

fedithead

Richard J. Mathar

Generated by Doxygen 1.8.2

Wed Oct 30 2013 09:53:07

Contents

1	Hierarchical Index	2
1.1	Class Hierarchy	2
2	Class Index	2
2.1	Class List	2
3	File Index	2
3.1	File List	2
4	Class Documentation	2
4.1	csv2Fits Class Reference	2
4.1.1	Detailed Description	3
4.1.2	Constructor & Destructor Documentation	3
4.1.3	Member Function Documentation	3
4.1.4	Member Data Documentation	4
4.2	FEditHead Class Reference	5
4.2.1	Detailed Description	6
4.2.2	Constructor & Destructor Documentation	6
4.2.3	Member Function Documentation	8
4.2.4	Member Data Documentation	12
5	File Documentation	13
5.1	fedithread/src/config.h File Reference	13
5.1.1	Macro Definition Documentation	13
5.2	fedithread/src/csv2Fits.cxx File Reference	14
5.2.1	Detailed Description	14
5.2.2	Macro Definition Documentation	14
5.2.3	Function Documentation	14
5.3	fedithread/src/csv2Fits.h File Reference	15
5.4	fedithread/src/FEditHead.cxx File Reference	15
5.5	fedithread/src/fedithread.cxx File Reference	15
5.5.1	Function Documentation	15
5.6	fedithread/src/FEditHead.h File Reference	16
6	Example Documentation	16
6.1	/home/mathar/work/geirs/fedithread/src/csv2Fits.cxx	17
	Index	18

1 Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

csv2Fits	2
FITS	
FEditHead	5

2 Class Index

2.1 Class List

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

csv2Fits	
Csv2Fits converts a file with comma-separated values to a binary FITS table	2
FEditHead	
The class edits header keywords FITS headers	5

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

fedithead/src/ config.h	13
fedithead/src/ csv2Fits.cxx	
Csv2Fits translates an ASCII file in the CSV format to a FITS file with a binary table	14
fedithead/src/ csv2Fits.h	15
fedithead/src/ fedithead.cxx	15
fedithead/src/ FEditHead.cxx	15
fedithead/src/ FEditHead.h	16

4 Class Documentation

4.1 csv2Fits Class Reference

csv2Fits converts a file with comma-separated values to a binary FITS table.

```
#include <csv2Fits.h>
```

Public Member Functions

- **csv2Fits** (const std::string infile, const char fsep=',')

Constructor.

- void `cnvrt` (const std::string *typs*) const
Scan all lines of the input file and generate the FITS file.

Public Attributes

- std::string `csvFil`
The name of the input file.
- char `fsep`
The field separator .

Protected Member Functions

- std::vector< std::string > `strtok` (const std::string *lin*) const
Decompose the CSV line by splitting it along commas.
- std::vector< int > `colWidths` (int &*lineCnt*) const
Determine the number of characters in the longest string in each column.

4.1.1 Detailed Description

`csv2Fits` converts a file with comma-separated values to a binary FITS table.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 csv2Fits::csv2Fits (const std::string *infile*, const char *f* = ' , ')

Constructor.

Parameters

<code>in</code>	<i>infile</i>	The name of the ASCII file in CSV format.
<code>in</code>	<i>f</i>	The field separator. This is a single character (the comma by default) which separates adjacent fields in the input file.

Author

Richard J. Mathar

Since

2013-04-30

4.1.3 Member Function Documentation

4.1.3.1 void csv2Fits::cnvrt (const std::string *typs*) const

Scan all lines of the input file and generate the FITS file.

Parameters

<code>in</code>	<i>typs</i>	A type string with letters a, d, f, b, i, k, and j indicating the FITS type of each column (each field in the CSV)
-----------------	-------------	--

Author

R. J. Mathar

Since

2013-04-30

4.1.3.2 `vector< string > csv2Fits::strtok (const std::string lin) const` [protected]

Decompose the CSV line by splitting it along commas.

Parameters

<i>in</i>	<i>lin</i>	The input line with any number of field separators.
-----------	------------	---

Returns

The first field in the 0-th element, the 2nd field in the 1-st element etc.

4.1.3.3 `vector< int > csv2Fits::colWidths (int & lineCt) const` [protected]

Determine the number of characters in the longest string in each column.

Parameters

<i>out</i>	<i>lineCt</i>	On return this contains the number of lines in the input file. The header line with the prospective names of the columns is not counted.
------------	---------------	--

Returns

A vector of maximum character widths for columns 0, 1, 2, ... etc This summarizes the widths of the associated column along all lines of the input file.

Author

R. J. Mathar

Since

2013-04-30

4.1.4 Member Data Documentation

4.1.4.1 `std::string csv2Fits::csvFil`

The name of the input file.

4.1.4.2 `char csv2Fits::fsep`

The field separator .

By default this is a comma. For many astronomical databases, the standard is the vertical bar (pipe symbol).

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

- [fedithead/src/csv2Fits.h](#)
- [fedithead/src/csv2Fits.cxx](#)

4.2 FEditHead Class Reference

The class edits header keywords FITS headers.

```
#include <FEditHead.h>
```

Inherits FITS.

Public Member Functions

- [FEditHead](#) (char *fitsname, char *fitstpl)
Constructor given a FITS file and a template ASCII file.
- [FEditHead](#) (char *fitsname, const vector< string > fitstpl)
Constructor given a FITS file and a list of template ASCII files.
- [FEditHead](#) (const char *gstream, int glines, int bitpix, int naxis, long *naxes, char fitstpl[], const char *fname)
- [FEditHead](#) (const char *gstream, int glines, int bitpix, int naxis, long *naxes, const vector< string > fitstpl, const char *fname)
Constructor given an old-fashioned GEIRS header stream and template ASCII files.
- [FEditHead](#) (const char *gstream, char *fname, int glines=-1)
Constructor given an old-fashioned GEIRS header stream.
- [~FEditHead](#) ()
Destructor.
- void [exec](#) (const bool verbose)
Execute the modifications of the template file in the header.
- char * [allLines](#) (int maxLines)

Public Attributes

- String [cfgFil](#)
Name of the configuration file with header template keywords.
- char * [buffer](#)
The concatenation of all input lines, trimmed/expanded to a multiple of 80 characters per line, one final \0.

Protected Member Functions

- vector< string > [scanTpl](#) (const char *fitstpl, const bool raw80=false) const
Scan the template file.
- vector< string > [scanTpl](#) (const string fitstpl, const bool raw80=false) const
Scan the template file.
- vector< string > [scanTpl](#) (const vector< string > fitstpl, const bool raw80=false) const
Scan a list of template files.
- void [readStream](#) (const char *gstream, int glines, const bool restr=false)

Static Protected Member Functions

- static string [trimws](#) (const string &instring)
Strip leading and trailing white space from a string.
- static char * [parse_template](#) (const string &instring)
Reformat an input string into a standard FITS header line.
- static int [templateAnal](#) (char *card, char keyname[], char keyval[], char cmt[], char dtyp[])
Analyse a header card.
- static vector< string > [splitColon](#) (const string &instring)

Split a string using colons or exclamation marks as delimiters.

- static vector< string > [splitColon](#) (const string &instring, const string &delim)

Split a string using a specific delimiting string.

- static bool [addHduCard](#) (HDU &hdu, char *card, const bool verbose=false)

Protected Attributes

- vector< string > [tplLines](#)

The assembly of all template keyword ASCII lines.

4.2.1 Detailed Description

The class edits header keywords FITS headers.

The object is created specifying the name of an existing FITS file and the name of a template file which contains instructions to add, delete or modify keywords. A method then allows to run through these instructions and apply them to a specified primary or extension header.

The main intend of this program is to eliminate the limitations known for the fmodhead, fthedit and eso2fits programs. The program `fedithead` does work with hierarchical keywords, and it even allows to replace hierarchical keywords by simple keywords and vice versa. Groups of keywords that are parseable by a `regex(7)` specification may also be removed or replaced with a syntax known from the `sed(1)`, calculated with the aid of the `boost-regex` package. Moving keywords between HDU's or re-calculation of keywords as with ESO's `FTU` is not supported.

Since

2012-10-18

Author

Richard J. Mathar

4.2.2 Constructor & Destructor Documentation

4.2.2.1 FEditHead::FEditHead (char * *fitsname*, char * *fitstpl*)

Constructor given a FITS file and a template ASCII file.

It attempts to open the FITS file and to scan the template file. Applying the template modifications to the FITS file is left to a subsequent call to [FEditHead::exec](#).

Parameters

in	<i>fitsname</i>	The name of the FITS file to be modified.
in	<i>fitstpl</i>	The name of the ASCII file with the header template lines.

Since

2012-10-18

Author

Richard J. Mathar

4.2.2.2 FEditHead::FEditHead (char * *fitsname*, const vector< string > *fitstpl*)

Constructor given a FITS file and a list of template ASCII files.

It attempts to open the FITS file and to scan all template files. Applying the template modifications to the FITS file is left to a subsequent call to [FEditHead::exec](#) .

Parameters

in	<i>fitsname</i>	The name of the FITS file to be modified.
in	<i>fitstpl</i>	The name of the ASCII files with the header template lines.

Since

2013-10-30

Author

Richard J. Mathar

4.2.2.3 `FEditHead::FEditHead (const char * gstream, int glines, int bitpix, int naxis, long * naxes, char fitstpl[], const char * fname)`

4.2.2.4 `FEditHead::FEditHead (const char * gstream, int glines, int bitpix, int naxis, long * naxes, const vector< string > fitstpl, const char * fnam)`

Constructor given an old-fashioned GEIRS header stream and template ASCII files.

It attempts to split the header stream in 80 bytes pieces and to scan all template files. Applying the template modifications to the FITS file is left to a subsequent call to [FEditHead::exec](#) .

Parameters

in	<i>gStream</i>	The bytes in the GEIRS header. These are characterized by having the shape of non-terminated 80-byte lines.
in	<i>glines</i>	The number of 80-byte lines in <i>gstream</i> .
in	<i>fitstpl</i>	The names of the ASCII files with the header template lines.
in	<i>fnam</i>	The name of the temporary FITS file to be created.

Since

2013-07-12

Author

Richard J. Mathar

4.2.2.5 `FEditHead::FEditHead (const char * gstream, char * fnam, int glines = -1)`

Constructor given an old-fashioned GEIRS header stream.

It attempts to split the header stream in 80 bytes pieces. Applying the template modifications to the FITS file is left to a subsequent call to [FEditHead::exec](#) .

Parameters

in	<i>gStream</i>	The bytes in the GEIRS header. These are characterized by having the shape of non-terminated 80-byte lines.
in	<i>fnam</i>	The name of the temporary FITS file to be created.

Since

2013-07-12

Author

Richard J. Mathar

4.2.2.6 FEditHead::~~FEditHead ()

Destructor.

De-allocates the buffer with the sequential concatenation of all the 80-bytes header lines. Removes the FITS file if the flag keepDtor was set to false.

4.2.3 Member Function Documentation

4.2.3.1 void FEditHead::exec (const bool *verbose*)

Execute the modifications of the template file in the header.

Request of adding or modifying keywords are executed in the order of request (in the order of the template file) and the new keywords are appended in that order. This basically means that header cards are not sorted in any particular way to be 'nice.'

Parameters

<i>in</i>	<i>verbose</i>	If true, comment on each substitution, deletion etc.
-----------	----------------	--

Since

2012-10-18

Author

Richard J. Mathar

Do nothing if the requested list of keywords is empty.

Consider in the outer loop each applicable template line.

4.2.3.2 char * FEditHead::allLines (int *maxLines*)

Dump the lines into the local buffer and return its location.

Parameters

<i>in</i>	<i>maxLines</i>	A cutoff for the number of lines to be returned. A negative number is interpreted as that no upper limit is set.
-----------	-----------------	--

Returns

A pointer to a locally allocated buffer. This will be removed once the instance of the class goes out of scope.

Since

2013-07-12

4.2.3.3 `vector< string > FEditHead::scanTpl (const char * fitstpl, const bool raw80 = false) const` [protected]

Scan the template file.

Parameters

<i>in</i>	<i>fitstpl</i>	<p>The name of the template ASCII file. The file contains six types of lines:</p> <ul style="list-style-type: none"> • Empty lines and lines starting immediately with a hash (#) are ignored (interpreted as lines that comment the template). • Lines starting with a dash (-) contain a keyword (right after the dash with optional white space in between) that is to be removed from the FITS header. If the keyword starts with <code>HIERARCH</code>, it may consist of blank-separated components, or otherwise it needs to follow the standard 8-letter limitation. • Lines starting with a delimiter, either the colon or the exclamation mark, contain two keywords, separated by another delimiter of the same type. The keyword between the first pair of delimiters is replaced by the keyword in the second pair. This syntax is an extension to the standard cfitsio/fitsio template syntax because this allows to delimit multi-strings as seen with the hierarchical keywords, and on which fedithd will choke if used with its implied rule of the (-)-syntax. • Other lines are interpreted as a keyword, value and comment in the standard FITS style. An equal sign separates keyword and value, and the comment following after a slash (/). Hierarchical keywords are allowed. Conversion to upper case letters occurs on the fly. Comments in long input lines are chopped. Note that if the corresponding keyword is already present in the header, it will be replaced by this new one.
-----------	----------------	--

- Lines starting with at least 8 blanks are inserted as COMMENT cards into the FITS header, effectively replacing these blanks by the COMMENT keyword. Long lines will be wrapped into the next COMMENT card to fit into the 80-bytes limits of the standard.

Parameters

<i>in</i>	<i>raw80</i>	If true, the file <i>fitstpl</i> contains lines of 80 bytes without line feeds
-----------	--------------	--

Returns

The non-comment lines of the file. The order of the input file (top to bottom) is preserved.

Since

2012-10-18

Author

Richard J. Mathar

4.2.3.4 `vector< string > FEditHead::scanTpl (const string fitstpl, const bool raw80 = false) const` [protected]

Scan the template file.

Parameters

<i>in</i>	<i>fitstpl</i>	The name of the template ASCII file. The contents is the same as documented in the signature with the character pointer.
<i>in</i>	<i>raw80</i>	If true, the file <i>fitstpl</i> contains lines of 80 bytes without line feeds

Returns

The non-comment lines of the file. The order of the input file (top to bottom) is preserved.

Since

2013-10-17

Author

Richard J. Mathar

4.2.3.5 `vector< string > FEditHead::scanTpl (const vector< string > fitstpl, const bool raw80 = false) const`
`[protected]`

Scan a list of template files.

The function implies a loop of scans over the template files in the order of the vector. This means files later (with higher index) in the vector may override the specs of lines in files earlier in the vector.

Parameters

<i>in</i>	<i>raw80</i>	If true, the file <i>fitstpl</i> contains lines of 80 bytes without line feeds
-----------	--------------	--

Returns

The non-comment lines of the file. The order of the input file (top to bottom) is preserved.

Since

2013-10-17

Author

Richard J. Mathar

4.2.3.6 `string FEditHead::trimws (const string & instring)` `[static], [protected]`

Strip leading and trailing white space from a string.

Parameters

<i>in</i>	<i>instring</i>	The initial string from which nonprintable leading and trailing characters or white space (tabs, blanks) ought to be removed.
-----------	-----------------	---

Returns

The string with leading and trailing blanks, tabs etc removed.

4.2.3.7 `char * FEditHead::parse_template (const string & instring)` `[static], [protected]`

Reformat an input string into a standard FITS header line.

This is an intermediate invocation of the `fits_parse_template()` functionality of `cfitsio`.

Parameters

<i>in</i>	<i>instring</i>	
-----------	-----------------	--

Returns

A validated copy of the card line. This char array should be deleted by the caller after return.

4.2.3.8 `int FEditHead::templateAnal (char * card, char keyname[], char keyval[], char cmt[], char dtyp[])` `[static]`, `[protected]`

Analyse a header card.

The header card is decomposed into name, value, comment and type.

Parameters

in	<i>card</i>	The card to be analysed, typically up to 80 characters.
out	<i>keyname</i>	Keyword name in the card.
out	<i>keyval</i>	Keyword value in the card.
out	<i>cmt</i>	Comment after the slash in the card.
out	<i>dtyp</i>	Rough type (string, integer, float, boolean or complex)

Returns

The cfitsio error code. 0 if there was no error.

4.2.3.9 `vector< string > FEditHead::splitColon (const string & instring)` `[static]`, `[protected]`

Split a string using colons or exclamation marks as delimiters.

Parameters

in	<i>instring</i>	This input has the sequential format :.....:..... or !...!...!... Either the colon or the exclamation mark are the local delimiters. This means the first, second and third mark delimit the two strings to be defined. The third mark is optional. If missing, the entire residual part of the input defines the 2nd string.
----	-----------------	---

Returns

A vector of 2 strings. The first contains the characters between the first and the second mark, the second between the 2nd and the third mark.

Since

2012-10-18

2013-01-29 With exclamation marks working as delimiters

Author

Richard J. Mathar

4.2.3.10 `vector< string > FEditHead::splitColon (const string & instring, const string & delim)` `[static]`, `[protected]`

Split a string using a specific delimiting string.

Parameters

in	<i>instring</i>	This input has the sequential format <delimit>....<delimit>.....<delimit>... This means the first, second and third mark delimit the two strings to be defined. The third mark is optional. If missing, the entire residual part of the input defines the 2nd string.
in	<i>delim</i>	The delimiting string.

Returns

A vector of 2 strings. The first contains the characters between the first and the second mark, the second between the 2nd and the third mark. If the delimiters were not found, an empty vector is returned.

Since

2013-01-29

Author

Richard J. Mathar

4.2.3.11 `void FEditHead::readStream (const char * gstream, int glines, const bool restr = false)` [protected]

Put the 80-bytes chunks of the stream into the primary header.

Parameters

	<i>gstream</i>	The characters to be fed into the primary header. This stream contains neither line feeds nor 0-bytes.
in	<i>glines</i>	The number of 80-byte blocks in <i>gstream</i>
in	<i>restr</i>	Restrict the lines such that NAXIS keywords are blocked

Since

2013-07-15

4.2.3.12 `bool FEditHead::addHduCard (HDU & hdu, char * card, const bool verbose = false)` [static], [protected]

Insert (or replace) a HDU header card.

Parameters

	<i>hdu</i>	The header-data unit to be checked for the card.
	<i>card</i>	The (first 80 bytes of) the new header card.
	<i>verbose</i>	

Returns

true if the new card was installed. Return values of false occur if the syntax of the card is not valid.

Since

2013-07-15

4.2.4 Member Data Documentation

4.2.4.1 `String FEditHead::cfgFil`

Name of the configuration file with header template keywords.

4.2.4.2 `char* FEditHead::buffer`

The concatenation of all input lines, trimmed/expanded to a multiple of 80 characters per line, one final \0.

4.2.4.3 `vector<string> FEditHead::tplLins` [protected]

The assembly of all template keyword ASCII lines.

These are all lines with the exception of empty or comment lines. They are kept in the order of the configuration file, which may be important if the same keyword may be targeted at more than one place (in case of which the templates are enacted from the top to the bottom of the configuration file.)

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

- [fedithead/src/FEditHead.h](#)
- [fedithead/src/FEditHead.cxx](#)

5 File Documentation

5.1 `fedithead/src/config.h` File Reference

Macros

- `#define HAVE_INTTYPES_H`
- `#define HAVE_LIBBOOST_REGEX`
- `#define HAVE_LIBCCFITS`
- `#define HAVE_MEMORY_H`
- `#define HAVE_STDINT_H`
- `#define HAVE_STDLIB_H`
- `#define HAVE_STRINGS_H`
- `#define HAVE_STRING_H`
- `#define HAVE_STRSTR`
- `#define HAVE_SYS_STAT_H`
- `#define HAVE_SYS_TYPES_H`
- `#define HAVE_UNISTD_H`
- `#define PACKAGE_BUGREPORT`
- `#define PACKAGE_NAME`
- `#define PACKAGE_STRING`
- `#define PACKAGE_TARNAME`
- `#define PACKAGE_URL`
- `#define PACKAGE_VERSION`
- `#define STDC_HEADERS`

5.1.1 Macro Definition Documentation

5.1.1.1 `#define HAVE_INTTYPES_H`

5.1.1.2 `#define HAVE_LIBBOOST_REGEX`

5.1.1.3 `#define HAVE_LIBCCFITS`

5.1.1.4 `#define HAVE_MEMORY_H`

5.1.1.5 `#define HAVE_STDINT_H`

5.1.1.6 `#define HAVE_STDLIB_H`

5.1.1.7 `#define HAVE_STRINGS_H`

5.1.1.8 `#define HAVE_STRING_H`

5.1.1.9 `#define HAVE_STRSTR`

5.1.1.10 `#define HAVE_SYS_STAT_H`

5.1.1.11 `#define HAVE_SYS_TYPES_H`

5.1.1.12 `#define HAVE_UNISTD_H`

5.1.1.13 `#define PACKAGE_BUGREPORT`

5.1.1.14 `#define PACKAGE_NAME`

5.1.1.15 `#define PACKAGE_STRING`

5.1.1.16 `#define PACKAGE_TARNAME`

5.1.1.17 `#define PACKAGE_URL`

5.1.1.18 `#define PACKAGE_VERSION`

5.1.1.19 `#define STDC_HEADERS`

5.2 fedithead/src/csv2Fits.cxx File Reference

[csv2Fits](#) translates an ASCII file in the CSV format to a FITS file with a binary table.

Macros

- `#define CSV2FITS_WITHLINECNT`
Activate some debugging.

Functions

- void [usage](#) (char *argv0)
Emit a short usage reminder.
- int [main](#) (int argc, char *argv[])

5.2.1 Detailed Description

[csv2Fits](#) translates an ASCII file in the CSV format to a FITS file with a binary table.

5.2.2 Macro Definition Documentation

5.2.2.1 `#define CSV2FITS_WITHLINECNT`

Activate some debugging.

Not useful for production. If defined, use two scans of the input CSV file to allocate all rows of the binary FITS table in advance

5.2.3 Function Documentation

5.2.3.1 void [usage](#) (char * argv0)

Emit a short usage reminder.

Parameters

in	argv0	The name of the executable.
----	-------	-----------------------------

Author

R. J. Mathar

Since

2013-04-30

5.2.3.2 int main (int argc, char * argv[])

5.3 fedithead/src/csv2Fits.h File Reference

Classes

- class [csv2Fits](#)
csv2Fits converts a file with comma-separated values to a binary FITS table.

5.4 fedithead/src/FEditHead.cxx File Reference

5.5 fedithead/src/fedithead.cxx File Reference

Functions

- void [usage](#) (char *argv0)
Print a usage syntax message detailing the command line.
- int [main](#) (int argc, char **argv)
Fedithead is a standalone program which edits FITS header data following directions from a configuration file.

5.5.1 Function Documentation

5.5.1.1 void usage (char * argv0)

Print a usage syntax message detailing the command line.

Parameters

in	argv0	The name of the main program.
----	-------	-------------------------------

Since

2012-10-18

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

Fedithead is a standalone program which edits FITS header data following directions from a configuration file.

An optional argument `-v` triggers a detailed message of the program for each keyword changed.

The first command line argument is the path/file name of an existing FITS file which is to be modified.

The second argument and optionally further arguments are ASCII files structured very similar to the template files used with cfitsio and fmodhead.

Each of these may contain empty lines and comment lines (starting with #) that have no effect.

It may contain lines starting with the dash (-) that demand removal of the keyword from the FITS header. (If that keyword does not exist this does not have any effect.) The keyword may have regex expressions to deal with a group of keywords at once.

It may contain lines that embed two keyword names between colons (:) or between exclamation marks (!), so there are three of these delimiters in that type of line. (This is a syntactical extension to template files of fmodhead, fedithd and the cfitsio templates). Fits header cards with names matching the regular expression delimited by the first two colons have their names substituted by the substitutional expression between the 2nd and third colon. (Values and comments remain as they are).

It may contain lines that start with at least 8 blanks. The rest of these lines is turned into COMMENT lines that are appended to the FITS header.

Finally, all other lines are interpreted as keyword-value-comment triples in FITS header style (with = and / as delimiters), that trigger adding that card to the header. (Existing keywords with the same name are removed).

Example file (see also the fmodhead examples in the source distribution:

```
# delete CHOP_A and CHOP_B
-CHOP_[AB]

# replace RHUM by a hierarchical version
:RHUM:HIERARCH LN AMBI RHUM:

# rename enumerated wheels to filters
:WHEEL(\1):HIERARCH LN ICS FILT\1:

# add a OBSERVAT keyword
OBSERVAT = "LBT" / on the mountain

# add a comment
Nice observation conditions. Dry with occasional snowflakes.
```

Parameters

in	argc	The number of command line switches and arguments.
in	argv	The vector of all command line arguments.

Returns

0 if the file manipulation was successful.

Since

2012-10-18
2013-10-30 Support more than one template header file.

5.6 fedithead/src/FEditHead.h File Reference

Classes

- class [FEditHead](#)
The class edits header keywords FITS headers.

6 Example Documentation

6.1 /home/mathar/work/geirs/fedithead/src/csv2Fits.cxx

Scan an ASCII file (CSV format) and convert this into a binary FITS table. Given a readable CSV file in the file system, the program generates a FITS file which contains a single extension in the binary FITS table format.

The name of the FITS file is chosen by replacing the .csv extension of the CSV source file by the extension .fits.

Each line (but the first) of the CSV file becomes a row in the FITS table. The comma-separated first line in the CSV file contains the names of the columns to be used in the FITS columns.

The syntax in overview:

```
csv2Fits [-h] [-F fieldsep] typstr infile.csv
```

The option -h just emits a reminder for the syntax of the call. The same reminder is given if the two mandatory command line arguments are missing.

The option -F followed by a single character changes the field separator definition in infile.csv. The default is the comma.

The first command line argument is a concatenation of one or more letters of the set {a, d, f, b, i, k, j}. There are exactly as many letters as columns in the CSV file, and from left to right, each letter denotes the FITS type of the associated column of the CSV file. The Makefile shows the example

```
csv2Fits aaaaadafafbbbaa csv2FitsXmpl.csv
```

with 5 ASCII strings in the first 5 columns, one double value in the 6th column etc.

- The letter 'a' triggers the xA FITS format for strings. The width is determined by the maximum length of strings in that particular column of the CSV file. In consequence the 'x' in this format will differ for most of the columns that are in the ASCII/string format.
- The letter 'd' triggers the 1D format for numbers in double-precision.
- The letter 'f' triggers the 1F format for numbers in single-precision
- The letter 'b' triggers the 1B format for 8-bit integers.
- The letter 'i' triggers the 1I format for 16-bit integers.
- The letter 'j' triggers the 1J format for 32-bit integers.
- The letter 'k' triggers the 1K format for 64-bit integers.

To convert the SAO star catalogue of
http://heasarc.gsfc.nasa.gov/FTP/heasarc/dbase/dump/heasarc_sao.tdat.gz
 into FITS file, (i) create an ASCII heasarc_sao.csv file by deleting the
 lines up to and including the <DATA> line, deleting the last <END> line,
 (ii) insert a first line

NAME|RA|PM_RA|PM_RA_ERR|RA_EPOCH|DEC|PM_DEC|PM_DEC_ERR|DEC_EPOCH|POS_ERR|LII|BII|P-
 G_MAG|V_MAG|SPECTYP|REF_VMAG|REF_STARNUM|REF_PG_MAG|REF_PM|REF_SPECTYP|REMARK-
 S|REF_SRC_CAT|NUM_SRC_CAT|DM|HD|HD_COMPO|GC|PM_RA_FK5|PM_DEC_FK5|CLASS at the place of
 the former <DATA> line, and (iii) call `csv2Fits -F '|' addbddd bdddddabbbbbbbjaaaaddi heasarc_sao.csv`

Parameters

in	argc	The number of command line arguments.
in	argv	The vector of options, arguments of the options and free-standing arguments

Author

Richard J. Mathar

Since

2013-04-30

Index

- ~FEditHead
 - FEditHead, [7](#)
- addHduCard
 - FEditHead, [11](#)
- allLines
 - FEditHead, [7](#)
- buffer
 - FEditHead, [11](#)
- CSV2FITS_WITHLINECNT
 - csv2Fits.cxx, [13](#)
- cfgFil
 - FEditHead, [11](#)
- cnvrt
 - csv2Fits, [2](#)
- colWidths
 - csv2Fits, [3](#)
- config.h
 - HAVE_INTTYPES_H, [12](#)
 - HAVE_LIBBOOST_REGEX, [12](#)
 - HAVE_LIBCCFITS, [12](#)
 - HAVE_MEMORY_H, [12](#)
 - HAVE_STDINT_H, [12](#)
 - HAVE_STDLIB_H, [12](#)
 - HAVE_STRING_H, [12](#)
 - HAVE_STRINGS_H, [12](#)
 - HAVE_STRSTR, [12](#)
 - HAVE_SYS_STAT_H, [13](#)
 - HAVE_SYS_TYPES_H, [13](#)
 - HAVE_UNISTD_H, [13](#)
 - PACKAGE_BUGREPORT, [13](#)
 - PACKAGE_NAME, [13](#)
 - PACKAGE_STRING, [13](#)
 - PACKAGE_TARNAME, [13](#)
 - PACKAGE_URL, [13](#)
 - PACKAGE_VERSION, [13](#)
 - STDC_HEADERS, [13](#)
- csv2Fits, [1](#)
 - cnvrt, [2](#)
 - colWidths, [3](#)
 - csv2Fits, [2](#)
 - csv2Fits, [2](#)
 - csvFil, [3](#)
 - fsep, [3](#)
 - strtok, [3](#)
- csv2Fits.cxx
 - CSV2FITS_WITHLINECNT, [13](#)
 - main, [14](#)
 - usage, [13](#)
- csvFil
 - csv2Fits, [3](#)
- exec
 - FEditHead, [7](#)
- FEditHead, [4](#)
 - ~FEditHead, [7](#)
 - addHduCard, [11](#)
 - allLines, [7](#)
 - buffer, [11](#)
 - cfgFil, [11](#)
 - exec, [7](#)
 - FEditHead, [5, 6](#)
 - FEditHead, [5, 6](#)
 - parse_template, [9](#)
 - readStream, [11](#)
 - scanTpl, [7–9](#)
 - splitColon, [10](#)
 - templateAnal, [10](#)
 - tplLins, [11](#)
 - trimws, [9](#)
- fedithead.cxx
 - main, [14](#)
 - usage, [14](#)
- fedithead/src/FEditHead.cxx, [14](#)
- fedithead/src/FEditHead.h, [15](#)
- fedithead/src/config.h, [12](#)
- fedithead/src/csv2Fits.cxx, [13](#)
- fedithead/src/csv2Fits.h, [14](#)
- fedithead/src/fedithead.cxx, [14](#)
- fsep
 - csv2Fits, [3](#)
- HAVE_INTTYPES_H
 - config.h, [12](#)
- HAVE_LIBBOOST_REGEX
 - config.h, [12](#)
- HAVE_LIBCCFITS
 - config.h, [12](#)
- HAVE_MEMORY_H
 - config.h, [12](#)
- HAVE_STDINT_H
 - config.h, [12](#)
- HAVE_STDLIB_H
 - config.h, [12](#)
- HAVE_STRING_H
 - config.h, [12](#)
- HAVE_STRINGS_H
 - config.h, [12](#)
- HAVE_STRSTR
 - config.h, [12](#)
- HAVE_SYS_STAT_H
 - config.h, [13](#)
- HAVE_SYS_TYPES_H
 - config.h, [13](#)
- HAVE_UNISTD_H
 - config.h, [13](#)
- main
 - csv2Fits.cxx, [14](#)
 - fedithead.cxx, [14](#)

PACKAGE_BUGREPORT

config.h, [13](#)

PACKAGE_NAME

config.h, [13](#)

PACKAGE_STRING

config.h, [13](#)

PACKAGE_TARNAME

config.h, [13](#)

PACKAGE_URL

config.h, [13](#)

PACKAGE_VERSION

config.h, [13](#)

parse_template

FEditHead, [9](#)

readStream

FEditHead, [11](#)

STDC_HEADERS

config.h, [13](#)

scanTpl

FEditHead, [7–9](#)

splitColon

FEditHead, [10](#)

strtok

csv2Fits, [3](#)

templateAnal

FEditHead, [10](#)

tplLins

FEditHead, [11](#)

trimws

FEditHead, [9](#)

usage

csv2Fits.cxx, [13](#)

fedithead.cxx, [14](#)