simple_options_file_parser

version 1.0.01

Generated by Doxygen 1.10.0

Chapter 1

The simple_options_file_parser application

The simple_options_file_parser library implements an options text file parser. Options are input by means of sentences adhering to a very simple syntax, assigning a value to a label identifying the option to set:

LABEL = some_value

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 simple_options_file_parser Class Reference

Simple parser for label / value option files.

`#include <simple_options_file_parser.hpp>`

Public Member Functions

- int get_option_double (const string &option_label, double &option_value)

  Retrieve the value of a label as a double value.
- int get_option_double_array (const string &option_label, vector< double > &option_value)

  Retrieve the value of a label as a list of double values.
- int get_option_int (const string &option_label, int &option_value)

  Retrieve the value of a label as an int value.
- int get_option_int_array (const string &option_label, vector< int > &option_value)

  Retrieve the value of a label as a list of int values.
- int get_option_string (const string &option_label, string &option_value)

  Retrieve the value of a label as a string.
- int get_option_string_array (const string &option_label, vector< string > &option_value)

  Retrieve the words in a string as separate elements.
- int parse (const string &options_file, int &error_line)

  Load and interpret a simple options file.
- **simple_options_file_parser** (void)

  Default constructor.
- ~**simple_options_file_parser** (void)

  Destructor.

Protected Member Functions

- bool read_data_line (ifstream &file, string &line, int &last_line_read, int &line_data)

  Retrieve the next line with actual data to process.
- string reduce (const string &str, const string &fill=" ", const string &whitespace=" \t")

  Removes leading and trailing whitespace from a string. Moreover, internal whitespace is purged (leaving a single occurence between non-whitespace characters) and replaced by some string selectable by the user.
- int split (const string &txt, vector< std::string > &strs, char ch=' ')

  Splits a string at the points specified by some configurable character.
- string trim (const string &str, const string &whitespace=" \t")

  Remove leading and trailing whitespace from a string. It is possible to define what is "whitespace".

Protected Attributes

- map< string, string > **options_**

  The map holding the label / value pairs loaded from the input file.

4.1.1 Detailed Description

Simple parser for label / value option files.

This class implements a very simple option files parser.

These option files must be text files. Values are input by means of unique labels. The specific format to write a label and its corresponding value is:

label = value

All the elements in the previous example (label, equal sign, value) must be written in the same line.

Multiple labels must be written in consecutive lines. For instance:

label1 = value1 label2 = value2

Values may be of type string, integer or double. Additionally, list of integers and doubles are allowed. These lists must separate the different integer o double values by means of spaces. For instance:

my_integer_list = 1 2 3 4 my_double_list = 1.2 3.4 5.6 7.8 9.0

When the values are strings, the leading a trailing spaces are removed. Internal whitespace, whowever is fully preserved. Therefore, in the following example:

my_string_value = this is a test

the actual value processed by this class would be "this is a test". Note how the leading and trailing spaces have been removed, but the internal whitespace has not.

Labels are case sensitive. Therefore, label "my_value" is not equal to "MY_VALUE" nor to "mY_vAlUe".

Blank lines are ignored. It is possible to write comments; lines whose first character is "#" will be considered as comments. For example:

This is a comment.

To use this class,

- Declare / instantiate a simple_options_file_parser object.
- Call method parse() to load the options file into memory.
- Call the different get_xxx_xxx() methods to retrieve the value of the labels of interest, using the proper type (int, double, string, int_array or double_array) for the kind of value expected.

4.1.2 Member Function Documentation

4.1.2.1 get_option_double()

```
int simple_options_file_parser::get_option_double (
            const string & option_label,
            double & option_value )
```

Retrieve the value of a label as a double value.

Parameters

option_label	The label identifying the option sought.
option_value	The value sought. It is only meaningful when the method finishes correctly.

Returns

An error code:

- 0: Successful completion.
- 1: Label not found. The label in "option_label" does not exist in the options file.
- 2: The value related to the label can not be interpreted as a double value.

This method tries to convert the string representing the value to double format. If such string is not a double, an error code will be returned.

Moreover, an error code is also returned if the label sought is not found. In this case, it is important to remember that labels are case sensitive.

4.1.2.2 get_option_double_array()

```
int simple_options_file_parser::get_option_double_array (
            const string & option_label,
            vector< double > & option_value )
```

Retrieve the value of a label as a list of double values.

Parameters

option_label	The label identifying the option sought.
option_value	The array of values sought. It is only meaningful when the method finishes correctly.

Returns

An error code:

- 0: Successful completion.
- 1: Label not found. The label in "option_label" does not exist in the options file.
- 2: One or more values in the expected list of doubles can not be interpreted as a double value.

This method tries to convert the string representing a list of double values separated by spaces to its array of doubles equivalent. If one or more of the components in the string cannot be interpreted as a double value, an error code will be returned. On the contrary, when such conversion works, the output option_value array will contain as many values as doubles found in the list. The number of values may be obtained by calling method size() of option_value.

Moreover, an error code is also returned if the label sought is not found. In this case, it is important to remember that labels are case sensitive.

4.1.2.3 get_option_int()

```
int simple_options_file_parser::get_option_int (
            const string & option_label,
            int & option_value )
```

Retrieve the value of a label as an int value.

Parameters

option_label	The label identifying the option sought.
option_value	The value sought. It is only meaningful when the method finishes correctly.

Returns

An error code:

- 0: Successful completion.
- 1: Label not found. The label in "option_label" does not exist in the options file.
- 2: The value related to the label can not be interpreted as an int value.

This method tries to convert the string representing the value to int format. If such string is not an int, an error code will be returned.

Moreover, an error code is also returned if the label sought is not found. In this case, it is important to remember that labels are case sensitive.

4.1.2.4 get_option_int_array()

```
int simple_options_file_parser::get_option_int_array (
            const string & option_label,
            vector< int > & option_value )
```

Retrieve the value of a label as a list of int values.

Parameters

option_label	The label identifying the option sought.
option_value	The array of values sought. It is only meaningful when the method finishes correctly.

Returns

An error code:

- 0: Successful completion.
- 1: Label not found. The label in "option_label" does not exist in the options file.
- 2: One or more values in the expected list of int can not be interpreted as an int value.

This method tries to convert the string representing a list of int values separated by spaces to its array of int equivalent. If one or more of the components in the string cannot be interpreted as an int value, an error code

will be returned. On the contrary, when such conversion works, the output option_value array will contain as many values as int found in the list. The number of values may be obtained by calling method size() of option_value.

Moreover, an error code is also returned if the label sought is not found. In this case, it is important to remember that labels are case sensitive.

4.1.2.5 get_option_string()

```
int simple_options_file_parser::get_option_string (
            const string & option_label,
            string & option_value )
```

Retrieve the value of a label as a string.

Parameters

option_label	The label identifying the option sought.
option_value	The value sought. It is only meaningful when the method finishes correctly.

Returns

An error code:

- 0: Successful completion.
- 1: Label not found. The label in "option_label" does not exist in the options file.

This method never tries to convert the text value in the options file, so no errors must be expected on this side.

However, an error code is also returned if the label sought is not found. In this case, it is important to remember that labels are case sensitive.

4.1.2.6 get_option_string_array()

```
int simple_options_file_parser::get_option_string_array (
            const string & option_label,
            vector< string > & option_value )
```

Retrieve the words in a string as separate elements.

Parameters

option_label	The label identifying the option sought.
option_value	The array of values sought. It is only meaningful when the method finishes correctly.

Returns

An error code:

- 0: Successful completion.
- 1: Label not found. The label in "option_label" does not exist in the options file.

This method splits all the words in a string into a list of strings. That is, the string "This is a string" is returned as the list of separate values "This", "is", "a" and "string". Leading and trailing Whitespace, as well as that found between words is completely removed.

An error code is returned if the label sought is not found. In this case, it is important to remember that labels are case sensitive.

4.1.2.7 parse()

```
int simple_options_file_parser::parse (
            const string & options_file,
            int & error_line )
```

Load and interpret a simple options file.

Parameters

options_file	The name of the file containing the options to load.
error_line	If an error is detected while loading the options file, this parameter will contain the line number where the aforementioned error was detected.

Returns

An error code:

- 0: Successful completion.
- 1: The name of the options file is empty (parameter options_file; no error line is set).
- 2: Error opening the input options file (no error line is set).
- 3: Missing equal sign ("=") when parsing a label / value pair.
- 4: Empty label or value found while parsing a label / value pair.

4.1.2.8 read_data_line()

```
bool simple_options_file_parser::read_data_line (
            ifstream & file,
            string & line,
            int & last_line_read,
            int & line_data )  [protected]
```

Retrieve the next line with actual data to process.

Parameters

file	The file to read the lines from.
line	The line of data retrieved.
last_line_read	Number of the last line read.
line_data	Number of the line where a line of data starts (may differ from line_read if comments are found). Used to report the line when syntax errors are detected when processing a label / value pair.

Returns

True if a new line has been retrieved, false if the end-of-file condition is arised.

This methods reads complete lines from the input file, skipping those that begin with a comment sign (#).

Normally, data lines (those that contain an statement label = value) take only that line, but in the case of lists of values (as in label = value1 value2 ... valueN) these may become too long so lines are split using the backslash character.

When such event occurs, lines are read consecutively until no backslash is found. The set of lines read is then merged into a single one removing the backlashes, as if it would have been written as a single line in the original input file.

4.1.2.9 reduce()

```
string simple_options_file_parser::reduce (
            const string & str,
            const string & fill = " ",
            const string & whitespace = " \t" )  [protected]
```

Removes leading and trailing whitespace from a string. Moreover, internal whitespace is purged (leaving a single occurence between non-whitespace characters) and replaced by some string selectable by the user.

Parameters

str	The string whose whitespace has to be removed, purged and replaced.
fill	The string to use to replace the internal whitespace once it's been purged.
whitespace	Set of characters to be considered as whitespace.

Examples: assuming that str is set to " abc d e f ", calling this method like this:

reduce(str, " ", " \t")

would yield this result: str = "abc d e f". This is the "normal" whay to use this method.

However, calling the method like this:

reduce(str, "***", " \t")

would make str look like this: "abc***d***e***f"

4.1.2.10 split()

```
int simple_options_file_parser::split (
            const string & txt,
            vector< std::string > & strs,
            char ch = ' ' )  [protected]
```

Splits a string at the points specified by some configurable character.

Parameters

txt	The string to be split.
strs	A vector of strings, containing the tokens resulting from the splitting process.
ch	The character used as delimiter to find the places where the input string has to be split.

Returns

The number of elements in output vector strs (that is, the number of tokens resulting from the splitting process). May be zero.

4.1.2.11 trim()

```
string simple_options_file_parser::trim (
            const string & str,
            const string & whitespace = " \t" )  [protected]
```

Remove leading and trailing whitespace from a string. It is possible to define what is "whitespace".

Parameters

str	The string whose leading and trailing whitespace has to be removed.
whitespace	Set of characters to be considered as whitespace.

Returns

A string equivalent to the input one, with leading and trailing whitespace removed.

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

- src/simple_options_file_parser.hpp
- src/simple_options_file_parser.cpp

Chapter 5

File Documentation

5.1 src/simple_options_file_parser.cpp File Reference

Implementation file for simple_options_file_parser.hpp.

```
#include "simple_options_file_parser.hpp"
```

5.1.1 Detailed Description

Implementation file for simple_options_file_parser.hpp.

5.2 src/simple_options_file_parser.hpp File Reference

Simple parser for label / value option files.

```
#include <string>
#include <map>
#include <vector>
#include <sstream>
#include <fstream>
```

Classes

- class simple_options_file_parser

  Simple parser for label / value option files.

5.2.1 Detailed Description

Simple parser for label / value option files.

5.3 simple_options_file_parser.hpp

Go to the documentation of this file.

```
00001
00005 #ifndef _SIMPLE_OPTIONS_FILE_PARSER_
00006 #define _SIMPLE_OPTIONS_FILE_PARSER_
00007
00008 #include <string>
00009 #include <map>
00010 #include <vector>
00011 #include <sstream>
00012 #include <fstream>
00013
00014 using namespace std;
00015
00017
00066 class simple_options_file_parser
00067 {
00068   public:
00069
00071
00091     int      get_option_double          (const string&        option_label,
00092                                          double&              option_value);
00093
00095
00120     int      get_option_double_array    (const string&        option_label,
00121                                          vector<double>&      option_value);
00122
00124
00144     int      get_option_int             (const string&        option_label,
00145                                          int&                 option_value);
00146
00148
00173     int      get_option_int_array       (const string&        option_label,
00174                                          vector<int>&         option_value);
00175
00177
00193     int      get_option_string          (const string&        option_label,
00194                                          string&              option_value);
00195
00197
00217     int      get_option_string_array    (const string&        option_label,
00218                                          vector<string>&      option_value);
00219
00221
00234     int      parse                      (const string&        options_file,
00235                                          int&                 error_line);
00236
00238
00239              simple_options_file_parser (void);
00240
00242
00243              ~simple_options_file_parser (void);
00244
00245   protected:
00246
00248
00274     bool     read_data_line             (ifstream&            file,
00275                                          string&              line,
00276                                          int&                 last_line_read,
00277                                          int&                 line_data);
00278
00283
00304     string   reduce                     (const string&        str,
00305                                          const string&        fill = " ",
00306                                          const string&        whitespace = " \t");
00307
00309
00320     int      split                      (const string&        txt,
00321                                          vector<std::string>& strs,
00322                                          char                 ch = ' ');
00323
00326
00332     string   trim                       (const string&        str,
00333                                          const string&        whitespace = " \t");
00334
00335   protected:
00336
00338
00339     map<string, string> options_;
00340 };
00341
00342 #endif // _SIMPLE_OPTIONS_FILE_PARSER_
```