CFITSIO 入門

木下大輔
Solar System Watching Team

1999 年 *10* 月 *27* 日

Abstract

この “CFITSIO 入門” では CFITSIO を使って FITS ファイルを読み込み、演算を施して、新たなファイルに書き出すという操作を行います。この基本的な作業ができるようになれば、自分の望む操作を実現するまでにそれほど時間は必要ないはずです。

1 CFITSIO とは何ぞや

CFITSIO とは天文学で事実上の業界標準のデータ・フォーマットになっている FITS (Flexible Image Transport System) ファイルを読み書きするためのサブルーチン・ライブラリです。 FITS の内部仕様にあまり深入りしなくてもデータをいじることができ、しかも、機種依存のないインターフェースを提供してくれます。詳しい背景については

`http://legacy.gsfc.nasa.gov/docs/software/fitsio/fitsio_info.html`

を参照してください。

CFITSIO は C 言語で書かれたライブラリで、 SunOS, Solaris, OSF/1, HP-UX, VAX/VMS, IRIX, Linux, MS-DOS, Windows95, MacOS など多くの OS 上での動作が確認されています。かつての FITSIO を利用したプログラムも利用できるように FORTRAN 用のラッパーも提供されています。CFITSIO のウェブページは

`http://legacy.gsfc.nasa.gov/docs/software/fitsio/fitsio.html`

です。

2 CFITSIO を手に入れよう

CFITSIO のソースコードは

`ftp://heasarc.gsfc.nasa.gov/software/fitsio/c/`

で入手することができます。 UN*X な環境では

`ftp://heasarc.gsfc.nasa.gov/software/fitsio/c/cfitsio2033.tar.gz`

をもらってきましょう (v2.033 が 1999 年 10 月 27 日時点での最新版です) 。詳しい Programmer's Guide が用意されています。 C ならば

`ftp://heasarc.gsfc.nasa.gov/software/fitsio/c/cfitsio.ps`

を、 Fortran ならば

`ftp://heasarc.gsfc.nasa.gov/software/fitsio/c/fitsio.ps`

も一緒に手に入れておくことをおすすめします。

3 CFITSIO のインストール

まずはもらってきた tar で固められ gzip で圧縮されたソースコードを元に戻します。 GNU の tar を利用できるなら

```
% tar xzvf cfitsio2033.tar.gz
```

とします。 GNU の tar が利用できない場合は

```
% gunzip -c cfitsio2033.tar.gz | tar xvf -
```

などとしてください。 cfitsio というディレクトリができているはずなので、そのディレクトリに移ります。

```
% cd cfitsio
```

コンパイルはいたって簡単です。

```
% ./configure
```

とすればシステムに合った Makefile が生成されます。そのあとで

```
% make
```

とすればコンパイルが始まります。無事にコンパイルが終了したら、できあがった libcfitsio.a と *.h を適当な場所にコピーします。例えば

```
% su
# cp libcfitsio.a /usr/local/lib/
# cp *.h /usr/local/include/
```

などとします。これでインストールは完了です。
もしもコンパイルがうまく行かない場合には、観念して C や UN*X の勉強をしましょう。それはそれで有意義な時間になるはずです。

4 CFITSIO を使おう

4.1 全体の流れ

まずはサンプル・プログラムを載せます。作業内容は、 FITS ファイルをオープンし、 FITS ファイルのキーワードを読み、データを読み込み、すべてのピクセルの値の平均値を計算し、すべてのピクセルの値を 2 倍し、新たな FITS ファイルを作っています。

```
#include <stdio.h>
#include <fitsio.h>

#define XMAX 2048
#define YMAX 2048

int main(int argc, char **argv) {
  /* Status */
  int status = 0;
  /* Declaration of Structure fitsfile */
  fitsfile *fptr;
  /* Number of found keywords */
  int num;
  /* Array for length of axis */
  long naxis[2];
  /* Value for Undefined Pixel */
```

```
double nulval = 0.0;
/* Any Null */
int anynull;
/* Non-grouped Data */
long group = 0;
/* Pixel Data */
double pix[YMAX][XMAX];

/* Sum */
double sum;
/* Average */
double average;
/* Variables for loops */
int i, j;

/* 2-Dimentional Image */
int dim = 2;
/* 64-bit Floating Point Pixel Value */
int bitpix = DOUBLE_IMG;


printf("File Name 1: %s\n", argv[1]);
printf("File Name 2: %s\n", argv[2]);

/* Open FITS file */
fits_open_file(&fptr, argv[1], READONLY, &status);

/* Read the keywords NAXIS1 and NAXIS2 */
fits_read_keys_lng(fptr, "NAXIS", 1, 2, naxis,
                   &num, &status);

printf("Number of found keywords: %d\n", num);
printf("Keyword NAXIS1: %ld\n", naxis[0]);
printf("Keyword NAXIS2: %ld\n", naxis[1]);

/* Read Data into 2-dim. Array */
fits_read_2d_dbl(fptr, group, nulval,
                 XMAX, naxis[0], naxis[1],
                 *pix, &anynull, &status);

/* Close FITS file */
fits_close_file(fptr, &status);

for (j = 0; j < naxis[1]; j++) {
  for (i = 0; i < naxis[0]; i++) {
    sum += pix[j][i];
  }
}

average = sum / (naxis[0] * naxis[1]);
printf("Average: %f\n", average);

for (j = 0; j < naxis[1]; j++) {
  for (i = 0; i < naxis[0]; i++) {
    pix[j][i] *= 2;
  }
}
```

```
  /* Open FITS File for Writing */
  fits_create_file(&fptr, argv[2], &status);
  /* Create FITS Image */
  fits_create_img(fptr, bitpix, dim, naxis, &status);
  /* Write the Array of Doubles to the FITS File */
  fits_write_2d_dbl(fptr, group, XMAX, naxis[0], naxis[1],
                    *pix, &status);
  /* Close FITS File */
  fits_close_file(fptr, &status);

  return(status);
}
```

4.2 ヘッダーファイルの読み込み

CFITSIO を利用するプログラムの先頭には

```
#include <fitsio.h>
```

という一行を必ず含めましょう。意味が分からない場合は基本的な C の勉強をする必要がありそうです。でも、とりあえず呪文のように書き加えておきましょう。

4.3 FITS ファイルのオープン

なにはともあれ FITS ファイルをオープンしなくてはなりません。儀式だと思ってください。ファイルにアクセスするためにはファイルハンドルをオープンしなくてはなりませんが、それと似たようなものです。 `fits_open_file()` という関数を使います。

```
fits_open_file(fitsfile **fptr, char *filename,
               int iomode, int *status);
```

最初に引数 `fptr` は CFITSIO で定義されている構造体です。あらかじめ宣言しておく必要があります。二つ目の引数 `filename` はオープンしたい FITS ファイルの名前です。三つ目の引数ではどんなモードでオープンするかを指定します。 `READONLY` または `READWRITE` のどちらかを指定してください。マクロになっています。最後の引数はオープンに成功したか失敗したかが返される変数です。

4.4 キーワードを読む

FITS ファイルのヘッダー部分に書かれているキーワードの値を読むには、例えば `fits_read_keys_lng()` という関数を用います。

```
fits_read_keys_lng(fitsfile *fptr, char *keyname,
                   int nstart, int nkeys,
                   DTYPE *numval, int *nfound,
                   int *status);
```

二つ目の引数 `keyname` で指定された文字列を含むキーワードの値が取り出されます。該当するキーワードのうち、 `nstart` 番目から `nkeys` 個のキーワードが配列 `numval` に代入されます。 `numval` は適切な型である必要があります。 `nfound` には該当するキーワードの数が代入されて返されます。

4.5 データを読み込む

データを読み込むためには、例えば `fits_read_2d_dbl()` という関数を用います。この関数は各ピクセルのデータを 2 次元配列に読み込みます。

```
fits_read_2d_dbl(fitsfile *fptr, long group, DTYPE nulval,
                 long dim1, long naxis1, long naxis2,
                 DTYPE array, int *anynul, int *status);
```

group は複雑な FITS ファイルを扱う場合には必要になってくるかもしれませんが、とりあえずは 0 をいれておけば問題ないはずです。 nulval は未定義のピクセルの値を置き換えるために使います。データは配列 array に格納されますが、その配列の最初の次元の長さを dim1 で指定します。 FITS データの大きさは naxis1 と naxis2 で指定します。

4.6 FITS ファイルのクローズ

読み込みや書き込みの終わった FITS ファイルはクローズします。

```
fits_close_file(fitsfile fptr, int *status);
```

fits_close_file() を使ってクローズすることができます。

4.7 FITS ファイルの書き出し

まず fits_create_file() を使って空の FITS ファイルを作ります。

```
fits_create_file(fitsfile **fptr, char *filename, int *status)
```

作り出すファイルの名前は filename で指定します。次に primary HDU (Header and Data Unit) を作ります。

```
fits_create_img(fitsfile *fptr, int bitpix,
                int naxis, long *naxes, int *status)
```

bitpix には

BYTE_IMG	8	unsigned char
SHORT_IMG	16	signed short integer
LONG_IMG	32	signed long integer
FLOAT_IMG	-32	float
DOUBLE_IMG	-64	double

のいずれかを選びます。お好きなものをどうぞ。 naxis にはデータの次元を、配列 naxes には各次元の長さを指定します。これでデータを書き込む準備が終わりました。実際にデータを新しい FITS ファイルとして書き出すためには、例えば fits_write_2d_dbl() という関数を使います。この関数は double でデータを書き込みます。

```
fits_write_2d_dbl(fitsfile *fptr, long group, DTYPE nulval,
                  long dim1, long naxis1, long naxis2,
                  DTYPE *array, int *anynul, int *status)
```

配列 array に詰まっている大事なデータが新しい FITS ファイルとして書き出されます。最後に fits_close_file() でクローズするのを忘れないようにしましょう。

4.8 コンパイル

ソースコードが完成したらコンパイルします。ソースファイルが fits.c で、ヘッダーファイルが /usr/local/include/cfitsio にあるとすると

```
% gcc -Wall -I/usr/local/include/cfitsio -L/usr/local/lib \
        -o fits fits.c -lcfitsio -lm
```

などとすればコンパイルできるはずです。新しくできた fits が実行ファイルです。

4.9 実行

最初に引数には読み込む FITS ファイルを指定します。二つ目の引数には書き出す FITS ファイルを指定します。

```
% ./fits foo.fits bar.fits
```

この例では foo.fits を読み込み、操作を行って bar.fits を書き出すことになります。実際に実行してみると

```
File Name 1: foo.fits
File Name 2: bar.fits
Number of found keywords: 2
Keyword NAXIS1: 256
Keyword NAXIS2: 256
Average: 104.193573
```

と表示されました。 IRAF の imstatistics で平均値を調べてみると

```
cl> imstat foo.fits
#               IMAGE      NPIX      MEAN    STDDEV       MIN       MAX
             foo.fits     65536     104.2     67.43       24.       500.
```

となり Average の値は MEAN の値と同じであることが確認できました。新しいファイルは各ピクセルの値を 2 倍にして書き出しているわけですが、こちらも確認してみると

```
cl> imstat bar.fits
#               IMAGE      NPIX      MEAN    STDDEV       MIN       MAX
             bar.fits     65536     208.4     134.9       48.     1000.
```

となっていて正しく動作していることが分かります。

5 次のステップ

これで最低限のことはできるようになったはずです。 CFITSIO は I/O しかやってくれないので、本質的にはこれですべてだと言っていいと思います。あとは自分の使いやすい関数を見つけて、使いやすく保守しやすいプログラムを書いてください。今回はプログラムを main 関数のなかにすべて押し込んでしまいましたが、実用的なプログラムを作るのであれば全体の流れと個々の詳細な処理を分けた設計の方がよいでしょう。使うデータ構造も構造体を使うなどしてもう少し洗練された形にすべきでしょう。

より詳しく CFITSIO について知るためには “CFITSIO User’s Guide” を参照することをおすすめします。個々の関数の仕様も記述されています。 CFITSIO のウェブページより手に入ります。

FITS それ自体については “FITS の手引き” が有用です。

http://www.fukuoka-edu.ac.jp/~kanamitu/fits/

より手に入ります。