

spec\_tools

Generated by Doxygen 1.8.13

## Contents

<b>1</b>	<b>Todo List</b>	<b>1</b>
<b>2</b>	<b>Class Documentation</b>	<b>1</b>
2.1	<a href="#">_csv&lt;_T&gt; Class Template Reference</a>	1
2.1.1	<a href="#">Detailed Description</a>	4
2.1.2	<a href="#">Constructor &amp; Destructor Documentation</a>	4
2.1.3	<a href="#">Member Function Documentation</a>	6
2.2	<a href="#">_marker&lt;_T&gt; Class Template Reference</a>	19
2.2.1	<a href="#">Detailed Description</a>	21
2.2.2	<a href="#">Member Function Documentation</a>	21
2.2.3	<a href="#">Member Data Documentation</a>	22
2.3	<a href="#">_msg Class Reference</a>	23
2.3.1	<a href="#">Detailed Description</a>	24
2.3.2	<a href="#">Member Function Documentation</a>	24
2.4	<a href="#">_spectra Class Reference</a>	25
2.5	<a href="#">_marker&lt;_T&gt;::Line Struct Reference</a>	25
2.5.1	<a href="#">Detailed Description</a>	26
<b>3</b>	<b>File Documentation</b>	<b>26</b>
3.1	<a href="#">csv.h File Reference</a>	26
3.1.1	<a href="#">Detailed Description</a>	27
3.2	<a href="#">der_snr.cpp File Reference</a>	28
3.2.1	<a href="#">Detailed Description</a>	29
3.2.2	<a href="#">Macro Definition Documentation</a>	29
3.2.3	<a href="#">Function Documentation</a>	29
3.3	<a href="#">findncopy.cpp File Reference</a>	36
3.3.1	<a href="#">Detailed Description</a>	37
3.3.2	<a href="#">Macro Definition Documentation</a>	37

3.4	<a href="#">genrandspec.cpp File Reference</a>	38
3.4.1	<a href="#">Detailed Description</a>	38
3.4.2	<a href="#">Macro Definition Documentation</a>	39
3.5	<a href="#">marker.cpp File Reference</a>	39
3.5.1	<a href="#">Detailed Description</a>	40
3.5.2	<a href="#">Macro Definition Documentation</a>	40
3.6	<a href="#">msg.h File Reference</a>	41
3.6.1	<a href="#">Detailed Description</a>	41
3.7	<a href="#">shift.cpp File Reference</a>	42
3.7.1	<a href="#">Detailed Description</a>	43
3.7.2	<a href="#">Macro Definition Documentation</a>	43
3.7.3	<a href="#">Function Documentation</a>	43
	<a href="#">Index</a>	45

## 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: {0123456789eE+-, 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 & 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  $\mathbf{data}[i][j] > val$ .

#### 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  $\mathbf{data}[i][j \neq list] > val$ .



## 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  $\mathbf{data}[i][j] < val$ .

## 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  $\mathbf{data}[i][j \neq list] < val$ .

## 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()`

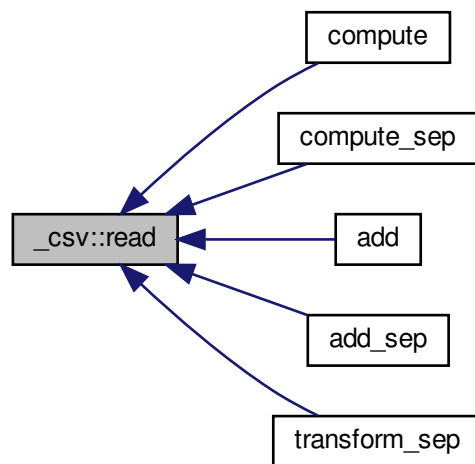
```
template<typename _T = double>
bool _csv<_T>::read ( )
```

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

Here is the caller graph for this function:

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.

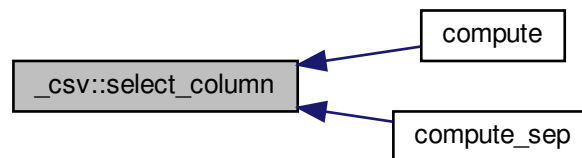
**Parameters**

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

**Returns**

`std::vector<_T>`

Here is the caller graph for this function:

**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>	<code>std::vector&lt;_T&gt; vCol</code>
<i>iCol</i>	Select a column

## Returns

true if all seems OK

## Parameters

<i>vRow</i>	<code>std::vector&lt;_T&gt; vRow</code>
<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>	<code>std::vector&lt;std::vector&lt;_T&gt; &gt; grid</code>
---------------	---

**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**

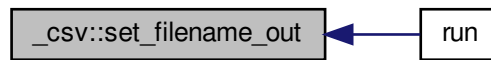
<i>sFilename</i>	The filename with extension or not.
------------------	-------------------------------------



**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", ',', ';;', ';` ...

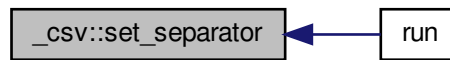
**Parameters**

<i>cSep</i>	The sep character: <code>"\t"</code> 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**

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

**Returns**

true if all seems OK

**2.1.3.26 set\_verbose()**

```

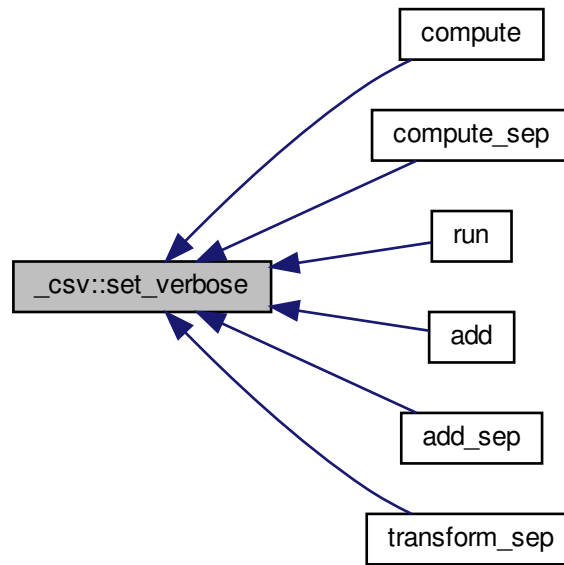
template<typename _T = double>
bool _csv< _T >::set_verbose (
    eVerbose evV )
  
```

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()**

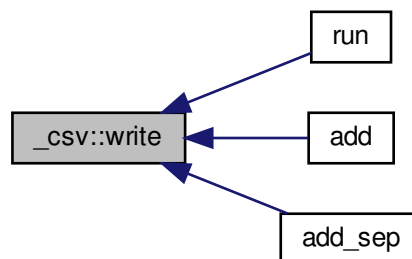
```
template<typename _T = double>
bool _csv< _T >::write ( )
```

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:

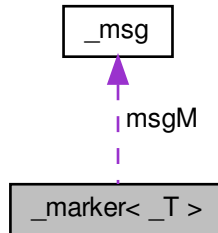
- [csv.h](#)

## 2.2 `_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\_ticklabelsiz** (int iSize)
- bool **set\_annotatesize** (int iSize)

*Set the font size of markers.*

- bool **set\_legendsize** (int iSize)
- bool **set\_continnumsize** (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.*

- 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.*

## Protected Attributes

- `_msg msgM`

## 2.2.1 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>&)`

## 2.2.2 Member Function Documentation

2.2.2.1 `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

2.2.2.2 `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`:  $[x_{min} \ x_{max}]$

2.2.2.3 `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"
---------------	--

## 2.2.2.4 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

## 2.2.2.5 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>	$x_{max}$

## 2.2.3 Member Data Documentation

## 2.2.3.1 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`

## 2.3 `_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 arbitratry 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 arbitratry 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 arbitratry 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.*

### 2.3.1 Detailed Description

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

### 2.3.2 Member Function Documentation

#### 2.3.2.1 msg()

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

Send a message...

##### Parameters

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

Here is the call graph for this function:



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

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

2.4 `_spectra` Class Reference

## Public Member Functions

- `_spectra` (const `_spectra` &other)
- `_spectra` & `operator=` (const `_spectra` &other)
- bool `operator==` (const `_spectra` &other) const
- bool `operator!=` (const `_spectra` &other) const

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

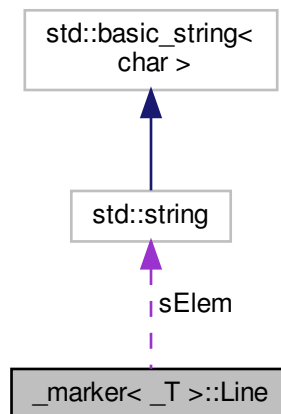
- `spectra.h`

2.5 `_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**

### 2.5.1 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

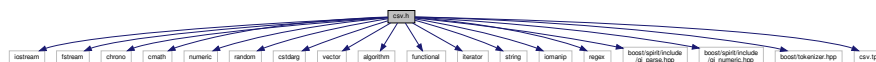
## 3 File Documentation

### 3.1 csv.h File Reference

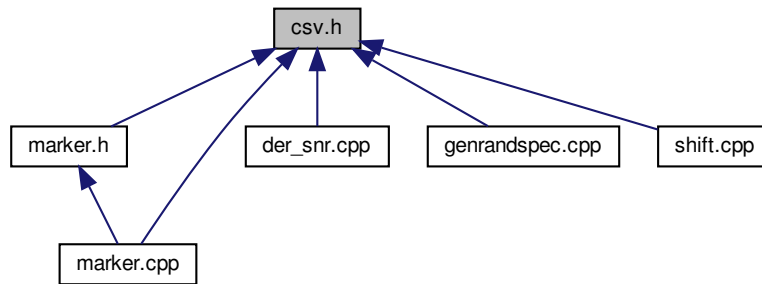
A basic class for csv manipulation.

```
#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



### 3.2.1 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

### 3.2.2 Macro Definition Documentation

#### 3.2.2.1 HISTFILE

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

Define the default histfile (shared)

#### 3.2.2.2 LOGFILE

```
#define LOGFILE ".der_snr.log"
```

Define the default logfile

### 3.2.3 Function Documentation

#### 3.2.3.1 compute()

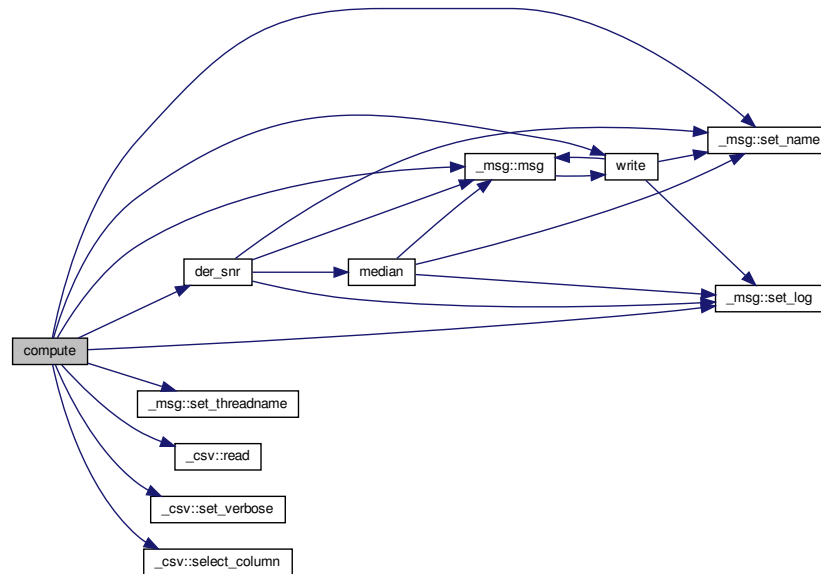
```
void compute (
    const std::vector< std::string > & list,
    const std::string & sOutput )
```

Compute S/N for all the string in the vector of strings. Default sep is tab. Used in the multithreaded mode.

## Parameters

<i>list</i>	list of files
<i>sOutput</i>	output filename

Here is the call graph for this function:



## 3.2.3.2 compute\_sep()

```

void compute_sep (
    const std::vector< std::string > & list,
    const std::string & sOutput,
    const char & cSep )

```

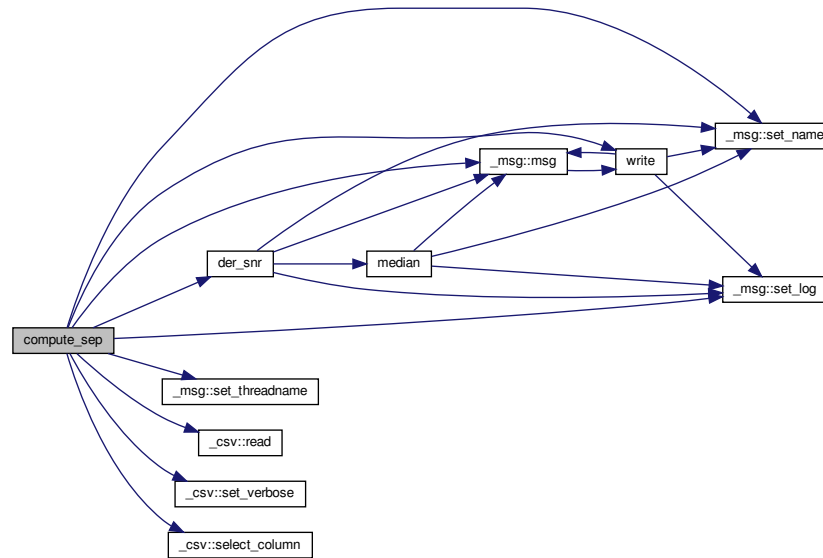
Compute S/N for all the string in the vector of strings. Used in the multithreaded mode.

## Parameters

<i>list</i>	list of files
<i>sOutput</i>	output filename
<i>cSep</i>	char separator



Here is the call graph for this function:



### 3.2.3.3 der\_snr()

```
float der_snr (
    const std::vector< float > & vFlux )
```

Compute the S/N with der\_snr method.

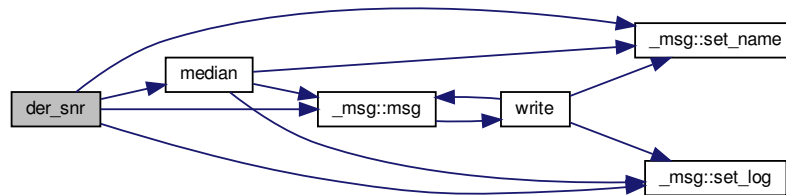
#### Parameters

<i>vFlux</i>	flux vector
--------------	-------------

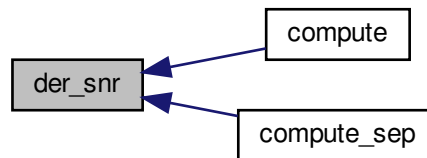
**Returns**

-1 if error happens

Here is the call graph for this function:



Here is the caller graph for this function:

**3.2.3.4 median()**

```
float median (
    const std::vector< float > & vFlux )
```

Simple computation of the median.

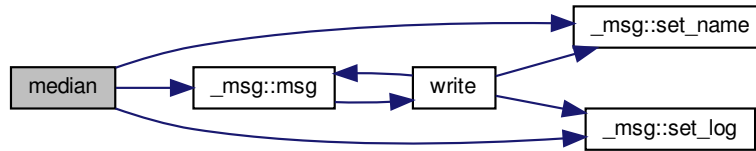
**Parameters**

<i>vFlux</i>	flux vector
--------------	-------------

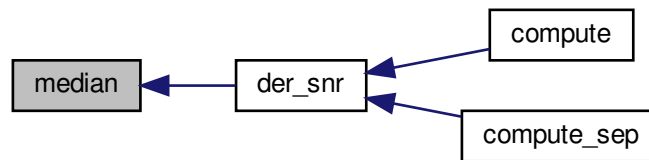
**Returns**

0 if error happens

Here is the call graph for this function:



Here is the caller graph for this function:



### 3.2.3.5 merge()

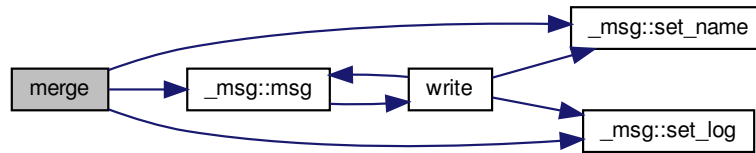
```
bool merge (
    const std::string & sPattern )
```

Merge files from threads following a filename pattern, i.e. the given output name.

#### Parameters

<i>sPattern</i>	basename without ext
-----------------	----------------------

Here is the call graph for this function:



### 3.2.3.6 write() [1/2]

```

bool write (
    std::vector< std::string > vsResults,
    const std::string & sOutput )

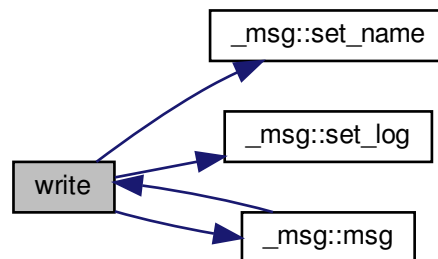
```

Write on disk results with the default sep.

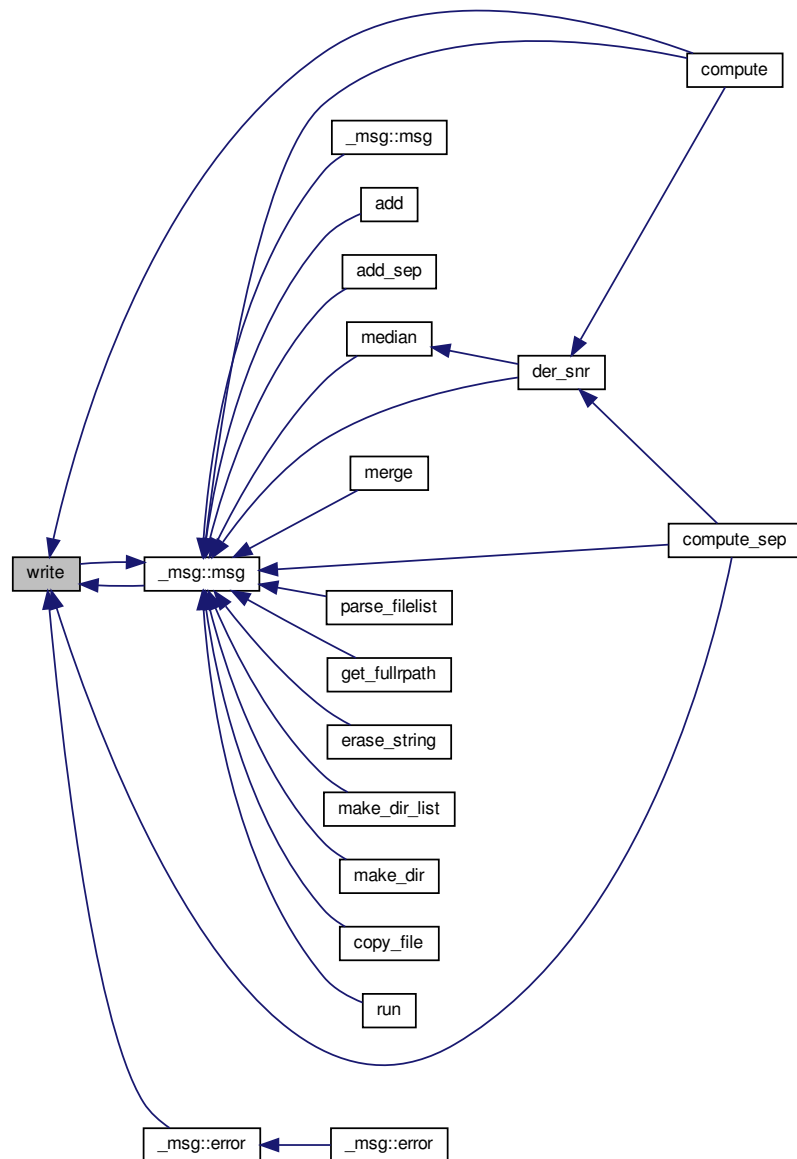
#### Parameters

<i>vsResults</i>	data to write
<i>sOutput</i>	output filename

Here is the call graph for this function:



Here is the caller graph for this function:



### 3.2.3.7 write() [2/2]

```

bool write (
    std::vector< std::string > vsResults,
    const std::string & sOutput,
    const char & cSep )

```

Write on disk results.

## Parameters

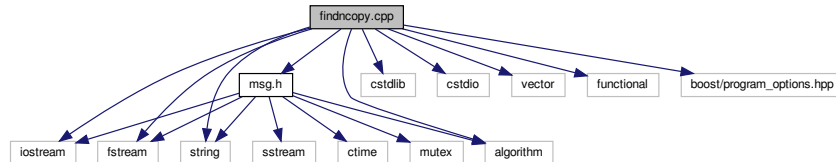
<i>vsResults</i>	data to write
<i>sOutput</i>	output filename
<i>cSep</i>	char separator

## 3.3 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 dependency graph for findncopy.cpp:



## Macros

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

## Functions

- `std::vector< std::string > parse_filelist (std::fstream &flux)`  
*Create a vector of strings from the filelist.*
- `std::vector< std::string > get_fullrpath (std::vector< std::string > &vsFilelist, const fs::path &fspPidir)`  
*Get the full relative path of all file.*
- `std::vector< std::string > get_fullrpath (std::vector< std::string > &vsFilelist, const fs::path &fspPidir, const std::string &sExclude)`  
*Get the full relative path of all file and exclude a string in paths.*
- `void erase_string (std::vector< std::string > &vsFullrpath, const std::string &sToerase)`

*Erase a string pattern in the path list.*

- `std::vector< std::string > make_dir_list` (const fs::path &fspPath, const std::string &sDirbase)

*Make a list of the folder structure.*

- `void make_dir` (const std::vector< std::string > &vsBaserpath, const std::string &sOfolder)

*Recreate the folder structure.*

- `void copy_file` (std::vector< std::string > &vsFullrpath, const std::string &sOfolder, const std::string &sIfolder)

*Copy the found files.*

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

### 3.3.1 Detailed Description

Copy files from a list in a new folder.

#### Author

Audric Lemonnier

#### Version

0.1

#### Date

09/03/2020

### 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 ".findncopy.log"
```

Define the default logfile





## Version

0.4

## Date

18/04/2020

## 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 ".genrandspec.log"
```

Define the default logfile

## 3.4.2.3 MaxFileDir

```
#define MaxFileDir 10
```

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

MaxFileDir

## 3.5 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 <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 <csv.h>
```

Include dependency graph for marker.cpp:



## Macros

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

## Functions

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

### 3.5.1 Detailed Description

Highlight lines on spectrum.

## Author

Audric Lemonnier

## Version

0.4

## Date

21/04/2020

### 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 ".marker.log"
```

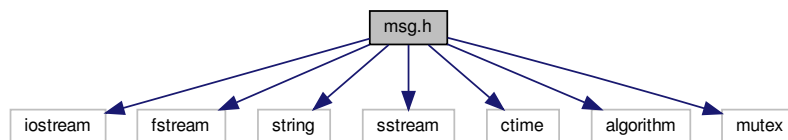
Define the default logfile

### 3.6 msg.h File Reference

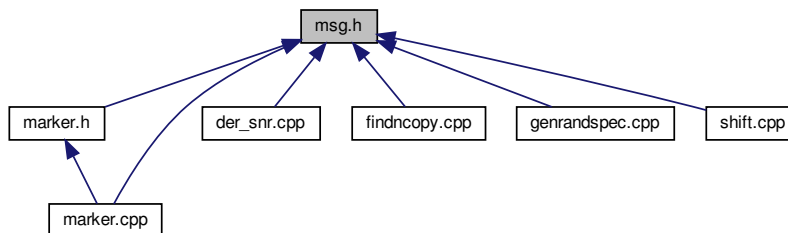
A class to print and write message.

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

Include 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...*

#### 3.6.1 Detailed Description

A class to print and write message.



### 3.7.1 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

### 3.7.2 Macro Definition Documentation

#### 3.7.2.1 HISTFILE

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

Define the default histfile (shared)

#### 3.7.2.2 LOGFILE

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

Define the default logfile

### 3.7.3 Function Documentation

#### 3.7.3.1 transform\_sep()

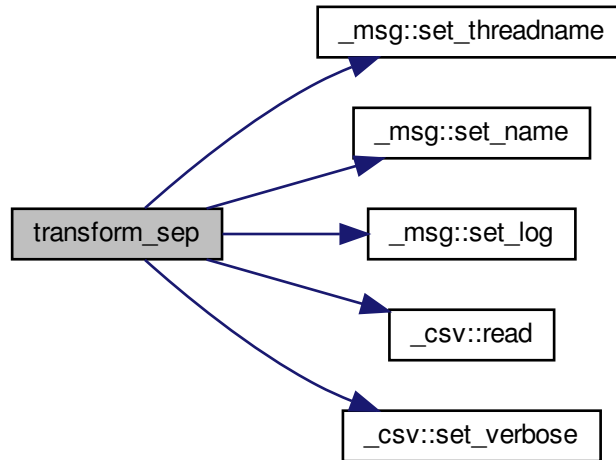
```
void transform_sep (
    const std::vector< std::string > & vsList,
    char cSep,
    float fVr )
```

Correct the radial velocity effect on spectra. Perform a linear transformation.

## Parameters

<i>fVr</i>	Radial Velocity
------------	-----------------

Here is the call graph for this function:



## Index

`_csv`

`_csv`, 4–6

`apply_max_threshold`, 6

`apply_min_threshold`, 7

`check_dim`, 7

`empty`, 8

`get_data`, 8

`get_data_size_i`, 8

`get_data_size_j`, 8

`get_filename`, 9

`get_filename_out`, 9

`get_header`, 9

`get_header_size`, 9

`get_separator`, 10

`read`, 10

`select`, 11

`select_column`, 12

`select_line`, 12

`set_column`, 13

`set_data`, 13

`set_filename`, 14

`set_filename_out`, 14

`set_header`, 15

`set_separator`, 15, 16

`set_verbose`, 16

`show`, 17

`transform_lin`, 18

`write`, 18

`_csv<_T>`, 1

`_marker`

`get_figsize`, 21

`get_supp`, 21

`msgM`, 22

`set_colorline`, 21

`set_output`, 22

`set_supp`, 22

`_marker<_T>`, 19

`_marker<_T>::Line`, 25

`_msg`, 23

`msg`, 24

`_spectra`, 25

`apply_max_threshold`

`_csv`, 6

`apply_min_threshold`

`_csv`, 7

`check_dim`

`_csv`, 7

`compute`

`der_snr.cpp`, 29

`compute_sep`

`der_snr.cpp`, 30

`csv.h`, 26

`der_snr`

`der_snr.cpp`, 31

`der_snr.cpp`, 28

`compute`, 29

`compute_sep`, 30

`der_snr`, 31

`HISTFILE`, 29

`LOGFILE`, 29

`median`, 32

`merge`, 33

`write`, 34, 35

`empty`

`_csv`, 8

`findncopy.cpp`, 36

`HISTFILE`, 37

`LOGFILE`, 37

`genrandspec.cpp`, 38

`HISTFILE`, 39

`LOGFILE`, 39

`MaxFileDir`, 39

`get_data`

`_csv`, 8

`get_data_size_i`

`_csv`, 8

`get_data_size_j`

`_csv`, 8

`get_figsize`

`_marker`, 21

`get_filename`

`_csv`, 9

`get_filename_out`

`_csv`, 9

`get_header`

`_csv`, 9

`get_header_size`

`_csv`, 9

`get_separator`

`_csv`, 10

`get_supp`

`_marker`, 21

`HISTFILE`

`der_snr.cpp`, 29

`findncopy.cpp`, 37

`genrandspec.cpp`, 39

`marker.cpp`, 40

- shift.cpp, [43](#)
- LOGFILE
  - der\_snr.cpp, [29](#)
  - findncopy.cpp, [37](#)
  - genrandspec.cpp, [39](#)
  - marker.cpp, [40](#)
  - shift.cpp, [43](#)
- marker.cpp, [39](#)
  - HISTFILE, [40](#)
  - LOGFILE, [40](#)
- MaxFilepDir
  - genrandspec.cpp, [39](#)
- median
  - der\_snr.cpp, [32](#)
- merge
  - der\_snr.cpp, [33](#)
- msg
  - \_msg, [24](#)
- msg.h, [41](#)
- msgM
  - \_marker, [22](#)
- read
  - \_csv, [10](#)
- select
  - \_csv, [11](#)
- select\_column
  - \_csv, [12](#)
- select\_line
  - \_csv, [12](#)
- set\_colorline
  - \_marker, [21](#)
- set\_column
  - \_csv, [13](#)
- set\_data
  - \_csv, [13](#)
- set\_filename
  - \_csv, [14](#)
- set\_filename\_out
  - \_csv, [14](#)
- set\_header
  - \_csv, [15](#)
- set\_output
  - \_marker, [22](#)
- set\_separator
  - \_csv, [15](#), [16](#)
- set\_supp
  - \_marker, [22](#)
- set\_verbose
  - \_csv, [16](#)
- shift.cpp, [42](#)
  - HISTFILE, [43](#)
  - LOGFILE, [43](#)
  - transform\_sep, [43](#)
- show
  - \_csv, [17](#)
- transform\_lin
  - \_csv, [18](#)
- transform\_sep
  - shift.cpp, [43](#)
- write
  - \_csv, [18](#)
  - der\_snr.cpp, [34](#), [35](#)