CXC-DM-013

CXC Data Model

Vol. 13

Using the ASCII Kernel

Jonathan McDowell
Chandra X-ray Center
June 25, 2007

Contents

1 Overview **3**

2 Overview **3**

3 Using text files with the DM **4**

4 Adding FITS headers to text files **5**

4.1 Fixed format tables 6

5 Tweaking the ASCII kernel behavior **6**

6 Limitations **8**

6.1 Problems 8

1 Overview

In CIAO4.0, CIAO tools which operated up till now on FITS files will also work on suitable ASCII text files. The CIAO DM allows users to filter and manipulate tabular data in a convenient way. In user analysis, it's often more convenient to operate on ASCII text files which can be manipulated in editors like emacs, instead of using binary formats like FITS, and the new 'ASCII kernel' supports this.

2 Overview

CIAO users are familiar with the flexible filtering and binning capability that the DM tools (dmcopy, dmlist, etc) provide with FITS files. In CIAO 4.0, the same tools also work on ASCII (text) files containing tables.

For example, imagine a simple text file `my.dat` containing a table with three columns and four rows:

```
 21.0  41.3  21.8
 22.0  41.1  20.2
 23.0  43.8  17.3
 24.0  12.3  11.1
```

Then

```
dmlist "my.dat[col3=11:20][cols col2,col3]" data,clean
```

will produce

```
                  43.80                    17.30
                  12.30                    11.10
```

We believe this capability will be very useful for advanced analysis, when one is often manipulating text files of derived data. The ASCII kernel also provides the ability to create FITS files from text files - simply by doing `dmcopy my.dat myfits.dat` - which is useful in testing and prototyping, decreasing the overhead needed to make a FITS file with faked-up data in it.

Of course, you can put anything at all in an ASCII file, and the CIAO DM is not clever enough to understand an arbitrary file. We have tried to provide ways to describe various kinds of special file to it by specifying tab or comma separated fields, ways to ignore header lines, and so on. In addition, we've implemented a 'psuedo-FITS' format which has free-format header keywords and multiple extensions, for cases in which the full metadata content of a FITS file has to be preserved going to to ASCII and back.

The ASCII kernel allows you to do the following operations:

- DM filtering on text files.

```
dmlist "foo.txt[time=100:1000,energy=10:20]" data
```

- Dumping FITS files to a form usable by other code:

```
dmcopy foo.fits "foo.txt[opt kernel=text/simple]"
```

- Creating FITS files by making text files in emacs and then doing

```
dmcopy foo.txt foo.fits
```

  (This can be done in a different way with FTOOLS' fcreate, but it needs you to make several different input files.)

- Using external text-based tools in a thread, propagating full header and ending up back in FITS:

```
dmextract "event.fits[bin pi]" "pha.txt[opt kernel=text/dtf]" type=pha1
text_spectrum_program pha.txt > new.pha.txt
dmsort new.pha.txt sort.pha.fits
```

These different cases are best supported by several different flavors of text file. For many cases, we just want DM filtering functionality on a simple text file of columns of numbers, of the kind that astronomers typically use. However, for FITS file creation and mixed use of FITS and text in an analysis thread, we want a text format which supports full equivalence with a FITS file.

Note that in the above examples, we use a DM directive (" `[opt kernel=blah]` ") to qualify an output file name - this is new in CIAO; up to now we've only had DM directive on the input file names to tools. The "[opt..]" directive is the only one you can use this way, you (obviously?) can't filter or bin output files.

3 Using text files with the DM

The simplest kind of ASCII text table contains just free-format columns of numbers. Often, however, the file also contains some kind of simple metadata which we have to either read or work around. For this reason, the ASCII kernel does its best to guess what is going on in the file, but also supports a range of special options to let users describe the file's peculiarities.

We recognize lines beginning with # as comment lines, and by default treat the first such line as defining the column names. So, a simple file looking like

```
1.4 5.2 1
2.3 1.8 0.99
4   11  12
```

will be interpreted as a three-column table with columns called col1, col2 and col3, all of type double. Only the first data line is parsed to identify the columns, so to handle to all-too-common case where - like here - the first line has something that looks like an integer but later lines are floating point, we don't try and distinguish between integer and floating point in these simple files.

If you add

```
# Time Energy Factor
#  s    keV    -
# List of energy versus time

  1.4 5.2 1
  2.3 1.8 0.99
  4   11  12
```

the columns get named "Time", "Energy", "Factor", and the DM treats the next lines containing the "s keV -" etc as comment lines. (We don't have a way of specifying units or data types in this format yet; that may come later). Blank lines are ignored.

For string-valued data, note that comparisons are done ignoring trailing spaces. Strings are assumed to be space-delimited.

4 Adding FITS headers to text files

If you want to add real FITS-style header keywords, the file becomes CIAO DM's standardized Data Text Format (DTF). For a single table, all you need to do is make sure you put the END keyword in so CIAO can figure out where the data starts.

```
HKEYNO = 5.8 / a header key
END
... (data)
```

You may want to add TTYPE keywords to name the columns and TFORM columns to specify data types (here, the 1J says that col 3 is integer):

```
HKEYNO = 5.8 / a header key
TTYPE1 = 'TIME'
TTYPE2 = 'COL2'
TTYPE3 = 'PHA'
TFORM3 = '1J'
TTYPE4 = 'N2'
TTYPE5 = 'N3'
END
```

```
78247473.5340545774    2    0          57        1
78247473.5750945807    7    2          57       66
78247473.5750945807    7    2          57       64
78247473.5750945807    7    2          57       72
```

If you want to add a second HDU, you need to write

```
XTENSION='TABLE'
```

as its first keyword, following the data from the previous HDU. When the DM writes out one of these files, it will also write

```
#TEXT/DTF
```

as the first line of the file to make it clear what format it is, and allow for the possibility of version markers (#TEXT/DTF-2.0 etc) in later versions. However, when making your own files you don't need this line.

This DTF format is conceived as an ASCII version of FITS. The END key is mandatory; data begins on the first non blank line following the END. New HDUs are allowed with lines beginning with XTENSION. This allows us to have full FITS table (and limited image) support, and gives an easy way to mock up a FITS file (with dmcopy dtf.txt dtf.fits kernel=fits). You can add the usual TUNITn, TFORMn keywords.

4.1 Fixed format tables

By default, the DTF XTENSION="TABLE" tables have free format data, but you can have nicely formatted fixed-field-length DTF files using the TBCOLn and TDISPn (but not TFORMn) keywords to give starting byte positions and Fortran-style formats, in FITS XTENSION="TABLE" style.

5 Tweaking the ASCII kernel behavior

The following options are used to qualify a text file, and may be combined in a comma-separated list. The first three are used on output files to control the form of the output; you may wish to experiment to see the differences.

- `[opt kernel=text/dtf]` Force the output to be DTF style.
- `[opt kernel=text/raw]` Force the output to be raw data columns with no header.
- `[opt kernel=text/simple]` Force the output to be simple or "SM style" with # as the default comment character.

- `[opt sep=:]` Define : (or some other character,e.g. \t) to be the separator for data fields. Unless 'white' is also used (see below), the sep directive says that each such character defines a new field, so that

  ```
  14.1::23.2:15.1
  ```

  represents 4 fields, with the second one being empty.

- `[opt sep=:,white]` Define : (or some other character,e.g. \t) to be the separator for data files, and treat as whitespace. This means that if you have multiple separator characters next to each other, they only count as one separator. For example

  ```
  14.1::23.2:15.1
  ```

  would represent three fields. The default separator is space, treated as a whitespace separator.

- You can use more than one character as a separator, for instance `[opt sep=":;"]` defines both : and ; to be separators.

- `[opt skip=3]` says to skip three lines at the beginning of the file. This helps handle some formats with fixed headers.

- `[opt comment=#]` says that lines beginning with # (this is the default) prior to the first data line will be treated as comments, except for one special line which the colnames directive controls:

- `[opt colnames=first]` The first comment-character line is treated specially, as a space-separated list of column names. This is the default; the other options are colnames=last (the last comment-char line prior to the first data line) and colnames=none (none of the lines are treated as a colnames definition).

The default options for text/raw are:

```
comment='#',sep=" \t\r",white,colnames=none,skip=0
```

For text/simple, we have:

```
comment='#',sep=" \t\r",white,colnames=first,skip=0
```

For text/dtf, we have:

```
sep=" \t\r",white,colnames=none,skip=0
```

6 Limitations

The ASCII kernel is particularly useful in interactive data analysis, to allow CIAO users to use the familiar DM syntax in manipulating and filtering text files they may have created for other reasons. It is not intended as a replacement for the FITS kernel in pipelines. Details of some limitations are given in this section.

6.1 Problems

1. The ASCII kernel has limitations on writing images. Therefore, you can use dmextract to make a simple ASCII equivalent of a PHA file, but not one with a WMAP image. If you copy a PHA file with a WMAP to ASCII, you can make dmcopy crash:

   ```
   dmcopy asc1w.pi.fits asc1w.pi.allasc"[opt kernel=text/dtf]" clob+ opt=all
   [SEGV]
   ```

   while a simple dmcopy, which just copies the PHA table without the WMAP, will work fine:

   ```
   dmcopy asc1w.pi.fits asc1w.pi.asc"[opt kernel=text/dtf]" clob+
   ```

2. If you want to create a dataset with more than one block, or write header keywords, you have to use the DTF flavor. Therefore, the Simple and Raw flavors cannot be used with many CIAO tools which require multiple blocks and header keywords.