

spec\_tools

Generated by Doxygen 1.8.13

## Contents

<b>1</b>	<b>Todo List</b>	<b>2</b>
<b>2</b>	<b>Class Documentation</b>	<b>2</b>
2.1	<a href="#">_csv&lt;_T&gt; Class Template Reference</a>	2
2.1.1	<a href="#">Detailed Description</a>	4
2.1.2	<a href="#">Constructor &amp; Destructor Documentation</a>	5
2.1.3	<a href="#">Member Function Documentation</a>	6
2.2	<a href="#">_io&lt;_T&gt; Class Template Reference</a>	19
2.2.1	<a href="#">Detailed Description</a>	20
2.2.2	<a href="#">Member Function Documentation</a>	20
2.3	<a href="#">_log Class Reference</a>	23
2.4	<a href="#">_marker&lt;_T&gt; Class Template Reference</a>	24
2.4.1	<a href="#">Detailed Description</a>	26
2.4.2	<a href="#">Member Function Documentation</a>	26
2.4.3	<a href="#">Member Data Documentation</a>	28
2.5	<a href="#">_msg Class Reference</a>	28
2.5.1	<a href="#">Detailed Description</a>	29
2.5.2	<a href="#">Member Function Documentation</a>	29
2.6	<a href="#">_op&lt;_T&gt; Class Template Reference</a>	30
2.6.1	<a href="#">Detailed Description</a>	30
2.6.2	<a href="#">Member Function Documentation</a>	31
2.7	<a href="#">_marker&lt;_T&gt;::Line Struct Reference</a>	33
2.7.1	<a href="#">Detailed Description</a>	34
2.8	<a href="#">_io&lt;_T&gt;::vec Struct Reference</a>	34
2.8.1	<a href="#">Detailed Description</a>	34

<b>3</b>	<b>File Documentation</b>	<b>35</b>
3.1	csv.h File Reference . . . . .	35
3.1.1	Detailed Description . . . . .	36
3.2	der_snr.cpp File Reference . . . . .	36
3.2.1	Detailed Description . . . . .	37
3.3	elemlist.cpp File Reference . . . . .	37
3.3.1	Detailed Description . . . . .	38
3.3.2	Macro Definition Documentation . . . . .	38
3.4	findncopy.cpp File Reference . . . . .	39
3.4.1	Detailed Description . . . . .	40
3.4.2	Macro Definition Documentation . . . . .	40
3.5	genrandspec.cpp File Reference . . . . .	40
3.5.1	Detailed Description . . . . .	41
3.5.2	Macro Definition Documentation . . . . .	42
3.6	log.h File Reference . . . . .	42
3.6.1	Detailed Description . . . . .	43
3.7	marker.cpp File Reference . . . . .	43
3.7.1	Detailed Description . . . . .	44
3.7.2	Macro Definition Documentation . . . . .	44
3.8	msg.h File Reference . . . . .	45
3.8.1	Detailed Description . . . . .	46
3.9	shift.cpp File Reference . . . . .	46
3.9.1	Detailed Description . . . . .	47
3.9.2	Macro Definition Documentation . . . . .	47
3.10	waverage.cpp File Reference . . . . .	48
3.10.1	Detailed Description . . . . .	48
	<b>Index</b>	<b>49</b>

## 1 Todo List

Member `_csv<_T>::set_separator` (const std::string &sSep)

Class `_marker<_T>`

`marker(const _marker<_T>&)`

## 2 Class Documentation

### 2.1 `_csv<_T>` Class Template Reference

This is the templated `_csv` class, initialized with double by default. STL parallel execution policy does not provide enhancements for simple operations.

```
#include <csv.h>
```

#### Public Types

- enum `eVerbose` { **QUIET**, **DEBUG** }
- Define verbosity values.*

#### Public Member Functions

- `_csv` ()  
*Default constructor without parameters. These parameters must be set after by methods. It will rise lot of errors if something is missing.*
- `_csv` (const std::string &sFilename, const char &cSep)  
*Constructor with two parameters such as the name of the working file and the separator character as usual with csv.*
- `_csv` (const std::string &sFilename, const std::string &sSep)
- `_csv` (const std::vector< std::vector<\_T> > &vvData)  
*Constructor fed with external data.*
- `_csv` (const std::vector< std::string > &vsHeader, const std::vector< std::vector<\_T> > &vvData)  
*Constructor fed with external header and data.*
- `_csv` (const std::vector< std::string > &vsHeader, const std::vector< std::vector<\_T> > &vvData, const char &cSep)  
*Constructor fed with external header and data.*
- bool `read` ()  
*Read the content of the file given to the constructor using boost. It detects the header and data consistency with digit sequence: {0123456789e+-, tab std::endl} and basic regex and dimension matching between header and data line. It is able to recover basic errors such as 'tab'==' '. The method put NaN in the grid if an unrecoverable error appends. Data will be store in private variables.*
- bool `show` () const  
*Show whole data, i.e. the header and data with no restriction on length or terminal size. It uses boost::format in order to correct spacing of number and strings.*
- bool `show` (int iLine\_stop) const

Show the header and data until "line\_stop" line. Print all columns with terminal end-of-line. It uses `boost::format` in order to correct spacing of number and strings.

- `bool write ()`  
Write on disk what data are store.
- `const std::vector<_T> select_line (int line) const`  
Select the line "line" in data.
- `const std::vector<_T> select_column (int iCol) const`  
Select the column "col" in data.
- `const std::vector< std::vector<_T> > select (int iLine_min, int iLine_max, int iCol_min, int iCol_max) const`  
Select a sub grid in data, i.e. trim data to the rectangular  $[i_{min}, i_{max}] \times [j_{min}, j_{max}]$ .
- `bool set_data (const std::vector< std::vector<_T> > &vvData)`  
Set data with a vector of a vector.
- `bool set_column (const std::vector<_T> &vCol, int iCol)`  
Set a column with a vector.
- `bool set_row (const std::vector<_T> &vRow, int iRow)`
- `bool set_header (const std::vector< std::string > &vsHeader)`  
Set the header: the first line containing column name.
- `bool set_filename (const std::string &sFilename)`  
Set the filename for output or input. The fstream do not care about extension...
- `bool set_filename_out (const std::string &sFilename)`  
Set the filename for output. The fstream do not care about extension...
- `bool set_separator (const char &cSep)`  
Set the csv separator. Usually: '\t', ',', ';', ';' ...
- `bool set_separator (const std::string &sSep)`  
Set the csv separator. Usually: '\t', ',', ';', ';' ...
- `void set_verbose (eVerbose evV)`  
Set the verbose mode for debug. It does not deactivate error raising.
- `const std::string get_filename () const`  
Get the filename.
- `const std::string get_filename_out () const`  
Get the output filename.
- `const char get_separator () const`  
Get the separator.
- `const size_t get_header_size () const`  
Get size of the header.
- `const size_t get_data_size_i () const`  
Get data line size.
- `const size_t get_data_size_j () const`  
Get data column size.
- `const std::vector< std::vector<_T> > & get_data () const`  
Get data and return it as a vector of vector.
- `const std::vector< std::string > & get_header () const`  
Get column names and return it in a vector.
- `bool empty () const`  
Check if data are empty, and the emptiness of the first line, i.e. `this->data[0]`.
- `bool check_dim ()`  
Check data dimension consistency, i.e. if all line dimensions are all equal.

- bool **genrandspec** (\_T TMin, \_T TMax, \_T TStep)
- bool **transform\_lin** (\_T TA, \_T TB, int iCol)
 

*Do  $Y=aX+b$  to the iCol-column.*
- bool **shift** (\_T TVal)
- bool **shift** (\_T TVal, int iCol)
- bool **apply\_max\_threshold** (\_T TVal)
 

*Delete i line from the grid where  $\text{data}[i][j] > \text{val}$ .*
- bool **apply\_min\_threshold** (\_T TVal)
 

*Delete i line from the grid where  $\text{data}[i][j] < \text{val}$ .*
- bool **apply\_max\_threshold** (\_T TVal, int iCol)
 

*Delete i line from the grid where  $\text{data}[i][j \neq \text{list}] > \text{val}$ .*
- bool **apply\_min\_threshold** (\_T TVal, int iCol)
 

*Delete i line from the grid where  $\text{data}[i][j \neq \text{list}] < \text{val}$ .*
- void **zeroize** ()
 

*Set to zero data. One should find this useful...*
- void **clear** ()
 

*Delete data and header.*
- **\_csv & operator=** (const **\_csv** &other) const
- bool **operator==** (const **\_csv** &other) const
- bool **operator!=** (const **\_csv** &other) const
- **\_csv & operator+** (const **\_csv** &other) const
 

*Sum with the 2nd column.*
- **\_csv & operator+** (const \_T &other) const
 

*Add a constant to the 2nd column.*
- **\_csv & operator-** (const **\_csv** &other) const
 

*Sum with the 2nd column.*
- **\_csv & operator-** (const \_T &other) const
 

*Subtract a constant to the 2nd column.*
- **\_csv & operator\*** (const **\_csv** &other) const
 

*Inner product with the 2nd column.*
- **\_csv & operator\*** (const \_T &other) const
 

*Multiply by a constant the 2nd column.*
- **\_csv & operator/** (const **\_csv** &other) const
 

*Divide element by element the two columns.*
- **\_csv & operator/** (const \_T &other) const
 

*Divide by a non zero constant the 2nd column.*

### 2.1.1 Detailed Description

```
template<typename _T = double>
class _csv<_T>
```

This is the templated **\_csv** class, initialized with double by default. STL parallel execution policy does not provide enhancements for simple operations.

## 2.1.2 Constructor &amp; Destructor Documentation

2.1.2.1 `_csv()` [1/6]

```
template<typename _T = double>
_csv<_T>::_csv ( )
```

Default constructor without parameters. These parameters must be set after by methods. It will rise lot of errors if something is missing.

Default constructor

2.1.2.2 `_csv()` [2/6]

```
template<typename _T = double>
_csv<_T>::_csv (
    const std::string & sFilename,
    const char & cSep ) [explicit]
```

Constructor with two parameters such as the name of the working file and the separator character as usual with csv.

Constructor

Parameters

<i>sFilename</i>	string Name of the input or output file with extension
<i>cSep</i>	char Separator char between column

2.1.2.3 `_csv()` [3/6]

```
template<typename _T = double>
_csv<_T>::_csv (
    const std::string & sFilename,
    const std::string & sSep ) [explicit]
```

Parameters

<i>sFilename</i>	string Name of the input or output file with extension
<i>sSep</i>	string Separator char between column

2.1.2.4 `_csv()` [4/6]

```
template<typename _T = double>
```

```
_csv< _T >::_csv (
    const std::vector< std::vector< _T > > & vvData ) [explicit]
```

Constructor fed with external data.

#### Parameters

<i>vvData</i>	the data
---------------	----------

#### 2.1.2.5 \_csv() [5/6]

```
template<typename _T = double>
_csv< _T >::_csv (
    const std::vector< std::string > & vsHeader,
    const std::vector< std::vector< _T > > & vvData ) [explicit]
```

Constructor fed with external header and data.

#### Parameters

<i>vsHeader</i>	The vector of column name
<i>vvData</i>	the data

#### 2.1.2.6 \_csv() [6/6]

```
template<typename _T = double>
_csv< _T >::_csv (
    const std::vector< std::string > & vsHeader,
    const std::vector< std::vector< _T > > & vvData,
    const char & cSep ) [explicit]
```

Constructor fed with external header and data.

#### Parameters

<i>vsHeader</i>	the vector of column name
<i>vvData</i>	the data
<i>cSep</i>	char Separator char between column

### 2.1.3 Member Function Documentation



2.1.3.1 `apply_max_threshold()` [1/2]

```
template<typename _T = double>
bool _csv<_T>::apply_max_threshold (
    _T TVal )
```

Delete  $i$  line from the grid where `data[i][j] > val`.

## Parameters

<i>TVal</i>	The max threshold
-------------	-------------------

## Returns

true if all seems OK

2.1.3.2 `apply_max_threshold()` [2/2]

```
template<typename _T = double>
bool _csv<_T>::apply_max_threshold (
    _T TVal,
    int iCol )
```

Delete  $i$  line from the grid where `data[i][j  $\neq$  list] > val`.

## Parameters

<i>TVal</i>	The max threshold
<i>iCol</i>	Select a column

## Returns

true if all seems OK

2.1.3.3 `apply_min_threshold()` [1/2]

```
template<typename _T = double>
bool _csv<_T>::apply_min_threshold (
    _T TVal )
```

Delete  $i$  line from the grid where `data[i][j] < val`.

**Parameters**

<i>TVal</i>	The min threshold
-------------	-------------------

**Returns**

true if all seems OK

**2.1.3.4 apply\_min\_threshold()** [2/2]

```
template<typename _T = double>
bool _csv< _T >::apply_min_threshold (
    _T TVal,
    int iCol )
```

Delete *i* line from the grid where **data**[*i*][*j* ≠ *list*] < *val*.

**Parameters**

<i>TVal</i>	The min threshold
<i>iCol</i>	Select a column

**Returns**

true if all seems OK

**2.1.3.5 check\_dim()**

```
template<typename _T = double>
bool _csv< _T >::check_dim ( )
```

Check data dimension consistency, i.e. if all line dimensions are all equal.

**Returns**

true if dimensions seem OK

### 2.1.3.6 `empty()`

```
template<typename _T = double>
bool _csv<_T>::empty ( ) const
```

Check if data are empty, and the emptiness of the first line, i.e. `this->data[0]`.

#### Returns

true if data are empty

### 2.1.3.7 `get_data()`

```
template<typename _T = double>
const std::vector< std::vector<_T> > & _csv<_T>::get_data ( ) const
```

Get data and return it as a vector of vector.

#### Returns

`std::vector<std::vector<_T> >`

### 2.1.3.8 `get_data_size_i()`

```
template<typename _T = double>
const size_t _csv<_T>::get_data_size_i ( ) const
```

Get data line size.

#### Returns

`size_t`

### 2.1.3.9 `get_data_size_j()`

```
template<typename _T = double>
const size_t _csv<_T>::get_data_size_j ( ) const
```

Get data column size.

#### Returns

`size_t`

#### 2.1.3.10 `get_filename()`

```
template<typename _T = double>
const std::string _csv< _T >::get_filename ( ) const
```

Get the filename.

##### Returns

`std::string`

#### 2.1.3.11 `get_filename_out()`

```
template<typename _T = double>
const std::string _csv< _T >::get_filename_out ( ) const
```

Get the output filename.

##### Returns

`std::string`

#### 2.1.3.12 `get_header()`

```
template<typename _T = double>
const std::vector< _T > & _csv< _T >::get_header ( ) const
```

Get column names and return it in a vector.

##### Returns

`std::vector<_T>`

#### 2.1.3.13 `get_header_size()`

```
template<typename _T = double>
const size_t _csv< _T >::get_header_size ( ) const
```

Get size of the header.

##### Returns

`size_t`

2.1.3.14 `get_separator()`

```
template<typename _T = double>
const char _csv<_T>::get_separator ( ) const
```

Get the separator.

**Returns**

char

2.1.3.15 `read()`

```
template<typename _T = double>
bool _csv<_T>::read ( )
```

Read the content of the file given to the constructor using boost. It detects the header and data consistency with digit sequence: {0123456789e+-, tab std::endl} and basic regex and dimension matching between header and data line. It is able to recover basic errors such as 'tab'=='. The method put NaN in the grid if an unrecoverable error appends. Data will be store in private variables.

**Returns**

true if all seems OK

2.1.3.16 `select()`

```
template<typename _T = double>
const std::vector< std::vector<_T> > & _csv<_T>::select (
    int iLine_min,
    int iLine_max,
    int iCol_min,
    int iCol_max ) const
```

Select a sub grid in data, i.e. trim data to the rectangular  $[i_{min}, i_{max}] \times [j_{min}, j_{max}]$ .

**Parameters**

<i>iLine_min</i>	upper line $i_{min}$
<i>iLine_max</i>	lower line $i_{max}$
<i>iCol_min</i>	left column $j_{min}$
<i>iCol_max</i>	right column $j_{max}$

**Returns**

`std::vector<std::vector<_T> >`

**2.1.3.17 select\_column()**

```
template<typename _T = double>
const std::vector< _T > & _csv< _T >::select_column (
    int iCol ) const
```

Select the column "col" in data.

**Parameters**

<i>iCol</i>	The column to select
-------------	----------------------

**Returns**

`std::vector<_T>`

**2.1.3.18 select\_line()**

```
template<typename _T = double>
const std::vector< _T > & _csv< _T >::select_line (
    int iLine ) const
```

Select the line "line" in data.

**Parameters**

<i>iLine</i>	The line to select
--------------	--------------------

**Returns**

`std::vector<_T>`

**2.1.3.19 set\_column()**

```
template<typename _T = double>
bool _csv< _T >::set_column (
    const std::vector< _T > & vRow,
    int iRow )
```

Set a column with a vector.

Set a row with a vector.

**Parameters**

<i>vCol</i>	std::vector<_T> vCol
<i>iCol</i>	Select a column

**Returns**

true if all seems OK

**Parameters**

<i>vRow</i>	std::vector<_T> vRow
<i>iRow</i>	Select a row

**Returns**

true if all seems OK

**2.1.3.20 set\_data()**

```
template<typename _T = double>
void _csv< _T >::set_data (
    const std::vector< std::vector< _T > > & vvData )
```

Set data with a vector of a vector.

**Parameters**

<i>vvData</i>	std::vector<std::vector<_T> > grid
---------------	------------------------------------

**Returns**

true if all seems OK

**2.1.3.21 set\_filename()**

```
template<typename _T = double>
bool _csv< _T >::set_filename (
    const std::string & sFilename )
```

Set the filename for output or input. The fstream do not care about extension...



## Parameters

<i>sFilename</i>	The filename with extension or not.
------------------	-------------------------------------

## Returns

true if all seems OK

2.1.3.22 `set_filename_out()`

```
template<typename _T = double>
bool _csv<_T>::set_filename_out (
    const std::string & sFilename )
```

Set the filename for output. The fstream do not care about extension...

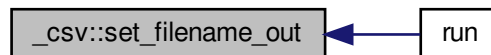
## Parameters

<i>sFilename</i>	The filename with extension or not.
------------------	-------------------------------------

## Returns

true if all seems OK

Here is the caller graph for this function:

2.1.3.23 `set_header()`

```
template<typename _T = double>
bool _csv<_T>::set_header (
    const std::vector< std::string > & vsHeader )
```

Set the header: the first line containing column name.

**Parameters**

<i>vsHeader</i>	string vector
-----------------	---------------

**Returns**

true if all seems OK

**2.1.3.24 set\_separator()** [1/2]

```
template<typename _T = double>
bool _csv< _T >::set_separator (
    const char & cSep )
```

Set the csv separator. Usually: '\t', ',', ';;', ';' ...

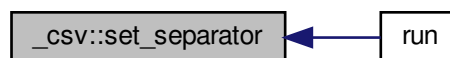
**Parameters**

<i>cSep</i>	The sep character: '\t' for tabulation
-------------	--

**Returns**

true if all seems OK

Here is the caller graph for this function:

**2.1.3.25 set\_separator()** [2/2]

```
template<typename _T = double>
bool _csv< _T >::set_separator (
    const std::string & sSep )
```

Set the csv separator. Usually: '\t', ',', ';;', ';' ...

**Todo**

## Parameters

<code>sSep</code>	The sep character: '\t' for tabulation
-------------------	--

## Returns

true if all seems OK

2.1.3.26 `set_verbose()`

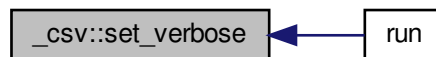
```
template<typename _T = double>
bool _csv<_T>::set_verbose (
    eVerbose evV )
```

Set the verbose mode for debug. It does not deactivate error raising.

## Parameters

<code>evV</code>	<code>eVerbose::DEBUG</code> for verbose mode and <code>eVerbose::QUIET</code> to keep quiet
------------------	--

Here is the caller graph for this function:

2.1.3.27 `show()` [1/2]

```
template<typename _T = double>
void _csv<_T>::show ( ) const
```

Show whole data, i.e. the header and data with no restriction on length or terminal size. It uses `boost::format` in order to correct spacing of number and strings.

## Returns

true if all seems OK

**2.1.3.28 show()** [2/2]

```
template<typename _T = double>
bool _csv< _T >::show (
    int iLine_stop ) const
```

Show the header and data until "line\_stop" line. Print all columns with terminal end-of-line. It uses boost::format in order to correct spacing of number and strings.

**Parameters**

<i>iLine_stop</i>	The number of lines where stop the display
-------------------	--

**Returns**

true if all seems OK

**2.1.3.29 transform\_lin()**

```
template<typename _T = double>
bool _csv< _T >::transform_lin (
    _T TA,
    _T TB,
    int iCol )
```

Do  $Y=aX+b$  to the iCol-column.

**Returns**

true if all seems OK

**2.1.3.30 write()**

```
template<typename _T = double>
bool _csv< _T >::write ( )
```

Write on disk what data are store.

## Returns

true if all seems OK

Here is the caller graph for this function:



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

- [csv.h](#)

2.2 `_io<_T>` Class Template Reference

The purpose of this class is only to provide methods to read ascii or FITS spectrum, and to convert vector to valarray.

```
#include <waverage.hpp>
```

## Classes

- struct [vec](#)  
*Define a (x,y) vector (seems better than std::pair)*

## Public Types

- typedef `std::vector< std::valarray< std::valarray<_T>>> Vvv`
- typedef `std::valarray< std::valarray<_T>> vv`

### Public Member Functions

- `_io` (Vvv &VvvSpectr)
- `bool read` (std::string sFilename)
- `bool read_dir` (std::string sExtension)
- `bool read_fits` (std::string sFilename)
- `bool read_fits_dir` (std::string sExtension)
- `bool read_fits_dir` (std::string sDirectory, std::string sExtension)
- `void show_data` () const
- `void show_data` (int n) const
- `void show_data` (std::string sName) const
- `bool write` () const
- `bool write` (std::string sFilename) const
- `void set_fileIn` (std::string sFilename)
- `void set_fileOut` (std::string sFilename)
- `void set_data` (Vvv &VvvSpectr)
- `void set_WaveScale` (\_T Scale)
- `int get_FileIndex` (std::string sName) const
- `const std::vector< vec > get_vector` () const
- `const std::vector< std::vector< vec > > get_vectors` () const
- `const vv get_valarray` ()
- `const Vvv get_valarrays` ()

### Static Public Member Functions

- `static void notify` (std::string sTitle, std::string sMsg, int iTime)

## 2.2.1 Detailed Description

```
template<typename _T = double long>
class _io<_T>
```

The purpose of this class is only to provide methods to read ascii or FITS spectrum, and to convert vector to valarray.

## 2.2.2 Member Function Documentation

### 2.2.2.1 get\_FileIndex()

```
template<typename _T = double long>
int _io<_T>::get_FileIndex (
    std::string sName ) const
```

return the position of sName in vsFileList

### 2.2.2.2 `get_valarray()`

```
template<typename _T = double long>
const vv _io<_T>::get_valarray ( )
```

convert vector to an valarray

### 2.2.2.3 `get_vector()`

```
template<typename _T = double long>
const std::vector<vec> _io<_T>::get_vector ( ) const
```

get spectrum

### 2.2.2.4 `get_vectors()`

```
template<typename _T = double long>
const std::vector<std::vector<vec> > _io<_T>::get_vectors ( ) const
```

get spectra

### 2.2.2.5 `notify()`

```
template<typename _T = double long>
static void _io<_T>::notify (
    std::string sTitle,
    std::string sMsg,
    int iTime ) [static]
```

send desktop notification using libnotify.

### 2.2.2.6 `read()`

```
template<typename _T = double long>
bool _io<_T>::read (
    std::string sFilename )
```

read one file

### 2.2.2.7 `read_dir()`

```
template<typename _T = double long>
bool _io<_T>::read_dir (
    std::string sExtension )
```

read the whole directory, only need the extension of files

#### 2.2.2.8 read\_fits()

```
template<typename _T = double long>
bool _io< _T >::read_fits (
    std::string sFilename )
```

convert fits to vectors

#### 2.2.2.9 set\_fileIn()

```
template<typename _T = double long>
void _io< _T >::set_fileIn (
    std::string sFilename )
```

set the input file name

#### 2.2.2.10 set\_fileOut()

```
template<typename _T = double long>
void _io< _T >::set_fileOut (
    std::string sFilename )
```

set the output name

#### 2.2.2.11 set\_WaveScale()

```
template<typename _T = double long>
void _io< _T >::set_WaveScale (
    _T Scale )
```

multiply wavelength by Scale

#### 2.2.2.12 show\_data() [1/3]

```
template<typename _T = double long>
void _io< _T >::show_data ( ) const
```

show a spectrum

#### 2.2.2.13 show\_data() [2/3]

```
template<typename _T = double long>
void _io< _T >::show_data (
    int n ) const
```

show spectrum n



2.2.2.14 `show_data()` [3/3]

```
template<typename _T = double long>
void _io< _T >::show_data (
    std::string sName ) const
```

show spectrum sName

2.2.2.15 `write()` [1/2]

```
template<typename _T = double long>
bool _io< _T >::write ( ) const
```

write the results

2.2.2.16 `write()` [2/2]

```
template<typename _T = double long>
bool _io< _T >::write (
    std::string sFilename ) const
```

write the results

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

- `waverage.hpp`

2.3 `_log` Class Reference

## Public Member Functions

- `_log` (char \*\*argv, const std::string &sFilename)  
*Initialize with the first argument of the command line, and the log name.*
- const std::string `get_execname` () const
- const std::string `get_logname` () const
- const std::string `get_historyname` () const
- bool `set_execname` (char \*\*argv)
- bool `set_logname` (const std::string &sFilename)
- bool `set_historyname` (const std::string &sFilename)
- bool `write` (const std::string &sS)  
*Write a string in the log file.*
- bool `write` (const std::stringstream &ssS)
- bool `write_history` (const boost::program\_options::variables\_map &vm)  
*Write history file with information from boost::program\_options.*
- bool `remove_duplicate` ()  
*Remove duplicates in history file.*

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

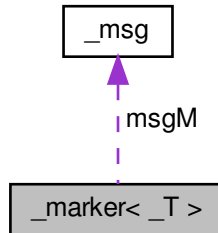
- `log.h`
- `log.cpp`

## 2.4 `_marker<_T>` Class Template Reference

A class to plot spectra with line markers using py matplotlib.

```
#include <marker.h>
```

Collaboration diagram for `_marker<_T>`:



### Classes

- struct [Line](#)  
*Define a line.*

### Public Types

- typedef `std::vector< Line >` **vList**

### Public Member Functions

- void **set\_verbose** (const bool bVerbose)
- bool **set\_data** (const std::vector<\_T> &vTX, const std::vector<\_T> &vTY)
- bool **set\_title** (const std::string &sTitle)
- bool **set\_label** (const std::string &sLabel)
- bool **set\_xlabel** (const std::string &sXlabel)
- bool **set\_ylabel** (const std::string &sYlabel)
- bool **set\_xunit** (const std::string &sXunit)
- bool **set\_yunit** (const std::string &sYunit)
- bool **set\_output** (const std::string &sFilename)
- bool **set\_output** (const std::string &sFilename, const int iDpi)  
*Set the picture filename with the extension (png, pdf, jpeg...) and the density (iDpi>50)*
- bool **set\_continuum** (const \_T TContinuum)  
*Set the continuum position and therefore ymax. Default is y=1.*
- bool **set\_supp** (const \_T TXmin, const \_T TXmax)

*Set the support of the first spectrum.*

- bool **set\_xmin** (const \_T TXmin)
- bool **set\_xmax** (const \_T TXmax)
- bool **set\_ymin** (const \_T TYmin)
- bool **set\_ymax** (const \_T TYmax)
- bool **set\_figsize** (int iHeight, int iWidth)
- void **set\_colorline** (const std::string &sColor)

*Set the color of the first curve.*

- bool **set\_linewidth** (float fWidth)
- bool **set\_titlesize** (int iSize)
- bool **set\_labelsize** (int iSize)
- bool **set\_ticklabelsize** (int iSize)
- bool **set\_annotatesize** (int iSize)

*Set the font size of markers.*

- bool **set\_legendsize** (int iSize)
- void **set\_legend** (bool bLegend)

*Enable or disable the legend.*

- void **set\_halfbox** (bool bHalfbox)

*Show only left and bottom axis.*

- bool **set\_contnumsize** (float fWidth)
- void **set\_showgrid** (bool bShowgrid)
- void **set\_dotted** (bool bDotted)

*Set secondary curves with dotted-style.*

- void **set\_dotdashed** (bool bDotdashed)

*Set secondary curves with dot-dashed-style.*

- void **set\_wide** (bool bWide)

*Define if the spectrum range is wide in order to reduce marker size with no overlaps.*

- bool **set\_scriptname** (const std::string &sScriptname)

*Set the name of the py script. Default is .plot.py.*

- bool **set\_log** (const std::string &sLog)

*Enable or disable log file. Default is .marker.log.*

- bool **add\_line** (\_T TWI, const std::string &sName)

*Add a marker with a name on the figure.*

- bool **add\_line** (\_T TWI, const std::string &sName, bool bBold)

*Add a marker with a name on the figure. bBold determines if the line must be highlighted.*

- bool **add\_data** (const std::vector<\_T> &vTX, const std::vector<\_T> &vTY)

*Add an additionnal spectrum which has to be plot.*

- bool **add\_data** (const std::vector<\_T> &vTX, const std::vector<\_T> &vTY, const std::string &sLabel)

*Add an additionnal spectrum which has to be plot.*

- \_T **get\_continuum** () const
- const std::pair<\_T, \_T> **get\_supp** ()

*Get the support of the first spectrum.*

- const std::string &**get\_scriptname** ()
- const std::string &**get\_output** ()
- const std::string &**get\_title** () const
- const std::string &**get\_label** () const
- const std::string &**get\_xlabel** () const
- const std::string &**get\_xunit** () const

- const std::string & **get\_ylabel** () const
- const std::string & **get\_yunit** () const
- const std::pair< int, int > **get\_figsize** () const  
*Get the defined figsize, if defined. First: Height and Second: Width.*
- int **get\_dpi** () const
- bool **make** ()  
*Write spectra, write script with markers.*
- int **plot** ()  
*Run the py script?*

### Static Public Member Functions

- static bool **sort\_elemlist** (const std::string &sElemlist)  
*Sort the elemlist.*

### Protected Attributes

- **\_msg msgM**

## 2.4.1 Detailed Description

```
template<typename _T = float>
class _marker< _T >
```

A class to plot spectra with line markers using py matplotlib.

**Todo** marker(const \_marker<\_T>&)

## 2.4.2 Member Function Documentation

### 2.4.2.1 get\_figsize()

```
template<typename _T = float>
const std::pair< int, int > _marker< _T >::get_figsize ( ) const
```

Get the defined figsize, if defined. First: Height and Second: Width.

### Returns

std::pair of 2 int

2.4.2.2 `get_supp()`

```
template<typename _T = float>
const std::pair<_T, _T> _marker<_T>::get_supp ( )
```

Get the support of the first spectrum.

## Returns

`std::pair of 2 _T: [  $x_{min}$   $x_{max}$  ]`

2.4.2.3 `set_colorline()`

```
template<typename _T = float>
void _marker<_T>::set_colorline (
    const std::string & sColor )
```

Set the color of the first curve.

## Parameters

<i>sColor</i>	A string like "red", "green", "blue" or and a rgba hex string like "#rrggbbaa"
---------------	--

2.4.2.4 `set_output()`

```
template<typename _T = float>
bool _marker<_T>::set_output (
    const std::string & sFilename,
    const int iDpi )
```

Set the picture filename with the extension (png, pdf, jpeg...) and the density (iDpi>50)

## Parameters

<i>sFilename</i>	Picture name
<i>iDpi</i>	Density

2.4.2.5 `set_supp()`

```
template<typename _T = float>
bool _marker<_T>::set_supp (
```

```
const _T TXmin,
const _T TXmax )
```

Set the support of the first spectrum.

#### Parameters

<i>TXmin</i>	$x_{min}$
<i>TXmax</i>	$x_{max}$

### 2.4.3 Member Data Documentation

#### 2.4.3.1 msgM

```
template<typename _T = float>
_msg _marker< _T >::msgM [protected]
```

Interface to print message to std output

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

- marker.h

## 2.5 \_msg Class Reference

A class that sends string to std output and in a file...

```
#include <msg.h>
```

#### Public Types

- enum `eMsg` {  
    **START, MID, END, ERROR,**  
    **THREADS }**

*enum for method in order to define whether the message is at the begin, at the end or an error,*

## Public Member Functions

- `_msg` (const `_msg` &other)
- bool `msg` (const std::string &sMsg)  
*Send a message with `eMsg::MID` as default.*
- bool `msg` (`eMsg` emType, const std::string &sMsg)  
*Send a message...*
- bool `error` (const std::string &sMsg)  
*Send an error message...*
- template<typename ... Args>  
bool `msg` (const Args &...args)  
*A variadic formatter method that indeed sends arbitrary number of variable to the std output... with `eMsg::MID` as default.*
- template<typename ... Args>  
bool `msg` (`eMsg` emType, const Args &...args)  
*A variadic formatter method that indeed sends arbitrary number of variable to the std output... The first parameter is always the enum `eMsg`.*
- template<typename ... Args>  
bool `error` (const Args &...args)  
*A variadic formatter method that indeed sends arbitrary number of variable to the std error output... with `eMsg::ERROR` as default.*
- bool `set_name` (const std::string sName)  
*Set the name of the main instance.*
- bool `set_threadname` (const std::string sName)  
*Set the name of threads.*
- bool `set_log` (const std::string sLog)  
*Enable or disable log file.*
- void `enable_log` (bool bLog)  
*Enable or disable the log file.*

## 2.5.1 Detailed Description

A class that sends string to std output and in a file...

## 2.5.2 Member Function Documentation

2.5.2.1 `msg()`

```
bool _msg::msg (
    eMsg emType,
    const std::string & sMsg )
```

Send a message...

## Parameters

<i>emType</i>	See enum eMsg::
---------------	--------------------

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

- [msg.h](#)
- [msg.cpp](#)

## 2.6 `_op<_T>` Class Template Reference

This class is a set of spectrum manipulation methods. It is work with `std::valarray`. everything.

```
#include <waverage.hpp>
```

### Public Types

- `typedef std::vector< std::valarray< std::valarray<_T>>> Vvv`
- `typedef std::valarray< std::valarray<_T>> vv`
- `typedef std::vector< std::vector<_T>> VV`

### Public Member Functions

- `_op (Vvv &VvvSpectr)`
- `bool resize\_spectr ()`
- `bool rebuild\_wlStep ()`
- `void remove\_zero ()`
- `bool filter\_SG (int n)`
- `vv compute\_SG2 (vv &vvSpectr) const`
- `bool filter\_SG ()`
- `void remove_peaks (int n)`
- `const vv compute\_mean () const`
- `void compute\_wmean ()`
- `std::pair<_T,_T> get\_wlRange (int n) const`
- `std::pair<_T,_T> get\_wlRangeMin () const`
- `void set\_data (Vvv &VvvSpectr)`
- `bool write (_io<_T> &ioInterface)`
- `bool write\_mean (std::string sFilename) const`

#### 2.6.1 Detailed Description

```
template<typename _T = double long>
class _op<_T>
```

This class is a set of spectrum manipulation methods. It is work with `std::valarray`. everything.



## 2.6.2 Member Function Documentation

### 2.6.2.1 `compute_mean()`

```
template<typename _T = double long>
const vv _op<_T>::compute_mean ( ) const
```

compute arithmetic mean

### 2.6.2.2 `compute_SG2()`

```
template<typename _T = double long>
vv _op<_T>::compute_SG2 (
    vv & vvSpectr ) const [inline]
```

Savitzky-Golay 2nd order.

### 2.6.2.3 `compute_wmean()`

```
template<typename _T = double long>
void _op<_T>::compute_wmean ( )
```

compute weighted arithmetic mean

### 2.6.2.4 `filter_SG()` [1/2]

```
template<typename _T = double long>
bool _op<_T>::filter_SG (
    int n ) [inline]
```

Savitzky-Golay on spectrum n.

### 2.6.2.5 `filter_SG()` [2/2]

```
template<typename _T = double long>
bool _op<_T>::filter_SG ( )
```

Savitzky-Golay on all spectra n.

### 2.6.2.6 `get_wlRange()`

```
template<typename _T = double long>
std::pair<_T, _T> _op<_T>::get_wlRange (
    int n ) const
```

get the minimum and maximum wavelength

**2.6.2.7 get\_wlRangeMin()**

```
template<typename _T = double long>
std::pair<_T, _T> _op< _T >::get_wlRangeMin ( ) const
```

return the smallest support

**2.6.2.8 rebuild\_wlStep()**

```
template<typename _T = double long>
bool _op< _T >::rebuild_wlStep ( )
```

rebuild wavelength axis.

**2.6.2.9 remove\_zero()**

```
template<typename _T = double long>
void _op< _T >::remove_zero ( )
```

trim spectra where flux is 0 (assuming zeros are at the beginning or the end).

**2.6.2.10 resize\_spectr()**

```
template<typename _T = double long>
bool _op< _T >::resize_spectr ( )
```

resize all spectra to the same size (maximal).

**2.6.2.11 set\_data()**

```
template<typename _T = double long>
void _op< _T >::set_data (
    Vvv & VvvSpectr )
```

set external spectra.

**2.6.2.12 write()**

```
template<typename _T = double long>
bool _op< _T >::write (
    _io< _T > & ioInterface )
```

write the data using a `_io` class.

2.6.2.13 `write_mean()`

```
template<typename _T = double long>
bool _op<_T>::write_mean (
    std::string sFilename ) const
```

write only the w mean.

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

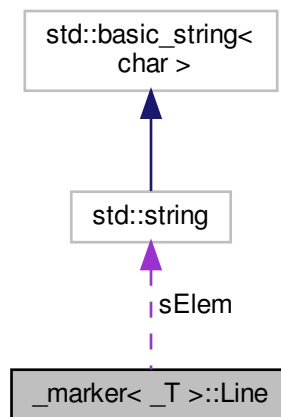
- `waverage.hpp`

2.7 `_marker<_T>::Line` Struct Reference

Define a line.

```
#include <marker.h>
```

Collaboration diagram for `_marker<_T>::Line`:



## Public Attributes

- `_T` **TWI**
- `std::string` **sElem**
- `bool` **bBold**

### 2.7.1 Detailed Description

```
template<typename _T = float>
struct _marker<_T>::Line
```

Define a line.

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

- marker.h

## 2.8 \_io<\_T>::vec Struct Reference

Define a (x,y) vector (seems better than std::pair)

```
#include <waverage.hpp>
```

### Public Attributes

- **\_T x**
- **\_T y**
- **\_T SNR = -1**

### 2.8.1 Detailed Description

```
template<typename _T = double long>
struct _io<_T>::vec
```

Define a (x,y) vector (seems better than std::pair)

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

- waverage.hpp

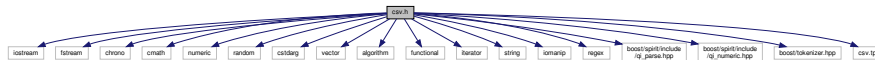
## 3 File Documentation

### 3.1 csv.h File Reference

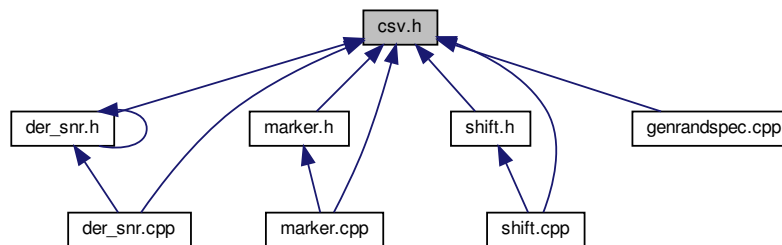
A basic class for csv manipulation.

```
#include <iostream>
#include <fstream>
#include <chrono>
#include <cmath>
#include <numeric>
#include <random>
#include <cstdint>
#include <vector>
#include <algorithm>
#include <functional>
#include <iterator>
#include <string>
#include <iomanip>
#include <regex>
#include <boost/spirit/include/qi_parse.hpp>
#include <boost/spirit/include/qi_numeric.hpp>
#include <boost/tokenizer.hpp>
#include "csv.hpp"
```

Include dependency graph for csv.h:



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



#### Classes

- class `_csv<_T>`

*This is the templated `_csv` class, initialized with double by default. STL parallel execution policy does not provide enhancements for simple operations.*

### 3.1.1 Detailed Description

A basic class for csv manipulation.

#### Author

Audric Lemonnier

#### Version

0.9

#### Date

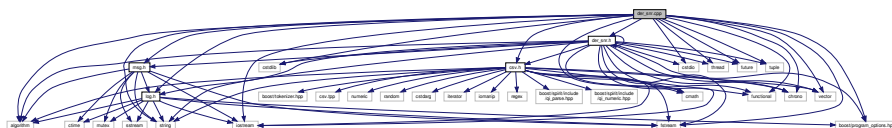
07/04/2020

## 3.2 der\_snr.cpp File Reference

An C++ implementation of the der\_snr fortran code from: F. Stoehr et al: DER\_SNR: A Simple & General Spectroscopic Signal-to-Noise Measurement Algorithm, 394, Astronomical Data Analysis Software and Systems (ADASS) XVII 2008ASPC..394..505S.

```
#include <iostream>
#include <cstdlib>
#include <cstdio>
#include <fstream>
#include <vector>
#include <algorithm>
#include <string>
#include <cmath>
#include <functional>
#include <thread>
#include <future>
#include <tuple>
#include <chrono>
#include <boost/program_options.hpp>
#include <csv.h>
#include <msg.h>
#include <log.h>
#include <der_snr.h>
```

Include dependency graph for der\_snr.cpp:



## Functions

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

## 3.2.1 Detailed Description

An C++ implementation of the der\_snr fortran code from: F. Stoehr et al: DER\_SNR: A Simple & General Spectroscopic Signal-to-Noise Measurement Algorithm, 394, Astronomical Data Analysis Software and Systems (ADASS) XVII 2008ASPC..394..505S.

Remove value under a threshold in a folder or in a file. This code is multi-threaded or not if not available.

## Author

Audric Lemonnier

## Version

0.2

## Date

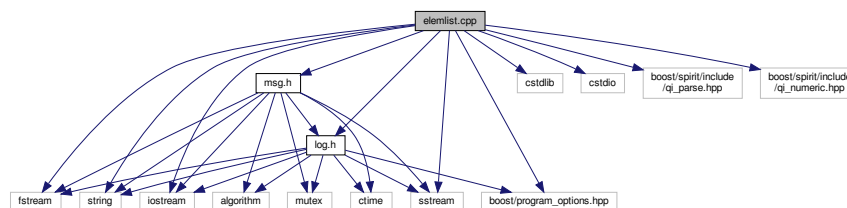
18/04/2020

## 3.3 elemlist.cpp File Reference

Add a line to the elemlist.

```
#include <iostream>
#include <cstdlib>
#include <cstdio>
#include <fstream>
#include <string>
#include <sstream>
#include <boost/program_options.hpp>
#include <boost/spirit/include/qi_parse.hpp>
#include <boost/spirit/include/qi_numeric.hpp>
#include <msg.h>
#include <log.h>
```

Include dependency graph for elemlist.cpp:



## Macros

- `#define LOGFILE ".elemlist.log"`
- `#define HISTFILE ".history"`

## Functions

- `template<typename _T = std::string>`  
`bool add_elem (const std::string &sElem, _T TWI, const std::string &sFilename)`  
*Add a line to a file.*
- `template<typename _T = std::string>`  
`bool add_elem (const std::string &sSymbol, const std::string &sElem, _T TWI, const std::string &sFilename)`  
*Add a line to a file, with the indicator symbol.*
- `bool is_float (const std::string &sVal)`  
*Determine if a string is a number.*
- `int main (int argc, char **argv)`

### 3.3.1 Detailed Description

Add a line to the elemlist.

#### Author

Audric Lemonnier

#### Version

0.1

#### Date

30/03/2020

### 3.3.2 Macro Definition Documentation

#### 3.3.2.1 HISTFILE

```
#define HISTFILE ".history"
```

Define the default histfile (shared)



## 3.3.2.2 LOGFILE

```
#define LOGFILE ".elemlist.log"
```

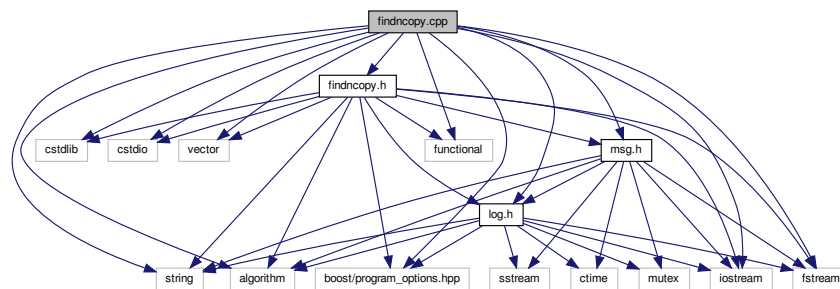
Define the default logfile

## 3.4 findncopy.cpp File Reference

Copy files from a list in a new folder.

```
#include <iostream>
#include <cstdlib>
#include <cstdio>
#include <fstream>
#include <vector>
#include <string>
#include <algorithm>
#include <functional>
#include <boost/program_options.hpp>
#include <msg.h>
#include <log.h>
#include <findncopy.h>
```

Include dependency graph for findncopy.cpp:



## Macros

- #define **LOGFILE** ".findncopy.log"
- #define **HISTFILE** ".history"

## Functions

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

### 3.4.1 Detailed Description

Copy files from a list in a new folder.

#### Author

Audric Lemonnier

#### Version

0.1

#### Date

09/03/2020

### 3.4.2 Macro Definition Documentation

#### 3.4.2.1 HISTFILE

```
#define HISTFILE ".history"
```

Define the default histfile (shared)

#### 3.4.2.2 LOGFILE

```
#define LOGFILE ".findncopy.log"
```

Define the default logfile

## 3.5 genrandspec.cpp File Reference

Generate a set of randomized-flux spectra between two wavelengths for test purposes.

```
#include <iostream>
#include <cstdlib>
#include <cstdio>
#include <fstream>
#include <vector>
#include <algorithm>
#include <numeric>
#include <string>
#include <cmath>
#include <random>
#include <thread>
```

```

#include <future>
#include <ctime>
#include <tuple>
#include <chrono>
#include <boost/program_options.hpp>
#include <csv.h>
#include <msg.h>
#include <log.h>

```

Include dependency graph for genrandspec.cpp:



## Macros

- #define **LOGFILE** ".genrandspec.log"
- #define **HISTFILE** ".history"
- #define **MaxFileDir** 10  
Set the maximum number of files to create in a folder.

## Functions

- void **run** (const std::string &sOutput, char cSep, float fMinw, float fMaxw, float fStep)  
Write random spectra on disk.
- double long **CPU\_utilization** ()
- std::tuple< double long, double long > **get\_stat** ()
- int **main** (int argc, char \*\*argv)

### 3.5.1 Detailed Description

Generate a set of randomized-flux spectra between two wavelengths for test purposes.

## Author

Audric Lemonnier

## Version

0.4

## Date

18/04/2020

### 3.5.2 Macro Definition Documentation

#### 3.5.2.1 HISTFILE

```
#define HISTFILE ".history"
```

Define the default histfile (shared)

#### 3.5.2.2 LOGFILE

```
#define LOGFILE ".genrandspec.log"
```

Define the default logfile

#### 3.5.2.3 MaxFilepDir

```
#define MaxFilepDir 10
```

Set the maximum number of files to create in a folder.

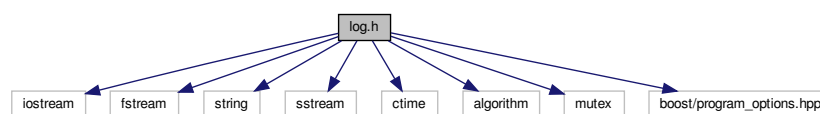
MaxFilepDir

## 3.6 log.h File Reference

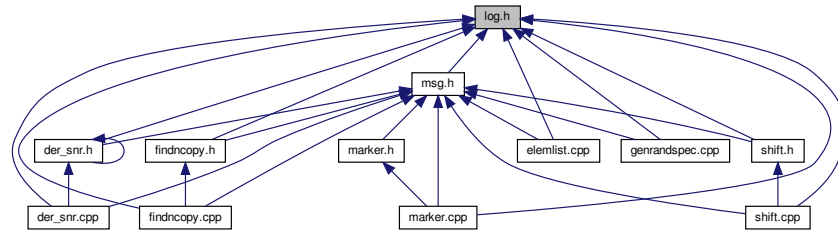
A class to write log file.

```
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <ctime>
#include <algorithm>
#include <mutex>
#include <boost/program_options.hpp>
```

Include dependency graph for log.h:



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



## Classes

- class [\\_log](#)

### 3.6.1 Detailed Description

A class to write log file.

#### Author

Audric Lemonnier

#### Version

0.1

#### Date

03/05/2020

## 3.7 marker.cpp File Reference

Highlight lines on spectrum.

```

#include <iostream>
#include <cstdlib>
#include <cstdio>
#include <fstream>
#include <vector>
#include <tuple>
#include <string>
#include <algorithm>
#include <iterator>
#include <limits>

```

```
#include <boost/program_options.hpp>
#include <boost/spirit/include/qi_parse.hpp>
#include <boost/spirit/include/qi_numeric.hpp>
#include <marker.h>
#include <msg.h>
#include <log.h>
#include <csv.h>
```

Include dependency graph for marker.cpp:



## Macros

- #define LOGFILE ".marker.log"
- #define HISTFILE ".history"

## Functions

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

### 3.7.1 Detailed Description

Highlight lines on spectrum.

## Author

Audric Lemonnier

## Version

0.6

## Date

29/04/2020

### 3.7.2 Macro Definition Documentation

## 3.7.2.1 HISTFILE

```
#define HISTFILE ".history"
```

Define the default histfile (shared)

## 3.7.2.2 LOGFILE

```
#define LOGFILE ".marker.log"
```

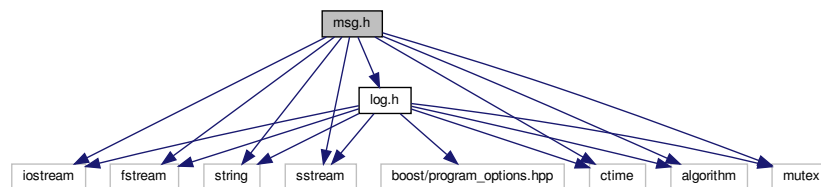
Define the default logfile

## 3.8 msg.h File Reference

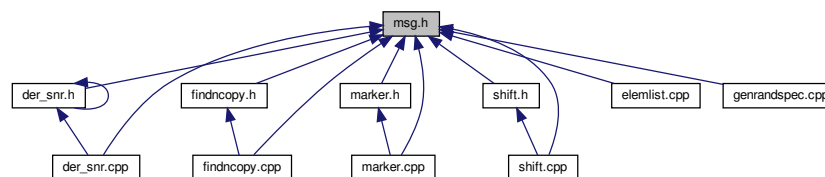
A class to print and write message.

```
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <ctime>
#include <algorithm>
#include <mutex>
#include <log.h>
```

Include dependency graph for msg.h:



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



## Classes

- class [\\_msg](#)

*A class that sends string to std output and in a file...*

### 3.8.1 Detailed Description

A class to print and write message.

#### Author

Audric Lemonnier

#### Version

0.2

#### Date

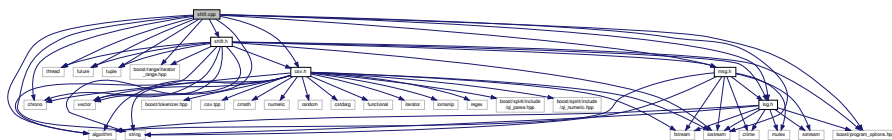
18/04/2020

## 3.9 shift.cpp File Reference

Shift whole spectrum by a given wavelength. This code is multi-threaded or not if not available.

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <thread>
#include <future>
#include <string>
#include <tuple>
#include <chrono>
#include <boost/program_options.hpp>
#include <boost/range/iterator_range.hpp>
#include <csv.h>
#include <msg.h>
#include <log.h>
#include <shift.h>
```

Include dependency graph for shift.cpp:





## Macros

- `#define CLIGHT 299792.458`
- `#define LOGFILE ".shift.log"`
- `#define HISTFILE ".history"`

## Functions

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

### 3.9.1 Detailed Description

Shift whole spectrum by a given wavelength. This code is multi-threaded or not if not available.

## Author

Audric Lemonnier

## Version

0.3

## Date

18/04/2020

### 3.9.2 Macro Definition Documentation

#### 3.9.2.1 HISTFILE

```
#define HISTFILE ".history"
```

Define the default histfile (shared)

#### 3.9.2.2 LOGFILE

```
#define LOGFILE ".shift.log"
```

Define the default logfile



## Index

### `_csv`

- `_csv`, 5, 6
- `apply_max_threshold`, 6, 7
- `apply_min_threshold`, 7, 8
- `check_dim`, 8
- `empty`, 8
- `get_data`, 9
- `get_data_size_i`, 9
- `get_data_size_j`, 9
- `get_filename`, 9
- `get_filename_out`, 10
- `get_header`, 10
- `get_header_size`, 10
- `get_separator`, 10
- `read`, 11
- `select`, 11
- `select_column`, 12
- `select_line`, 12
- `set_column`, 12
- `set_data`, 14
- `set_filename`, 14
- `set_filename_out`, 15
- `set_header`, 15
- `set_separator`, 16
- `set_verbose`, 17
- `show`, 17
- `transform_lin`, 18
- `write`, 18

### `_csv< _T >`, 2

### `_io`

- `get_FileIndex`, 20
- `get_valarray`, 20
- `get_vector`, 21
- `get_vectors`, 21
- `notify`, 21
- `read`, 21
- `read_dir`, 21
- `read_fits`, 21
- `set_WaveScale`, 22
- `set_fileIn`, 22
- `set_fileOut`, 22
- `show_data`, 22
- `write`, 23

### `_io< _T >`, 19

### `_io< _T >::vec`, 34

### `_log`, 23

### `_marker`

- `get_figsize`, 26
- `get_supp`, 26
- `msgM`, 28
- `set_colorline`, 27

### `set_output`, 27

### `set_supp`, 27

### `_marker< _T >`, 24

### `_marker< _T >::Line`, 33

### `_msg`, 28

### `msg`, 29

### `_op`

- `compute_SG2`, 31
- `compute_mean`, 31
- `compute_wmean`, 31
- `filter_SG`, 31
- `get_wlRange`, 31
- `get_wlRangeMin`, 31
- `rebuild_wlStep`, 32
- `remove_zero`, 32
- `resize_spectr`, 32
- `set_data`, 32
- `write`, 32
- `write_mean`, 32

### `_op< _T >`, 30

### `apply_max_threshold`

#### `_csv`, 6, 7

### `apply_min_threshold`

#### `_csv`, 7, 8

### `check_dim`

#### `_csv`, 8

### `compute_SG2`

#### `_op`, 31

### `compute_mean`

#### `_op`, 31

### `compute_wmean`

#### `_op`, 31

### `csv.h`, 35

### `der_snr.cpp`, 36

### `elemlist.cpp`, 37

#### `HISTFILE`, 38

#### `LOGFILE`, 38

### `empty`

#### `_csv`, 8

### `filter_SG`

#### `_op`, 31

### `findncopy.cpp`, 39

#### `HISTFILE`, 40

#### `LOGFILE`, 40

### `genrandspec.cpp`, 40

#### `HISTFILE`, 42

- LOGFILE, 42
  - MaxFilepDir, 42
- get\_FileIndex
  - \_io, 20
- get\_data
  - \_csv, 9
- get\_data\_size\_i
  - \_csv, 9
- get\_data\_size\_j
  - \_csv, 9
- get\_figsize
  - \_marker, 26
- get\_filename
  - \_csv, 9
- get\_filename\_out
  - \_csv, 10
- get\_header
  - \_csv, 10
- get\_header\_size
  - \_csv, 10
- get\_separator
  - \_csv, 10
- get\_supp
  - \_marker, 26
- get\_valarray
  - \_io, 20
- get\_vector
  - \_io, 21
- get\_vectors
  - \_io, 21
- get\_wlRange
  - \_op, 31
- get\_wlRangeMin
  - \_op, 31
- HISTFILE
  - elemlist.cpp, 38
  - findncopy.cpp, 40
  - genrandspec.cpp, 42
  - marker.cpp, 44
  - shift.cpp, 47
- LOGFILE
  - elemlist.cpp, 38
  - findncopy.cpp, 40
  - genrandspec.cpp, 42
  - marker.cpp, 45
  - shift.cpp, 47
- log.h, 42
- marker.cpp, 43
  - HISTFILE, 44
  - LOGFILE, 45
- MaxFilepDir
  - genrandspec.cpp, 42
- msg
  - \_msg, 29
- msg.h, 45
- msgM
  - \_marker, 28
- notify
  - \_io, 21
- read
  - \_csv, 11
  - \_io, 21
- read\_dir
  - \_io, 21
- read\_fits
  - \_io, 21
- rebuild\_wlStep
  - \_op, 32
- remove\_zero
  - \_op, 32
- resize\_spectr
  - \_op, 32
- select
  - \_csv, 11
- select\_column
  - \_csv, 12
- select\_line
  - \_csv, 12
- set\_WaveScale
  - \_io, 22
- set\_colorline
  - \_marker, 27
- set\_column
  - \_csv, 12
- set\_data
  - \_csv, 14
  - \_op, 32
- set\_fileIn
  - \_io, 22
- set\_fileOut
  - \_io, 22
- set\_filename
  - \_csv, 14
- set\_filename\_out
  - \_csv, 15
- set\_header
  - \_csv, 15
- set\_output
  - \_marker, 27
- set\_separator
  - \_csv, 16
- set\_supp
  - \_marker, 27
- set\_verbose

---

- [\\_csv](#), [17](#)
- shift.cpp, [46](#)
  - [HISTFILE](#), [47](#)
  - [LOGFILE](#), [47](#)
- show
  - [\\_csv](#), [17](#)
- show\_data
  - [\\_io](#), [22](#)
- transform\_lin
  - [\\_csv](#), [18](#)
- waverage.cpp, [48](#)
- write
  - [\\_csv](#), [18](#)
  - [\\_io](#), [23](#)
  - [\\_op](#), [32](#)
- write\_mean
  - [\\_op](#), [32](#)