fedithead

RichardJ.Mathar

Generated by Doxygen 1.8.2

Tue May 14 2013 16:39:52

Contents

1 Class Index

1.1 Class List

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

2 File Index

2.1 File List

Here is a list of all files with brief descriptions:

3 Class Documentation

3.1 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.

3.1.1 Detailed Description

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

3.1.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*infile*	The name of the ASCII file in CSV format.
in	*f*	The field separator. This is a single character (the comma by default) whic separates adjacent fields in the input file.

Author

Richard J. Mathar

Since

2013-04-30

3.1.3 Member Function Documentation

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

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

Parameters

in	*typs*	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

3.1.3.2 vector< string > csv2Fits::strtok (const std::string *lin*) const [protected]

Decompose the CSV line by splitting it along commas.

Parameters

in	*lin*	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.

3.1.3.3 vector< int > csv2Fits::colWidths (int & *lineCt*) const [protected]

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

Parameters

out	*lineCt*	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

3.1.4 Member Data Documentation

3.1.4.1 std::string csv2Fits::csvFil

The name of the input file.

3.1.4.2 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

3.2 FEditHead Class Reference

The class edits header keywords FITS headers.

`#include <FEditHead.h>`

Public Member Functions

- FEditHead (char *fitsname, char *fitstpl)

  Constructor given a FITS file and a template ASCII file.
- void exec (const bool verbose)

  Execute the modifications of the template file in the header.

Public Attributes

- String cfgFil

  Name of the configuration file with header template keywords.

Protected Member Functions

- vector< string > scanTpl (char *fitstpl) const

  Scan the template file.

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 exclamaion marks as delimiters.
- static vector< string > splitColon (const string &instring, const string &delim)

  Split a string using a specific delimiting string.

Protected Attributes

- FITS fits

  The FITS file that will be read and rewritten with modified header keywords.
- vector< string > tplLins

  The assembly of all template keyword ASCII lines.

3.2.1 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 and it even allows to replace hierarchical keywords by simple keywords and vice versa. Groups of keywords that are pareseable 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.

Since

2012-10-18

Author

Richard J. Mathar

3.2.2 Constructor & Destructor Documentation

3.2.2.1 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	*fitsname*	The name of the FITS file to be modified.
in	*fitstpl*	The name of the ASCII file with the header template lines.

Since

2012-10-18

Author

Richard J. Mathar

3.2.3 Member Function Documentation

3.2.3.1 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

in	*verbose*	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.

If any keywords were changed or added, write a HISTORY line keeping track of the template file, and remove the CHECKSUM keyword if present.

3.2.3.2 vector< string > FEditHead::scanTpl (char * *fitstpl*) const [protected]

Scan the template file.

Parameters

in	*fitstpl*	The name of the template ASCII file. The file contains six types of lines: • 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 HIERARCH, 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.

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

3.2.3.3 string FEditHead::trimws (const string & *instring*) [static], [protected]

Strip leading and trailing white space from a string.

Parameters

in	*instring*	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.

3.2.3.4 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

in	*instring*	

Returns

A validated copy of the card line. This char array should be delete[]d by the caller after return.

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

Returns

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

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

Split a string using colons or exclamaion marks as delimiters.

Parameters

in	*instring*	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

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

Split a string using a specific delimiting string.

Parameters

in	*instring*	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	*delim*	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

3.2.4 Member Data Documentation

3.2.4.1 String FEditHead::cfgFil

Name of the configuration file with header template keywords.

3.2.4.2 FITS FEditHead::fits [protected]

The FITS file that will be read and rewritten with modified header keywords.

3.2.4.3 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 configuation file.)

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

- fedithead/src/FEditHead.h
- fedithead/src/FEditHead.cxx

4 File Documentation

4.1 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[])

4.1.1 Detailed Description

csv2Fits translates an ASCII file in the CSV format to a FITS file with a binary table.

4.1.2 Macro Definition Documentation

4.1.2.1 #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

4.1.3 Function Documentation

4.1.3.1 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

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

4.2 fedithead/src/csv2Fits.h File Reference

Classes

- class csv2Fits

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

4.3 fedithead/src/FEditHead.cxx File Reference

4.4 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.

4.4.1 Function Documentation

4.4.1.1 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

4.4.1.2 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 is an ASCII file structured very similar to the template files used with cfitsio and fmodhead.

It 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 man 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. These are converted to FITS COMMENT lines.

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).

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.

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

4.5 fedithead/src/FEditHead.h File Reference

Classes

- class FEditHead

  The class edits header keywords FITS headers.

5 Example Documentation

5.1 /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 aaaaadafafbbaa 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|-PG_MAG|V_MAG|SPECTYP|REF_VMAG|REF_STARNUM|REF_PG_MAG|REF_PM|REF_SPECTYP|REMAR-KS|REF_SRC_CAT|NUM_SRC_CAT|DM|HD|HD_COMPO|GC|PM_RA_FK5|PM_DEC_FK5|CLASS and (iii) call csv2Fits -F '|' addbdddbddddddabbbbbbbjaaaaddi heasarc_sao.csv

Parameters

in	*argc*	
in	*argv*	

Author

Richard J. Mathar

Since

2013-04-30

Index