a Configuration File or stdout.

Entries with the CF_NOSAVE flag will be excluded from the output. Use cfnosave() to inquire or alter that flag for an entry.

Parameters

file	• <*string*> : Name of regular file to be opened • "" : Write configuration to stdout • NULL : Private configuration file will be overwritten or created, if an appropriate entry exists
savemode	File open mode: • "w" : Overwrite • "a" : Append

Returns

- 0 : Configuration has been saved to file successfully
- !=0 : An error occurred:
  - CFE_NCA :No Configuration database Available.
  - CFE_IFP :Invalid Filename entry for Private setfile.
  - CFE_WAE :Write Access Error.

18 Data Structure Documentation

18.1 CONFERR Struct Reference

Library Internal: Error List Item.

Data Fields

- struct _cfe * next
  Next Error Pointer.
- int errcode
  Numeric Error Code.
- char errstr [CF_MAXERRSTR]
  Error Message String of maximum length CF_MAXERRSTR.

18.1.1 Detailed Description

Library Internal: Error List Item.

18.2 CONFIG Struct Reference

CFLIB Configuration Database Entry.

Data Fields

- char * name
  Parameter Name
- char * inhalt
  Parameter Content, see Parameter Default Value.
- char option
  Commandline Option for Parameter
- CFFLAGTYP flag
  Special Options Mask

18.2.1 Detailed Description

CFLIB Configuration Database Entry.

19 File Documentation

19.1 include/cf.h File Reference

C Header File for CFLIB Flexible Configuration Library.

Data Structures

- struct CONFIG
  CFLIB Configuration Database Entry.

Defines

- #define Patchlevel "21"

  CFLIB Identification.
- #define MAXCONF 4096

  Maximum number of entries in configuration database.
- #define CF_MAXERRSTR 512

  Maximum string length for error message.
- #define CF_MAXLINE 20480

  Maximum string length for setfile and form parsing.
- #define CF_MAXQLINE 512

  Maximum string length for query.
- #define CF_MAXUSAGE 1024

  Maximum string length for usage string.
- #define CF_USG_DEFCOLS 80

  Default terminal width for usage string.
- #define CF_MAXTIMEBUF 256

  Buffer size for time and day.
- #define FALSE 0

  < TRUE, if not defined
- #define NULL (void *)(0L)

  NULL, if not defined.
- #define CF_FLAG_ON "\1"

  Flag is set.
- #define CF_FLAG_OFF ""

  Flag is not set.
- #define CF_NO_OPTION ' '

  Option is not set.
- #define TABLEN 8

  TAB length.
- #define CFP_PUT 0

  Function Call or Automatic Initialization.
- #define CFP_ARG 1

  Commandline.
- #define CFP_ENV 2

  Environment.
- #define CFP_PRIV 3

  Private Configuration File.
- #define CFP_SYS 4

  System Configuration File.
- #define CFP_DEF 5

  Built-in Default.
- #define CFP_QRY 6

  Standard Input Channel.
- #define CFP_RESERVED 7

  Reserved for Subprojects.
- #define CF_DEF_VARDELIM "$()"

  Default variable delimiter for cfform()
- #define CF_MAXINC 8

  Maximum number of nested includes for cfform()
- #define CF_BACKBUFLEN 102400

File copy buffer for BackupFile()

- #define CF_LAST 0x0001

  Last entry in settings array.
- #define CF_PATH 0x0002

  Search Path (for FindFile feature)
- #define CF_SETFILE 0x0004

  Entry is Private Configuration File.
- #define CF_SYS_SETFILE 0x0008

  Entry is System Configuration File.
- #define CF_SECTION 0x0010

  Section in Configuration File.
- #define CF_PRGNAME 0x0020

  Running Program's Name from commandline.
- #define CF_TIME 0x0040

  Time string.
- #define CF_DATE 0x0080

  Date string.
- #define CF_SET_PUT 0x0100

  Source: Function Call.
- #define CF_SET_ARG 0x0200

  Source: Commandline Argument.
- #define CF_SET_ENV 0x0400

  Source: Environment Variable.
- #define CF_SET_PRIV 0x0800

  Source: Private Configuration File.
- #define CF_SET_SYS 0x1000

  Source: System Configuration File.
- #define CF_SET_DEF 0x2000

  Source: Built-in Default.
- #define CF_SET_QRY 0x4000

  Source: Interactive Terminal Input.
- #define CF_NO_OPT_ARG 0x10000

  Commandline argument not following an option.
- #define CF_CONCAT 0x20000

  Argument is concatenated to option.
- #define CF_IGN_ENV 0x40000

  Do not check environment for variable.
- #define CF_QUERY 0x80000

  Ask the user for unresolved item after configuration parsing.
- #define CF_STR 0x100000

  Entry is String.
- #define CF_INT 0x200000

  Entry is Integer.
- #define CF_FLAG 0x400000

  Entry is Flag.
- #define CF_REAL 0x800000

  Entry is Float.
- #define CF_FINDFILE 0x1000000

  Entry is filename to be searched for in the path.
- #define CF_MUST 0x2000000

  Entry may not be empty (`NULL` or "")

- #define CF_RESID 0x4000000

  Residual/additional entry from commandline/setfile/cfput.
- #define CF_USAGE 0x8000000

  Usage Message format string.
- #define CF_EXPHOME 0x10000000

  Expand Home Directory.
- #define CF_MALLOC 0x20000000

  Space for entry's content was mallocated.
- #define CF_FORCED 0x40000000

  Setting has been forced (already)
- #define CF_NOSAVE 0x80000000

  Don't include in savefile / mark entry.
- #define CF_SRC (CF_INT|CF_FLAG)

  Type for source/origin inquiry.
- #define CF_FLGINQ (CF_STR|CF_FLAG)

  Type for options mask inquiry.
- #define CF_DOUBLE (CF_REAL|CF_FLAG)

  Type for double inquiry.
- #define CF_TD (CF_DATE|CF_TIME)

  Date or Time entry.
- #define CFE_INIT 0

  INITialize error input.
- #define CFE_OK 0

  No error / everything OKay.
- #define CFE_NEP 1

  New Entry successfully Put into DB.
- #define CFE_EXIT 1

  Finish error input.
- #define CFE_ORA 20

  Option Requires an Argument.
- #define CFE_UKO 25

  UnKnown Option.
- #define CFE_FNF 30

  File Not Found, read access error.
- #define CFE_NSE 40

  No Section specifier End found, missing "]".
- #define CFE_NSC 50

  No private Setfile Configured.
- #define CFE_WAE 60

  Write Access Error.
- #define CFE_IFP 61

  Invalid Filename entry for Private setfile.
- #define CFE_EWN 70

  Entry Without Name.
- #define CFE_ICF 80

  Invalid Combination of Flags.
- #define CFE_EWC 90

  Entry Without Content.
- #define CFE_UOS 100

  Unlikely Option Specifier.
- #define CFE_IFC 110

Invalid Flag Combination.

- #define CFE_NLE 120

  No Last Entry flag found.
- #define CFE_TIN 130

  Error reTurn from stdIN query.
- #define CFE_EFE 140

  Empty string in content for Filename Entry.
- #define CFE_USG 200

  Entries missing: USaGge advice.
- #define CFE_URI 210

  UnResolved Item (CF_MUST was set!)
- #define CFE_FBF -500

  File Backup Failed.
- #define CFE_BMF -510

  Backup: Memory allocation Failed.
- #define CFE_BOF 520

  Backup: Open source file Failed.
- #define CFE_BBF 530

  Backup: open target Backup file Failed.
- #define CFE_BRF 540

  Backup: Rename Failed.
- #define CFE_BWF 550

  Backup: Write Failed.
- #define CFE_NCA -10

  No Configuration database Available.
- #define CFE_NEA -20

  No Entry with that name Available.
- #define CFE_NSS -30

  No Source/origin is Set.
- #define CFE_ECP -40

  Entry's Content is a NULL Pointer.
- #define CFE_MEF -100

  Memory allocation in Error routine Failed.
- #define CFE_MCF -200

  Memory allocation for Configuration Failed.
- #define CFE_INF -9999

  Integer iNquiry Failed (?!)
- #define CFE_RNF -999.999

  Real/float iNquiry Failed (?!)
- #define CFS_NOT 0

  Error Response Modes for cfstart()
- #define CFS_ALL 1

  Start Mode: All error messages.
- #define CFS_NEG 2

  Start Mode: Only severe errors.
- #define CFS_USG 3

  Start Mode: Usage message if error was negative, error output like CFS_NEG.
- #define CFS_DEBUG 4

  Start Mode: Output like CFS_USG plus raw dump of configuration.
- #define CFD_CFDUMP 0

  Dump Mask Minimal.

- #define CFD_LIBHEAD 1

  Dump option CFLIB header.
- #define CFD_COLHEAD 2

  Dump option Column headers.
- #define CFD_SRCFLAGS 4

  Dump option Source flag description.
- #define CFD_FLAGS 8

  Dump option All Flags description.
- #define CFD_DEFAULT CFD_COLHEAD|CFD_SRCFLAGS

  Dump Mask Default.
- #define cfget(a) cfgetent(a,0)

  Get value (content) of named entry.
- #define cfgetstr(a) ((char *)cfgetent(a,CF_STR))

  Inquire CFLIB DB for String in content of named entry.
- #define cfgetnum(a) (*(int *)cfgetent(a,CF_INT))

  Inquire CFLIB DB for Integer value in content of named entry.
- #define cfgetreal(a) (*(float *)cfgetent(a,CF_REAL))

  Inquire CFLIB DB for Float (Real) value in content of named entry.
- #define cfgetdouble(a) (*(double *)cfgetent(a,CF_DOUBLE))

  Inquire CFLIB DB for Double value in content of named entry.
- #define cfgetflag(a) (*(int *)cfgetent(a,CF_FLAG))

  Inquire CFLIB DB for Flag value in content of named entry.
- #define cfflaginq(a, b) (*(int *)cfgetent(a,CF_FLGINQ|(31&b)))

  Inquire CFLIB DB for Bit set in entry's Special Options Flag CONFIG::flag.
- #define cfgetsrc(a) (*(int *)cfgetent(a,CF_SRC))

  Inquire CFLIB DB for Source of named entry's content.
- #define cfgetres() ((char *)cfgetent("",CF_RESID))

  Get next Residual Command Line Argument from CFLIB DB.
- #define cfgetcpr() "CFLIB (c) 1994-2012 Stefan Habermehl"

  Get CFLIB Copyright Notice.
- #define cfput(a, b) cfputstr(a,(char *)b)

  Update or Add a Parameter.
- #define DelFlag(a, b) a&=(~b)

  Delete bits of Mask `b` from Mask `a`.
- #define SetFlag(a, b) a|=b

  Set bits of Mask `b` in Mask `a`.
- #define __CF_H__

  Marker: cf.h has been included.

Typedefs

- typedef unsigned long CFFLAGTYP

  Special Options Mask Type.

Functions

- int cfinit (CONFIG *set, int argc, char **argv)

  cfinit() Initialize Configuration Database
- void cfexit (void)

  cfexit() Free allocated memory and reset configuration database and error stack
- int cfstart (CONFIG *set, int ac, char **av, char *help, int mode)

  cfstart() Start Configuration Database (with error reporting and usage message)
- void * cfgetent (char *name, CFFLAGTYP typ)

  cfgetent() Inquire configuration database for content of entry name
- int cfgetvers (void)

  cfgetvers() Get Library Version/Patchlevel
- char * cfgetsubvers (void)

  cfgetsubvers() Get Library Subversion Details
- char * cfgetusg (void)

  cfgetusg() Get Usage Message for (Terminal) Output
- int cfgeterr (char *string, size_t len)

  cfgeterr() Error Code and Message Inquiry Function
- int cfputerr (int ecode, char *string,...)

  cfputerr() Init, exit or append to Error List
- void cfclearerr (void)

  cfclearerr() Free all entries in error list
- int cfreverr (void)

  cfreverr() Revert order of entries in error list from last->first to first->last
- char * cfhomexp (char *name)

  cfhomexp() Expand ~ or ~`user` in parameter content
- int cfdinichk (CONFIG *set)

  *cfdinichk() Debugging Function (**experimental**)*
- int cfdump (FILE *fout)

  cfdump() Dump CFLIB DB content to `fout`
- int cfnosave (char *name, const char *onoff)

  cfnosave() Alter or query the CF_NOSAVE Flag of Parameter name
- int cfputstr (char *name, char *content)

  cfputstr() Update or Add Parameter name with string content
- int cfputtime (CFFLAGTYP td)

  cfputtime() Set all Time and/or Date entries in CFLIB DB to now or today
- int cfsave (char *fname, const char *savemode)

  cfsave() Write configuration data to a Configuration File or `stdout`
- int BackupFile (const char *file, char *modus)

  BackupFile() Copy or Rename File "`file`" to Backup File "`file~`" or "`file.bak`"
- int cfform (char *infile, char *outfile, char *vardelim, int mode)

  cfform() Process a Template from file or `stdin` and write generated Report to File or `stdout`
- void RemoveCR (char *string)

  Strip Carriage Return at end of string (after fgets) by introducing zero byte at CR position.
- void RemoveTrailSpace (char *string)

  Strip whitespaces at end of string by introducing zero byte after last non-whitespace character.
- char * EatWhiteSpace (char *string)

  EatWhiteSpace() Set pointer to next non-whitespace-character in string.
- int IsATerminal (FILE *fp)

  IsATerminal() Test whether stream is a terminal
- char * FindFile (const char *fname, const char *fpath, const char *const *fext)

  FindFile() Find a File in Path trying Extensions

19.1.1 Detailed Description

C Header File for CFLIB Flexible Configuration Library. Public Functions and Definitions

Note

Include this file in the source code and link with the library executable, usually referring to *libcf.a* by calling "`(g)cc -lcf ...`"

Version

SVN: $Id: cf.h 67 2013-02-27 11:24:49Z stefan_x $

Author

Stefan Habermehl `stefan.habermehl@mcff.de`

Copyright:

(c) 1994,1995,1996,1997,1998,2006,2007,2008,2009,2013 Stefan Habermehl

License:

`http://www.gnu.org/licenses` GNU Lesser General Public License version 3.0 (LGPLv3)

Index