spec_tools

Generated by Doxygen 1.8.17

1 Todo List

Member main (int argc, char **argv)

Parsing command line to get folder name and csv separator, for example.

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 ()

  This is the 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)

  This is the constructor with two parameters such as the name of the working file and the separator character as usual with csv.
- _csv (const std::vector< std::vector< _T > > &vvData)

  This is the constructor fed with external data.
- _csv (const std::vector< std::string > &vsHeader, const std::vector< std::vector< _T > > &vvData)

  This is the constructor fed with external header and data.
- _csv (const std::vector< std::string > &vsHeader, const std::vector< std::vector< _T > > &vvData, const char &cSep)

  This is the 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', ' ', ',', ';' ...
- 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 int get_header_size () const

  Get size of the header.
- const int get_data_size_i () const

  Get data line size.
- const int 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 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 $\mathbf{data}[i][j] > val$.
- bool apply_min_threshold (_T TVal)

  Delete i line from the grid where $\mathbf{data}[i][j] < val$.
- bool apply_max_threshold (_T TVal, int iCol)

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

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

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

  Delete data and header.
- **_csv** (_csv &other)
- _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

  Substract 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() [1/5] template<typename _T = double>

```
_csv< _T >::_csv ( )
```

This is the 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() [2/5] template<typename _T = double>

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

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

Constructor

Parameters

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

2.1.2.3 _csv() [3/5] template<typename _T = double>

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

This is the constructor fed with external data.

Parameters

vvData	The data

2.1.2.4 _csv() [4/5] template<typename _T = double>

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

This is the constructor fed with external header and data.

Parameters

vsHeader	The vector of column name
vvData	The data

2.1.2.5 _csv() [5/5] template<typename _T = double>

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

This is the constructor fed with external header and data.

Parameters

vsHeader	The vector of column name
vvData	The data
cSep	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

TVal	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

TVal	The max threshold
iCol	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

TVal	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

TVal	The min threshold
iCol	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 int _csv< _T >::get_data_size_i ( ) const
```

Get data line size.

Returns

int

2.1.3.9 get_data_size_j() template<typename _T = double>

```
const int _csv< _T >::get_data_size_j ( ) const
```

Get data column size.

Returns

int

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 int _csv< _T >::get_header_size ( ) const
```

Get size of the header.

Returns

int

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>

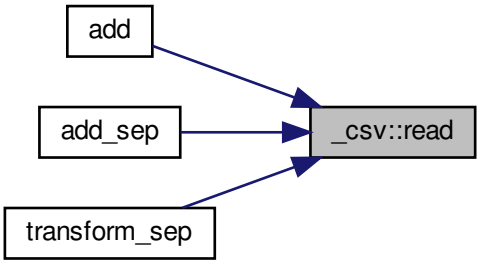
```
bool _csv< _T >::read ( )
```

Read the content of the file given to the constructor using boost. It detects the header and data consistency with digit sequence: {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.

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

iLine_min	upper line i_{min}
iLine_max	lower line i_{max}
iCol_min	left column j_{min}
iCol_max	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

iCol	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

iLine	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

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

Returns

true if all seems OK

Parameters

vRow	std::vector<_T> vRow
iRow	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

vvData	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

sFilename	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

sFilename	The filename with extension or not.

Returns

true if all seems OK

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

vsHeader	string vector

Returns

true if all seems OK

2.1.3.24 set_separator() template<typename _T = double>

```
bool _csv< _T >::set_separator (
            const char & cSep )
```

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

Parameters

cSep	The sep character: '\t' for tabulation

Returns

true if all seems OK

2.1.3.25 set_verbose() `template<typename _T = double>`

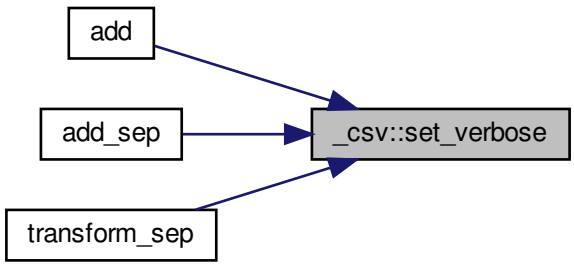
```
bool _csv< _T >::set_verbose (
            eVerbose evV )
```

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

Parameters

evV	eVerbose::DEBUG for verbose mode and eVerbose::QUIET to keep quiet

Here is the caller graph for this function:



2.1.3.26 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

2.1.3.27 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

iLine_stop	The line number where stop the display

Returns

true if all seems OK

2.1.3.28 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

2.1.3.29 write() template<typename _T = double>

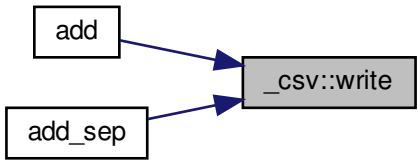
```
bool _csv< _T >::write ( )
```

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 _msg Class Reference

A class that sends string to std output...

`#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)
- void msg (const std::string &sMsg) const

  Send a message with eMsg::MID as default.
- void msg (eMsg emType, const std::string &sMsg) const

  Send a message...
- void error (const std::string &sMsg) const

  Send an error message...
- template<typename ... Args>
  void msg (const Args &...args) const

A variadic formatter method that indeed sends arbitratry number of variable to the std output... with eMsg::MID as default.

- template<typename ... Args>
  void msg (eMsg emType, const Args &...args) const

  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>
  void error (const Args &...args) const

  A variadic formatter method that indeed sends arbitratry number of variable to the std error output... with eMsg::ERROR as default.
- void set_name (const std::string sName)

  Set the name of the main instance.
- void set_threadname (const std::string sName)

  Set the name of threads.

2.2.1 Detailed Description

A class that sends string to std output...

2.2.2 Member Function Documentation

2.2.2.1 msg()

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

Send a message...

Parameters

emType	See enum eMsg::

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

- msg.h
- msg.cpp

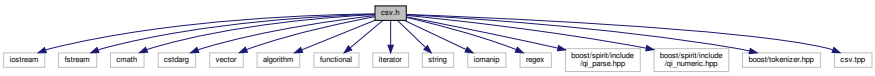
3 File Documentation

3.1 csv.h File Reference

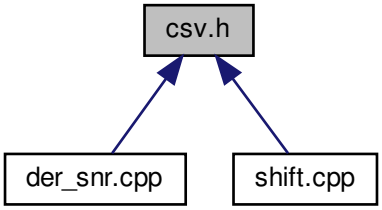
A basic class for csv manipulation.

```
#include <iostream>
#include <fstream>
#include <cmath>
#include <cstdarg>
#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.tpp"
```

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.

Macros

- #define PARALLEL_EXEC

3.1.1 Detailed Description

A basic class for csv manipulation.

Author

Audric Lemonnier

Version

0.7

Date

13/03/2020

3.1.2 Macro Definition Documentation

3.1.2.1 PARALLEL_EXEC `#define PARALLEL_EXEC`

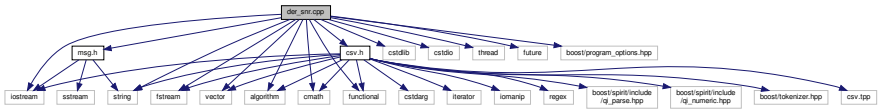
If c++17 and TBB available

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 This code is multi-threaded or not if not available.

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

Include dependency graph for der_snr.cpp:



Functions

- void **compute** (const std::vector< std::string > &list, const std::string &sOutput)
- 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.
- bool merge (const std::string &sPattern)

  Merge files from threads following a filename pattern, i.e. the given output name.
- bool **write** (std::vector< std::string > vsResults, const std::string &sOutput)
- bool write (std::vector< std::string > vsResults, const std::string &sOutput, const char &cSep)

  Write on disk results.
- float der_snr (const std::vector< float > &vFlux)

  Compute the S/N with der_snr method.
- double **der_snr** (const std::vector< double > &vFlux)
- float median (const std::vector< float > &vFlux)

  Simple computation of the median.
- double **median** (const std::vector< double > &vFlux)
- int main (int argc, char **argv)

  This code removes zeros and negative values in csv located in "./data". The maximum of thread has been used to accelerate code.

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 This code is multi-threaded or not if not available.

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

0.1

Date

16/03/2020

3.2.2 Function Documentation

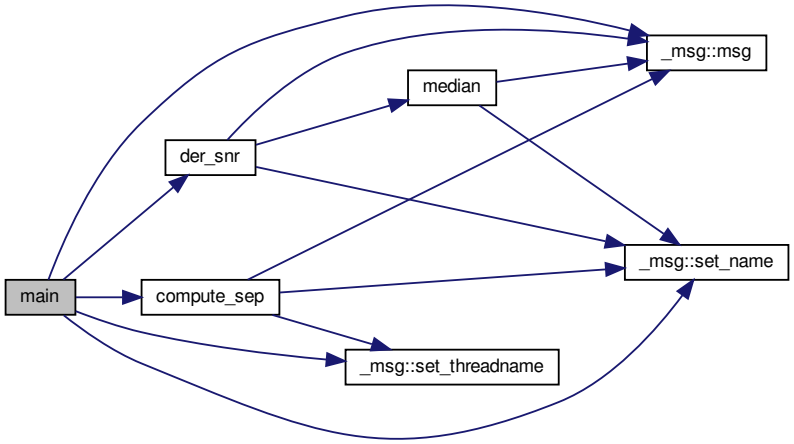
3.2.2.1 main()

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

This code removes zeros and negative values in csv located in "./data". The maximum of thread has been used to accelerate code.

Todo Parsing command line to get folder name and csv separator, for example.

Here is the call graph for this function:



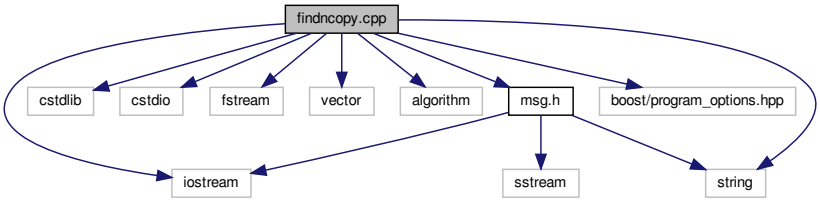
3.3 findncopy.cpp File Reference

Copy files from a list in a new folder.

```
#include <iostream>
#include <cstdlib>
#include <cstdio>
#include <fstream>
#include <vector>
#include <algorithm>
#include <string>
#include <boost/program_options.hpp>
```

```
#include "msg.h"
```

Include dependency graph for findncopy.cpp:



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

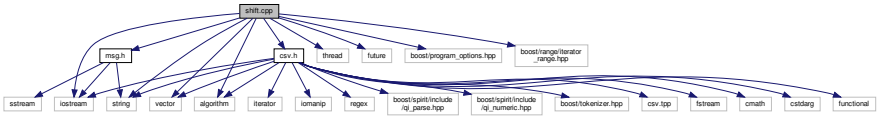
09/03/2020

3.4 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 <boost/program_options.hpp>
#include <boost/range/iterator_range.hpp>
#include "csv.h"
#include "msg.h"
```

Include dependency graph for shift.cpp:



Macros

- #define **CLIGHT** 299792458

Functions

- void add (const std::vector< std::string > &vsList, float fWavelength)

  Add the defined wavelength to the first column of spectra. Default sep is '\t'.
- void add_sep (const std::vector< std::string > &vsList, char cSep, float fWavelength)

  Add the defined wavelength to the first column of spectra.
- void transform_sep (const std::vector< std::string > &vsList, char cSep, float fVr)

  Correct the radial velocity effect on spectra. Perform a linear transformation.
- int **main** (int argc, char **argv)

3.4.1 Detailed Description

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

Author

Audric Lemonnier

Version

0.2

Date

16/03/2020

3.4.2 Function Documentation

3.4.2.1 transform_sep() `void transform_sep (`

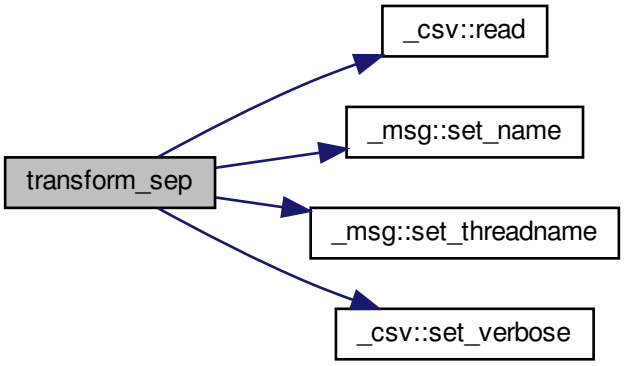
```
const std::vector< std::string > & vsList,
char cSep,
float fVr )
```

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

Parameters

fVr	Radial Velocity

Here is the call graph for this function:



Index