Advanced Astronomical Observations 2021
Session 02: Manipulating FITS Files

Kinoshita Daisuke

26 February 2021
publicly accessible version

About this file. . .

- Important information about this file
    - The author of this file is Kinoshita Daisuke.
    - The original version of this file was used for the course “Advanced Astronomical Observations” (course ID: AS6005) offered at Institute of Astronomy, National Central University from February 2021 to June 2021.
    - The file is provided in the hope that it will be useful, but there is no guarantee for the correctness. Use this file at your own risk.
    - If you are willing to use this file for your study, please feel free to use. I’ll be very happy to receive feedback from you.
    - If you are willing to use this file for your teaching, please contact to Kinoshita Daisuke. When you use this file partly or entirely, please mention clearly that the author of the original version is Kinoshita Daisuke. Please help me to improve the contents of this file by sending your feedback.
    - Contact address: `https://www.instagram.com/daisuke23888/`

For this session, we deal with FITS files. FITS (Flexible Image Transport System) is a standard file format for astronomy. Almost all the data produced at research-oriented astronomical observatories are FITS files. For astronomical data reduction and analysis, we need to read and write FITS files. During today’s session, we download a set of FITS files, and try basic operations of FITS files.

1 Downloading data

A set of FITS files is placed at following location. Download the file.

- `https://s3b.astro.ncu.edu.tw/advobs_202102/data/data_ao2021_s02.tar.xz`

If you prefer to use a command-line tool, such as `curl`, then try following command.

```
% curl -k -o data_ao2021_s02.tar.xz \
? https://s3b.astro.ncu.edu.tw/advobs_202102/data/data_ao2021_s02.tar.xz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  409M  100  409M    0     0  5572k      0  0:01:15  0:01:15 --:--:-- 5588k
% ls -l data_ao2021_s02.tar.xz
-rw-r--r--  1 daisuke  taiwan  429905432 Feb 25 12:59 data_ao2021_s02.tar.xz
```

If you prefer to use a web browser, such as Firefox, then start a web browser and download the file.

2 Extracting data

The file you have downloaded is a compressed TAR archive file. To extract FITS files, try following command. 120 FITS files should be extracted from the archive file.

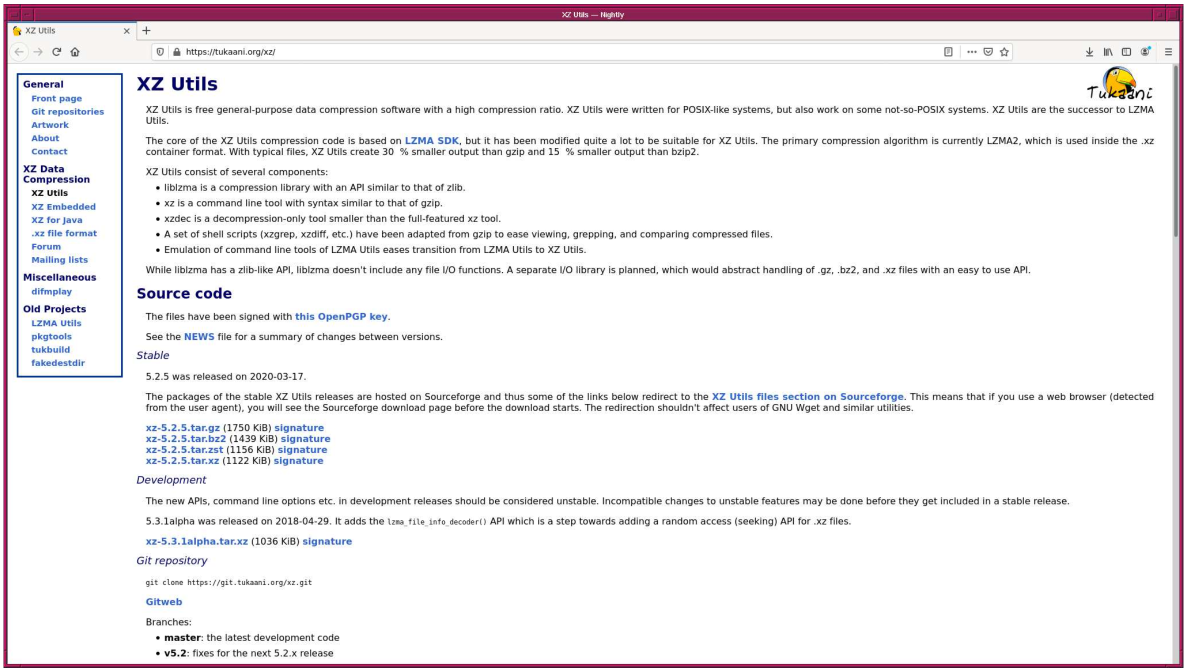
```
% ls -l data_ao2021_s02.tar.xz
-rw-r--r--  1 daisuke  taiwan  429905432 Feb 25 12:59 data_ao2021_s02.tar.xz
% tar xJvf data_ao2021_s02.tar.xz
x data_ao2021_s02/
x data_ao2021_s02/lot_20210214_0245.fits
x data_ao2021_s02/lot_20210214_0246.fits
x data_ao2021_s02/lot_20210214_0247.fits
x data_ao2021_s02/lot_20210214_0248.fits
x data_ao2021_s02/lot_20210214_0249.fits

.....

x data_ao2021_s02/lot_20210214_0534.fits
x data_ao2021_s02/lot_20210214_0535.fits
x data_ao2021_s02/lot_20210214_0536.fits
x data_ao2021_s02/lot_20210214_0537.fits
x data_ao2021_s02/lot_20210214_0538.fits
% ls -l data_ao2021_s02 | head
total 968
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 13 02:16 lot_20210214_0245.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 13 02:16 lot_20210214_0246.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 13 02:16 lot_20210214_0247.fits
-rw-r--r--  1 daisuke  taiwan  8400960 Feb 13 02:21 lot_20210214_0248.fits
-rw-r--r--  1 daisuke  taiwan  8400960 Feb 13 02:25 lot_20210214_0249.fits
-rw-r--r--  1 daisuke  taiwan  8400960 Feb 13 02:28 lot_20210214_0250.fits
-rw-r--r--  1 daisuke  taiwan  8400960 Feb 13 02:31 lot_20210214_0251.fits
-rw-r--r--  1 daisuke  taiwan  8400960 Feb 13 02:35 lot_20210214_0252.fits
-rw-r--r--  1 daisuke  taiwan  8400960 Feb 13 02:38 lot_20210214_0253.fits
% ls -l data_ao2021_s02/*.fits | wc
     120    1080   10920
```

If above command does not work on your computer, then try following.

```
% unxz -c data_ao2021_s02.tar.xz | tar xvf -
x data_ao2021_s02/
x data_ao2021_s02/lot_20210214_0245.fits
x data_ao2021_s02/lot_20210214_0246.fits
x data_ao2021_s02/lot_20210214_0247.fits
x data_ao2021_s02/lot_20210214_0248.fits
x data_ao2021_s02/lot_20210214_0249.fits

.....

x data_ao2021_s02/lot_20210214_0534.fits
x data_ao2021_s02/lot_20210214_0535.fits
x data_ao2021_s02/lot_20210214_0536.fits
x data_ao2021_s02/lot_20210214_0537.fits
x data_ao2021_s02/lot_20210214_0538.fits
```

If above command fails, you probably do not have XZ Utils. If you do not have XZ Utils, visit following website (Fig. 1) and install XZ Utils.

- `https://tukaani.org/xz/`



XZ Utils

XZ Utils is free general-purpose data compression software with a high compression ratio. XZ Utils were written for POSIX-like systems, but also work on some not-so-POSIX systems. XZ Utils are the successor to LZMA Utils.

The core of the XZ Utils compression code is based on **LZMA SDK**, but it has been modified quite a lot to be suitable for XZ Utils. The primary compression algorithm is currently LZMA2, which is used inside the .xz container format. With typical files, XZ Utils create 30 % smaller output than gzip and 15 % smaller output than bzip2.

XZ Utils consist of several components:

- liblzma is a compression library with an API similar to that of zlib.
- xz is a command line tool with syntax similar to that of gzip.
- xzdec is a decompression-only tool smaller than the full-featured xz tool.
- A set of shell scripts (xzgrep, xzdiff, etc.) have been adapted from gzip to ease viewing, grepping, and comparing compressed files.
- Emulation of command line tools of LZMA Utils eases transition from LZMA Utils to XZ Utils.

While liblzma has a zlib-like API, liblzma doesn't include any file I/O functions. A separate I/O library is planned, which would abstract handling of .gz, .bz2, and .xz files with an easy to use API.

Source code

The files have been signed with **this OpenPGP key**.

See the **NEWS** file for a summary of changes between versions.

Stable

5.2.5 was released on 2020-03-17.

The packages of the stable XZ Utils releases are hosted on Sourceforge and thus some of the links below redirect to the **XZ Utils files section on Sourceforge**. This means that if you use a web browser (detected from the user agent), you will see the Sourceforge download page before the download starts. The redirection shouldn't affect users of GNU Wget and similar utilities.

xz-5.2.5.tar.gz (1750 KiB) **signature**
xz-5.2.5.tar.bz2 (1439 KiB) **signature**
xz-5.2.5.tar.zst (1156 KiB) **signature**
xz-5.2.5.tar.xz (1122 KiB) **signature**

Development

The new APIs, command line options etc. in development releases should be considered unstable. Incompatible changes to unstable features may be done before they get included in a stable release.

5.3.1alpha was released on 2018-04-29. It adds the `lzma_file_info_decoder()` API which is a step towards adding a random access (seeking) API for .xz files.

xz-5.3.1alpha.tar.xz (1036 KiB) **signature**

Git repository

```
git clone https://git.tukaani.org/xz.git
```

Gitweb

Branches:

- **master**: the latest development code
- **v5.2**: fixes for the next 5.2.x release

Figure 1: The website of XZ Utils.

3 Importing Astropy module

3.1 Importing Astropy module in interactive mode

For today's session, we need to use Astropy module. Check whether or not you have Astropy on your computer. If Astropy is properly installed on your computer, you can successfully import Astropy by typing `import astropy`. (Fig. 2)

```
% python3.9
Python 3.9.2 (default, Feb 21 2021, 12:39:42)
[GCC 7.5.0] on netbsd9
Type "help", "copyright", "credits" or "license" for more information.
>>> import astropy
>>> exit ()
```

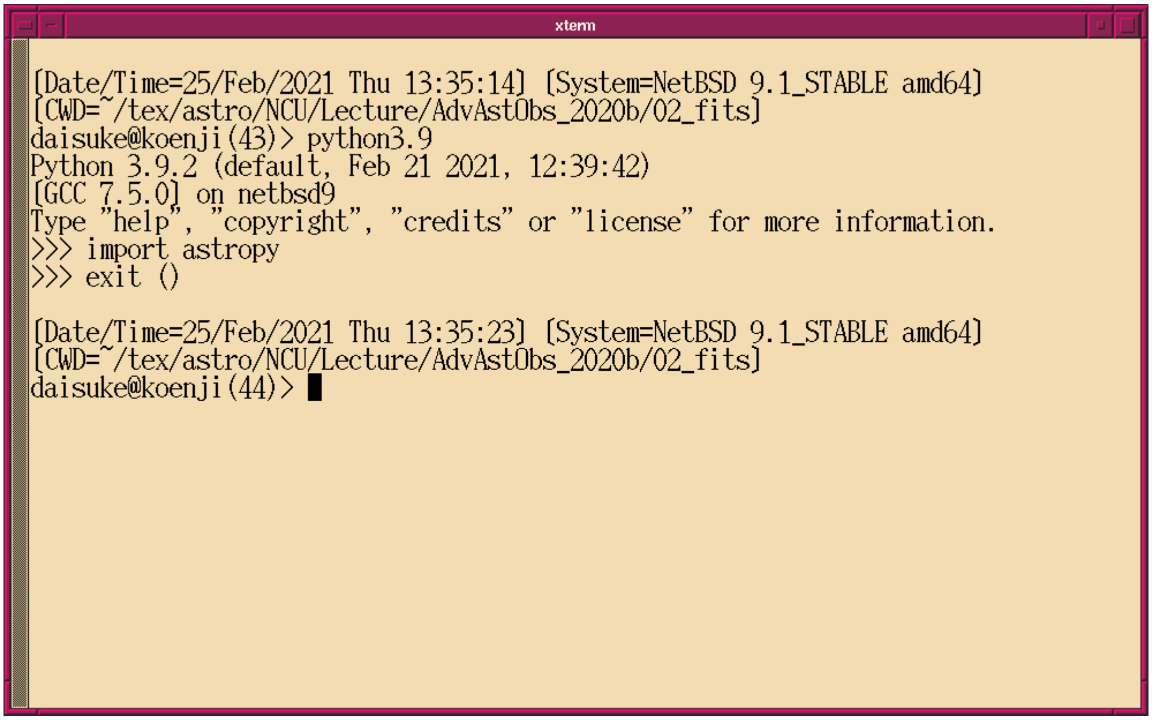
If you see error message after typing `import astropy`, then you do not have Astropy on your computer. Visit the official website of Astropy (Fig. 3), and install Astropy on your computer.

3.2 Making a simple Python script using Astropy

Make a simple Python script using Astropy.

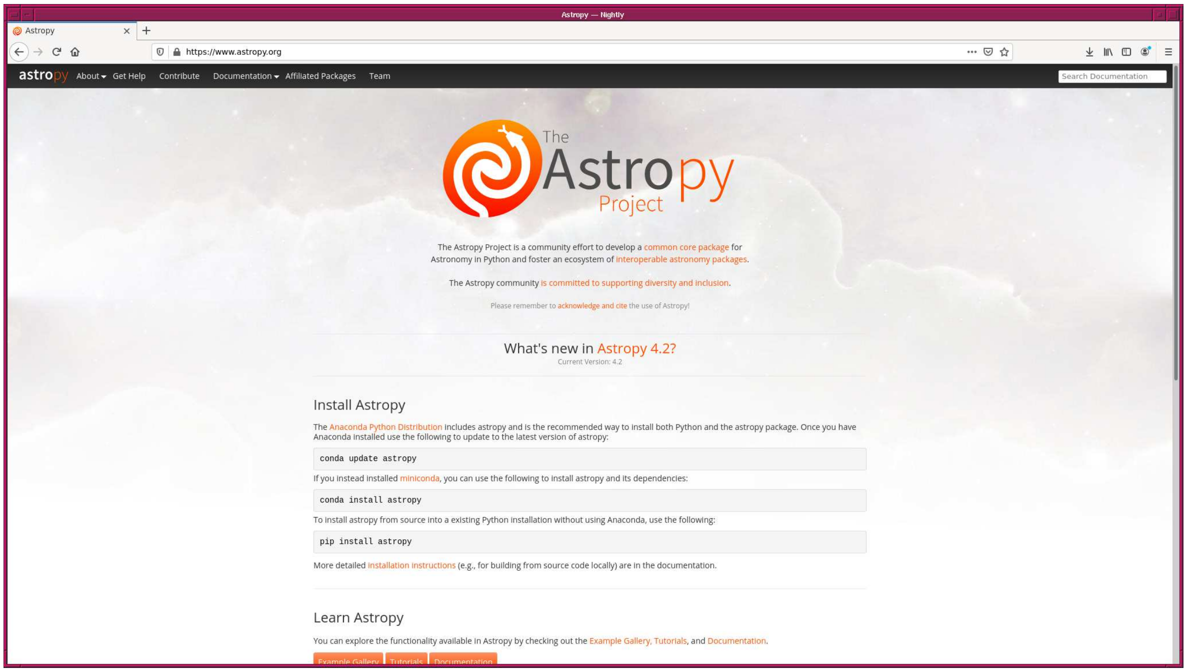
Python Code 1: ao2021_s02_01.py

```
#!/usr/pkg/bin/python3.9

# importing Astropy module
import astropy.constants

# speed of light
c = astropy.constants.c

# printing speed of light c
print (c)
```



```
[Date/Time=25/Feb/2021 Thu 13:35:14] [System=NetBSD 9.1_STABLE amd64]
[CWD=~/tex/astro/NCU/Lecture/AdvAstObs_2020b/02_fits]
daisuke@koenji(43)> python3.9
Python 3.9.2 (default, Feb 21 2021, 12:39:42)
[GCC 7.5.0] on netbsd9
Type "help", "copyright", "credits" or "license" for more information.
>>> import astropy
>>> exit ()

[Date/Time=25/Feb/2021 Thu 13:35:23] [System=NetBSD 9.1_STABLE amd64]
[CWD=~/tex/astro/NCU/Lecture/AdvAstObs_2020b/02_fits]
daisuke@koenji(44)> 
```

Figure 2: Importing Astropy module in interactive mode of Python.



The Astropy Project is a community effort to develop a common core package for Astronomy in Python and foster an ecosystem of interoperable astronomy packages.

The Astropy community is committed to supporting diversity and inclusion.

Please remember to acknowledge and cite the use of Astropy!

What's new in Astropy 4.2?
Current Version: 4.2

Install Astropy

The Anaconda Python Distribution includes astropy and is the recommended way to install both Python and the astropy package. Once you have Anaconda installed use the following to update to the latest version of astropy:

```
conda update astropy
```

If you instead installed miniconda, you can use the following to install astropy and its dependencies:

```
conda install astropy
```

To install astropy from source into a existing Python installation without using Anaconda, use the following:

```
pip install astropy
```

More detailed installation instructions (e.g., for building from source code locally) are in the documentation.

Learn Astropy

You can explore the functionality available in Astropy by checking out the Example Gallery, Tutorials, and Documentation.

Figure 3: The official website of Astropy.

Execute the script.

```
% chmod a+x ao2021_s02_01.py
% ./ao2021_s02_01.py
  Name    = Speed of light in vacuum
  Value   = 299792458.0
  Uncertainty  = 0.0
  Unit   = m / s
  Reference = CODATA 2018
```

Astropy is successfully imported, and the information about the speed of light is shown.

4 Opening a FITS file

Make a Python script to open a FITS file and print HDU list information. Here is an example.

Python Code 2: ao2021_s02_02.py

```
#!/usr/pkg/bin/python3.9

# importing astropy module
import astropy.io.fits

# file name of FITS file
file_fits = 'data_ao2021_s02/lot_20210214_0245.fits'

# opening FITS file
hdu_list = astropy.io.fits.open (file_fits)

# printing HDU list information
print (hdu_list.info () )

# closing FITS file
hdu_list.close ()
```

Execute the script.

```
% chmod a+x ao2021_s02_02.py
% ./ao2021_s02_02.py
Filename: data_ao2021_s02/lot_20210214_0245.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU      61   (2048, 2048)  int16 (rescales to uint16)
None
```

One HDU (Header Data Unit) is found in the file. The dimensions of the image data is 2048 × 2048, and the image data is stored as 16-bit integer.

5 Printing FITS header

Make a Python script to read the header part of a FITS file and print it.

Python Code 3: ao2021_s02_03.py

```
#!/usr/pkg/bin/python3.9
```

```
# importing astropy module
import astropy.io.fits

# file name of FITS file
file_fits = 'data_ao2021_s02/lot_20210214_0245.fits'

# opening FITS file
hdu_list = astropy.io.fits.open (file_fits)

# primary HDU
hdu0 = hdu_list[0]

# header of primary HDU
header0 = hdu0.header

# closing FITS file
hdu_list.close ()

# printing FITS header
print (repr (header0))
```

Execute the script. You will see the header of a FITS file.

```
% chmod 755 ao2021_s02_03.py
% ./ao2021_s02_03.py
SIMPLE  =                    T
BITPIX  =                   16 /8 unsigned int, 16 & 32 int, -32 & -64 real
NAXIS   =                    2 /number of axes
NAXIS1  =                 2048 /fastest changing axis
NAXIS2  =                 2048 /next to fastest changing axis
BSCALE  =   1.0000000000000000 /physical = BZERO + BSCALE*array_value
BZERO   =   32768.000000000000 /physical = BZERO + BSCALE*array_value
DATE-OBS= '2021-02-12' /        YYYY-MM-DDThh:mm:ss observation start, UT
TIME-OBS= '18:16:20' /          HH:MM:SS observation start time, UT
EXPTIME =  0.00000000000000000 /Exposure time in seconds
EXPOSURE=  0.00000000000000000 /Exposure time in seconds
SET-TEMP=  -80.000000000000000 /CCD temperature setpoint in C
CCD-TEMP=  -80.000000000000000 /CCD temperature at start of exposure in C
XPIXSZ  =   15.000000000000000 /Pixel Width in microns (after binning)
YPIXSZ  =   15.000000000000000 /Pixel Height in microns (after binning)
XBINNING=                    1 /Binning factor in width
YBINNING=                    1 /Binning factor in height
XORGSUBF=                    0 /Subframe X position in binned pixels
YORGSUBF=                    0 /Subframe Y position in binned pixels
IMAGETYP= 'BIAS    ' /          Type of image
OBJCTRA = '12 32 04' /          Nominal Right Ascension of center of image
OBJCTDEC= '-02 06 03' /         Nominal Declination of center of image
OBJCTALT= ' 62.4274' /          Nominal altitude of center of image
OBJCTAZ = '157.3035' /          Nominal azimuth of center of image
OBJCTHA = ' -0.6864' /          Nominal hour angle of center of image
SITELAT = '23 28 07' /          Latitude of the imaging location
SITELONG= '120 52 25' /         Longitude of the imaging location
JD      =   2459258.2613425925 /Julian Date at start of exposure
JD-HELIO=   2459258.2654279913 /Heliocentric Julian Date at exposure midpoint
AIRMASS =   1.1277189765535098 /Relative optical path length through atmosphere
FOCALLEN=   8000.0000000000000 /Focal length of telescope in mm
APTDIA  =   1000.0000000000000 /Aperture diameter of telescope in mm
APTAREA =   772124.95592236519 /Aperture area of telescope in mm^2
```

```
SWCREATE= ’MaxIm DL Version 5.24 130419 0CYVP’ /Name of software that created
         the image
OBJECT  = ’dark    ’
TELESCOP= ’LOT     ’ /          telescope used to acquire this image
INSTRUME= ’Driver for Princeton Instruments cameras’
OBSERVER= ’lulin   ’
NOTES   = ’        ’
DETECTOR= ’        ’
OWNER   = ’Institute of Astronomy, NCU, Taiwan’
TIMESYS = ’UTC     ’
EQUINX  = ’2000    ’
EPOCH   = ’2000    ’
RADECSYS= ’FK5     ’
CAMERA  = ’SOPHIA  ’
GAIN    = ’2       ’
SITEELEV=                 2862
RMSNOISE= ’8.5     ’
OPERATOR= ’lulin   ’
CTYPE1  = ’RA---TAN’ /          fastest changing axis name
CTYPE2  = ’DEC--TAN’ /          slowest changing axis name
FOV     = ’13’’ 08" x 13’’ 08"’
PIXSIZE =  0.39000000000000001
FLIPSTAT= ’        ’
SWOWNER = ’Ming-Hsin Chang’ /   Licensed owner of software
CSTRETCH= ’Medium  ’ /          Initial display stretch mode
CBLACK  =                  594 /Initial display black level in ADUs
CWHITE  =                  611 /Initial display white level in ADUs
PEDESTAL=                    0 /Correction to add for zero-based ADU
```

We can find that the FITS file “lot_20210214_0245.fits” is a bias frame taken at 18:16:20 on 2021-02-12.

6 Printing a specific keyword in the header

Make a Python script to print a specific keyword and its value in the FITS header.

Python Code 4: ao2021_s02_04.py

```python
#!/usr/pkg/bin/python3.9

# importing astropy module
import astropy.io.fits

# file name of FITS file
file_fits = ’data_ao2021_s02/lot_20210214_0245.fits’

# opening FITS file
hdu_list = astropy.io.fits.open (file_fits)

# primary HDU
hdu0 = hdu_list[0]

# header of primary HDU
header0 = hdu0.header

# closing FITS file
hdu_list.close ()

# printing a specific keyword and its value in FITS header
```

```
print ("%-8s = %s" % (’TIME-OBS’, header0[’TIME-OBS’]) )
```

Run the script.

```
% chmod a+x ao2021_s02_04.py
% ./ao2021_s02_04.py
TIME-OBS = 18:16:20
```

7 Generating a simple observing log

Make a Python script to generate a simple observing log. Here is an example.

Python Code 5: ao2021_s02_05.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing astropy module
import astropy.io.fits

# construction of parser object
desc = ’Generating a simple observing log’
parser = argparse.ArgumentParser (description=desc)

# adding arguments
default_keyword = ’TIME-OBS,IMAGETYP,OBJECT,EXPTIME,FILTER’
parser.add_argument (’-k’, ’--keyword’, default=default_keyword, \
                     help=’a list of keyword to check’)
parser.add_argument (’files’, nargs=’+’, help=’FITS files’)

# command-line argument analysis
args = parser.parse_args ()

# input parameters
keyword = args.keyword
files   = args.files

# a list of keywords
list_keyword = keyword.split (’,’)

# processing files
for file in files:
    # if the extension of the file is not ’.fits’, then skip
    if (file[-5:] != ’.fits’):
        continue

    # file name
    path = file.split (’/’)
    filename = path[-1]

    # opening FITS file
    hdu_list = astropy.io.fits.open (file)

    # primary HDU
    hdu0 = hdu_list[0]
    # header of primary HDU
    header0 = hdu0.header

    # gathering information from FITS header
    record = filename
    for key in list_keyword:
        if key in header0:
            value = str (header0[key])
        else:
            value = "__NONE__"
        record += " %8s" % value

    # closing FITS file
    hdu_list.close ()

    # printing information
    print (record)
```

Execute the script as follows, and generate a simple observing log. The file name, time of the observation in UT, data type, target object name, exposure time, and filter name are shown.

```
% chmod a+x ao2021_s02_05.py
% ./ao2021_s02_05.py data_ao2021_s02/*.fits
lot_20210214_0245.fits 18:16:20     BIAS     dark      0.0 __NONE__
lot_20210214_0246.fits 18:16:27     BIAS     dark      0.0 __NONE__
lot_20210214_0247.fits 18:16:34     BIAS     dark      0.0 __NONE__
lot_20210214_0248.fits 18:18:33    LIGHT   PG1047    180.0 gp_Astrodon_2019
lot_20210214_0249.fits 18:21:50    LIGHT   PG1047    180.0 gp_Astrodon_2019
lot_20210214_0250.fits 18:25:16    LIGHT   PG1047    180.0 rp_Astrodon_2019
lot_20210214_0251.fits 18:28:33    LIGHT   PG1047    180.0 rp_Astrodon_2019
lot_20210214_0252.fits 18:31:58    LIGHT   PG1047    180.0 ip_Astrodon_2019
lot_20210214_0253.fits 18:35:15    LIGHT   PG1047    180.0 ip_Astrodon_2019
lot_20210214_0254.fits 18:38:55     BIAS     dark      0.0 __NONE__
lot_20210214_0255.fits 18:39:02     BIAS     dark      0.0 __NONE__
lot_20210214_0256.fits 18:39:09     BIAS     dark      0.0 __NONE__
lot_20210214_0257.fits 18:44:18    LIGHT    HCG57     60.0 gp_Astrodon_2019
lot_20210214_0258.fits 18:45:34    LIGHT    HCG57     60.0 gp_Astrodon_2019
lot_20210214_0259.fits 18:46:49    LIGHT    HCG57     60.0 gp_Astrodon_2019
lot_20210214_0260.fits 18:48:02    LIGHT    HCG57     60.0 gp_Astrodon_2019
lot_20210214_0261.fits 18:49:25    LIGHT    HCG57     60.0 rp_Astrodon_2019
lot_20210214_0262.fits 18:50:40    LIGHT    HCG57     60.0 rp_Astrodon_2019
lot_20210214_0263.fits 18:51:55    LIGHT    HCG57     60.0 rp_Astrodon_2019
lot_20210214_0264.fits 18:53:10    LIGHT    HCG57     60.0 rp_Astrodon_2019
lot_20210214_0265.fits 18:54:34    LIGHT    HCG57     60.0 ip_Astrodon_2019
lot_20210214_0266.fits 18:55:50    LIGHT    HCG57     60.0 ip_Astrodon_2019
lot_20210214_0267.fits 18:57:05    LIGHT    HCG57     60.0 ip_Astrodon_2019
lot_20210214_0268.fits 18:58:18    LIGHT    HCG57     60.0 ip_Astrodon_2019
lot_20210214_0269.fits 18:59:54     BIAS     dark      0.0 __NONE__
lot_20210214_0270.fits 19:00:01     BIAS     dark      0.0 __NONE__
lot_20210214_0271.fits 19:00:08     BIAS     dark      0.0 __NONE__
lot_20210214_0272.fits 19:02:27    LIGHT V0678VIR     60.0 rp_Astrodon_2019
lot_20210214_0273.fits 19:03:43    LIGHT V0678VIR     60.0 rp_Astrodon_2019
lot_20210214_0274.fits 19:04:57    LIGHT V0678VIR     60.0 rp_Astrodon_2019
lot_20210214_0275.fits 19:11:50     BIAS     dark      0.0 __NONE__
lot_20210214_0276.fits 19:11:57     BIAS     dark      0.0 __NONE__
lot_20210214_0277.fits 19:12:04     BIAS     dark      0.0 __NONE__
lot_20210214_0452.fits 22:03:45     FLAT     dark      5.0 ip_Astrodon_2019
```

```
lot_20210214_0453.fits 22:03:57     FLAT     dark      5.0 rp_Astrodon_2019
lot_20210214_0454.fits 22:04:09     FLAT     dark      5.0 gp_Astrodon_2019
lot_20210214_0455.fits 22:04:26     FLAT     dark      5.0 ip_Astrodon_2019
lot_20210214_0456.fits 22:04:37     FLAT     dark      5.0 rp_Astrodon_2019
lot_20210214_0457.fits 22:04:49     FLAT     dark      5.0 gp_Astrodon_2019
lot_20210214_0458.fits 22:05:05     FLAT     dark      5.0 ip_Astrodon_2019
lot_20210214_0459.fits 22:05:17     FLAT     dark      5.0 rp_Astrodon_2019
lot_20210214_0460.fits 22:05:29     FLAT     dark      5.0 gp_Astrodon_2019
lot_20210214_0461.fits 22:05:45     FLAT     dark      5.0 ip_Astrodon_2019
lot_20210214_0462.fits 22:05:57     FLAT     dark      5.0 rp_Astrodon_2019
lot_20210214_0463.fits 22:06:08     FLAT     dark      5.0 gp_Astrodon_2019
lot_20210214_0464.fits 22:06:24     FLAT     dark      5.0 ip_Astrodon_2019
lot_20210214_0465.fits 22:06:36     FLAT     dark      5.0 rp_Astrodon_2019
lot_20210214_0466.fits 22:06:48     FLAT     dark      5.0 gp_Astrodon_2019
lot_20210214_0467.fits 22:07:03     FLAT     dark      5.0 ip_Astrodon_2019
lot_20210214_0468.fits 22:07:15     FLAT     dark      5.0 rp_Astrodon_2019
lot_20210214_0469.fits 22:07:26     FLAT     dark      5.0 gp_Astrodon_2019
lot_20210214_0470.fits 22:07:42     FLAT     dark      5.0 ip_Astrodon_2019
lot_20210214_0471.fits 22:07:53     FLAT     dark      5.0 rp_Astrodon_2019
lot_20210214_0472.fits 22:08:05     FLAT     dark      5.0 gp_Astrodon_2019
lot_20210214_0473.fits 22:08:22     FLAT     dark      5.0 ip_Astrodon_2019
lot_20210214_0474.fits 22:08:34     FLAT     dark      5.0 rp_Astrodon_2019
lot_20210214_0475.fits 22:08:45     FLAT     dark      5.0 gp_Astrodon_2019
lot_20210214_0476.fits 22:09:01     FLAT     dark      5.0 ip_Astrodon_2019
lot_20210214_0477.fits 22:09:13     FLAT     dark      5.0 rp_Astrodon_2019
lot_20210214_0478.fits 22:09:24     FLAT     dark      5.0 gp_Astrodon_2019
lot_20210214_0479.fits 22:14:37     BIAS     dark      0.0 __NONE__
lot_20210214_0480.fits 22:14:42     BIAS     dark      0.0 __NONE__
lot_20210214_0481.fits 22:14:47     BIAS     dark      0.0 __NONE__
lot_20210214_0482.fits 22:14:52     BIAS     dark      0.0 __NONE__
lot_20210214_0483.fits 22:14:58     BIAS     dark      0.0 __NONE__
lot_20210214_0484.fits 22:15:03     BIAS     dark      0.0 __NONE__
lot_20210214_0485.fits 22:15:08     BIAS     dark      0.0 __NONE__
lot_20210214_0486.fits 22:15:13     BIAS     dark      0.0 __NONE__
lot_20210214_0487.fits 22:15:18     BIAS     dark      0.0 __NONE__
lot_20210214_0488.fits 22:15:23     BIAS     dark      0.0 __NONE__
lot_20210214_0489.fits 22:15:28     BIAS     dark      0.0 __NONE__
lot_20210214_0490.fits 22:15:33     BIAS     dark      0.0 __NONE__
lot_20210214_0491.fits 22:15:39     BIAS     dark      0.0 __NONE__
lot_20210214_0492.fits 22:15:44     BIAS     dark      0.0 __NONE__
lot_20210214_0493.fits 22:15:49     BIAS     dark      0.0 __NONE__
lot_20210214_0494.fits 22:15:54     BIAS     dark      0.0 __NONE__
lot_20210214_0495.fits 22:15:59     BIAS     dark      0.0 __NONE__
lot_20210214_0496.fits 22:16:04     BIAS     dark      0.0 __NONE__
lot_20210214_0497.fits 22:16:09     BIAS     dark      0.0 __NONE__
lot_20210214_0498.fits 22:16:14     BIAS     dark      0.0 __NONE__
lot_20210214_0499.fits 22:16:20     DARK     dark      5.0 __NONE__
lot_20210214_0500.fits 22:16:30     DARK     dark      5.0 __NONE__
lot_20210214_0501.fits 22:16:40     DARK     dark      5.0 __NONE__
lot_20210214_0502.fits 22:16:50     DARK     dark      5.0 __NONE__
lot_20210214_0503.fits 22:17:00     DARK     dark      5.0 __NONE__
lot_20210214_0504.fits 22:17:10     DARK     dark      5.0 __NONE__
lot_20210214_0505.fits 22:17:20     DARK     dark      5.0 __NONE__
lot_20210214_0506.fits 22:17:30     DARK     dark      5.0 __NONE__
lot_20210214_0507.fits 22:17:40     DARK     dark      5.0 __NONE__
lot_20210214_0508.fits 22:17:50     DARK     dark      5.0 __NONE__
lot_20210214_0509.fits 22:17:59     DARK     dark      5.0 __NONE__
lot_20210214_0510.fits 22:18:10     DARK     dark      5.0 __NONE__
lot_20210214_0511.fits 22:18:20     DARK     dark      5.0 __NONE__
```

```
lot_20210214_0512.fits 22:18:30     DARK     dark      5.0 __NONE__
lot_20210214_0513.fits 22:18:40     DARK     dark      5.0 __NONE__
lot_20210214_0514.fits 22:18:50     DARK     dark      5.0 __NONE__
lot_20210214_0515.fits 22:19:00     DARK     dark      5.0 __NONE__
lot_20210214_0516.fits 22:19:10     DARK     dark      5.0 __NONE__
lot_20210214_0517.fits 22:19:20     DARK     dark      5.0 __NONE__
lot_20210214_0518.fits 22:19:30     DARK     dark      5.0 __NONE__
lot_20210214_0519.fits 22:19:40     BIAS     dark      0.0 __NONE__
lot_20210214_0520.fits 22:19:45     BIAS     dark      0.0 __NONE__
lot_20210214_0521.fits 22:19:50     BIAS     dark      0.0 __NONE__
lot_20210214_0522.fits 22:19:55     BIAS     dark      0.0 __NONE__
lot_20210214_0523.fits 22:20:01     BIAS     dark      0.0 __NONE__
lot_20210214_0524.fits 22:20:06     BIAS     dark      0.0 __NONE__
lot_20210214_0525.fits 22:20:11     BIAS     dark      0.0 __NONE__
lot_20210214_0526.fits 22:20:16     BIAS     dark      0.0 __NONE__
lot_20210214_0527.fits 22:20:21     BIAS     dark      0.0 __NONE__
lot_20210214_0528.fits 22:20:26     BIAS     dark      0.0 __NONE__
lot_20210214_0529.fits 22:20:31     BIAS     dark      0.0 __NONE__
lot_20210214_0530.fits 22:20:36     BIAS     dark      0.0 __NONE__
lot_20210214_0531.fits 22:20:41     BIAS     dark      0.0 __NONE__
lot_20210214_0532.fits 22:20:46     BIAS     dark      0.0 __NONE__
lot_20210214_0533.fits 22:20:51     BIAS     dark      0.0 __NONE__
lot_20210214_0534.fits 22:20:56     BIAS     dark      0.0 __NONE__
lot_20210214_0535.fits 22:21:01     BIAS     dark      0.0 __NONE__
lot_20210214_0536.fits 22:21:06     BIAS     dark      0.0 __NONE__
lot_20210214_0537.fits 22:21:11     BIAS     dark      0.0 __NONE__
lot_20210214_0538.fits 22:21:16     BIAS     dark      0.0 __NONE__
```

Now we know better about what we have.

8 Reading image data from a FITS file

Make a Python script to read image data from a FITS file.

Python Code 6: ao2021_s02_06.py

```python
#!/usr/pkg/bin/python3.9

# importing astropy module
import astropy.io.fits

file_fits = 'data_ao2021_s02/lot_20210214_0245.fits'

# opening FITS file
hdu_list = astropy.io.fits.open (file_fits)

# primary HDU
hdu0 = hdu_list[0]

# header of primary HDU
header0 = hdu0.header

# data of primary HDU
data0 = hdu0.data

# closing FITS file
hdu_list.close ()
```

```
# printing a value of a pixel [1024,1024]
print ("image[1024,1024] =", data0[1024,1024])

# printing values of 10x10 subframe of the image
print ("image[1024:1034,1024:1034] =")
print (data0[1024:1034,1024:1034])
```

Run the script.

```
% chmod a+x ao2021_s02_06.py
% ./ao2021_s02_06.py
image[1024,1024] = 586
image[1024:1034,1024:1034] =
[[586 585 604 589 600 594 586 604 597 606]
 [608 587 605 584 584 585 597 611 590 597]
 [593 590 599 615 595 609 598 601 591 589]
 [594 593 589 600 595 595 592 584 598 598]
 [605 590 586 602 595 603 594 594 599 598]
 [593 589 585 592 600 594 594 590 606 601]
 [593 584 586 579 588 587 602 599 595 591]
 [591 605 604 598 604 601 583 599 606 606]
 [581 592 588 584 583 606 585 579 598 589]
 [594 593 595 592 588 599 602 593 592 578]]
```

The pixel values are shown. It is a bias frame, and pixels have values around 600.

9 Average value of the image

Make a Python script to calculate the average value of the image of a FITS file.

Python Code 7: ao2021_s02_07.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing numpy module
import numpy

# importing astropy module
import astropy.io.fits

# construction of parser object
desc = 'Calculating average value of the image'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('files', nargs='+', help='FITS files')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
files   = args.files

# printing header
```

```
print ("%-32s %8s %8s %8s %8s %8s" \
       % ('file', 'max', 'min', 'mean', 'median', 'stddev') )
print ("%s" % '-' * 77)

# processing files
for file in files:
    # if the extension of the file is not '.fits', then skip
    if (file[-5:] != '.fits'):
        continue

    # file name
    path = file.split ('/')
    filename = path[-1]

    # opening FITS file
    hdu_list = astropy.io.fits.open (file)

    # primary HDU
    hdu0 = hdu_list[0]

    # header of primary HDU
    header0 = hdu0.header

    # data of primary HDU
    data0 = hdu0.data

    # closing FITS file
    hdu_list.close ()

    # calculations of statistical values
    data_max      = numpy.amax (data0)
    data_min      = numpy.amin (data0)
    data_mean     = numpy.mean (data0)
    data_median   = numpy.median (data0)
    data_variance = numpy.var (data0)
    data_stddev   = numpy.std (data0)

    # printing results
    print ("%-32s %8.2f %8.2f %8.2f %8.2f %8.2f" \
           % (filename, data_max, data_min, \
              data_mean, data_median, data_stddev) )
```

Use above script to examine bias frames.

```
% chmod a+x ao2021_s02_07.py
% ./ao2021_s02_07.py data_ao2021_s02/lot_20210214_024[5-7].fits
file                                  max      min     mean   median   stddev
-----------------------------------------------------------------------------
lot_20210214_0245.fits            1363.00   556.00   595.62   596.00     7.88
lot_20210214_0246.fits            1590.00   551.00   594.80   595.00     7.90
lot_20210214_0247.fits             891.00   554.00   595.48   595.00     7.88
```

The mean count of bias frames is around 595.

```
% ./ao2021_s02_07.py data_ao2021_s02/lot_20210214_045?.fits
file                                  max      min     mean   median   stddev
-----------------------------------------------------------------------------
```

```
lot_20210214_0452.fits          20811.00  3808.00  7421.30  7425.00   247.37
lot_20210214_0453.fits          47063.00  4564.00  8972.19  8988.00   262.94
lot_20210214_0454.fits          36558.00  4873.00  9471.12  9501.00   288.62
lot_20210214_0455.fits          26382.00  4386.00  8709.72  8714.00   289.10
lot_20210214_0456.fits          49441.00  5253.00 10564.77 10583.00   305.15
lot_20210214_0457.fits          32915.00  5757.00 11303.81 11343.00   345.61
lot_20210214_0458.fits          31871.00  5127.00 10235.88 10241.00   341.95
lot_20210214_0459.fits          52127.00  6100.00 12433.64 12456.00   353.98
```

The mean count of flatfield frames is around 10,000.

10 Writing a new FITS file

Make a Python script to read a FITS file, normalise the image, and write a new FITS file.

Python Code 8: ao2021_s02_08.py

```python
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing numpy module
import numpy

# importing astropy module
import astropy.io.fits

# construction of parser object
desc = 'Normalising an image'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('fits', nargs=1, help='FITS file')
parser.add_argument ('-o', '--output', default='new.fits', \
                     help='new FITS file name')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
file_input  = args.fits[0]
file_output = args.output

print ("input file  =", file_input)
print ("output file =", file_output)

# opening FITS file
hdu_list = astropy.io.fits.open (file_input)

# primary HDU
hdu0 = hdu_list[0]

# header of primary HDU
header0 = hdu0.header

# data of primary HDU
data0 = hdu0.data
```

```
# calculations of statistical values
data_mean      = numpy.mean (data0)

# normalisation
data_new = data0 / data_mean

# writing normalised image into a file
hdu_new = astropy.io.fits.PrimaryHDU (data=data_new, header=header0)
hdu_new.writeto (file_output)

# closing FITS file
hdu_list.close ()
```

Execute the script.

```
% chmod a+x ao2021_s02_08.py
% ./ao2021_s02_08.py -o 0452n.fits data_ao2021_s02/lot_20210214_0452.fits
input file  = data_ao2021_s02/lot_20210214_0452.fits
output file = 0452n.fits
% ls -l 0452n.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Feb 25 17:08 0452n.fits
```

A new file “0452n.fits” is created. Examine the mean count of newly created file “0452n.fits”.

```
% ./ao2021_s02_07.py 0452n.fits
file                                      max      min     mean   median   stddev
-----------------------------------------------------------------------------
0452n.fits                               2.80     0.51     1.00     1.00     0.03
```

The mean value of the file “0452n.fits” is a unity.

11 The other way to create a new FITS file

Here is the other way to create a new FITS file. The function astropy.io.fits.writeto can be used to make a new FITS file.

Python Code 9: ao2021_s02_09.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing numpy module
import numpy

# importing astropy module
import astropy.io.fits

# construction of parser object
desc = 'Normalising an image'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('fits', nargs=1, help='FITS file')
parser.add_argument ('-o', '--output', default='new.fits', \
                     help='new FITS file name')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
file_input  = args.fits[0]
file_output = args.output

print ("input file  =", file_input)
print ("output file =", file_output)

# opening FITS file
hdu_list = astropy.io.fits.open (file_input)

# primary HDU
hdu0 = hdu_list[0]

# header of primary HDU
header0 = hdu0.header

# data of primary HDU
data0 = hdu0.data

# calculations of statistical values
data_mean     = numpy.mean (data0)

# normalisation
data_new = data0 / data_mean

# writing normalised image into a file
astropy.io.fits.writeto (file_output, data_new, header=header0)

# closing FITS file
hdu_list.close ()
```

Try this script.

```
% chmod a+x ao2021_s02_09.py
% ./ao2021_s02_09.py -o 0452n2.fits data_ao2021_s02/lot_20210214_0452.fits
input file  = data_ao2021_s02/lot_20210214_0452.fits
output file = 0452n2.fits
% ls -l 0452n*
-rw-r--r--  1 daisuke  taiwan  33560640 Feb 25 17:08 0452n.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Feb 25 17:15 0452n2.fits
% diff 0452n*
```

The file “0452n2.fits” is found to be identical to the file “0452n.fits”.

12 Adding new comments to a FITS file

After the processing of FITS files, we should leave some comments. Here is a sample script to add new comments to a FITS file.

Python Code 10: ao2021_s02_10.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing numpy module
import numpy

# importing astropy module
import astropy.io.fits

# construction of parser object
desc = ’Normalising an image’
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument (’fits’, nargs=1, help=’FITS file’)
parser.add_argument (’-o’, ’--output’, default=’new.fits’, \
                     help=’new FITS file name’)

# command-line argument analysis
args = parser.parse_args ()

# input parameters
file_input  = args.fits[0]
file_output = args.output

print ("input file  =", file_input)
print ("output file =", file_output)

# opening FITS file
hdu_list = astropy.io.fits.open (file_input)

# primary HDU
hdu0 = hdu_list[0]

# header of primary HDU
header0 = hdu0.header

# data of primary HDU
data0 = hdu0.data

# calculations of statistical values
data_mean     = numpy.mean (data0)

# normalisation
data_new = data0 / data_mean

# adding new comments to header
header0[’history’] = ’updated by Daisuke on 25/Feb/2021.’
header0[’comment’] = ’image was normalised.’

# writing normalised image into a file
astropy.io.fits.writeto (file_output , data_new , header=header0)

# closing FITS file
hdu_list.close ()
```

Try this script.

```
% chmod a+x ao2021_s02_10.py
% ./ao2021_s02_10.py -o 0452n3.fits data_ao2021_s02/lot_20210214_0452.fits
input file  = data_ao2021_s02/lot_20210214_0452.fits
output file = 0452n3.fits
% ls -l 0452n*
-rw-r--r--  1 daisuke  taiwan  33560640 Feb 25 17:08 0452n.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Feb 25 17:15 0452n2.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Feb 25 17:26 0452n3.fits
% diff 0452n2.fits 0452n3.fits
Binary files 0452n2.fits and 0452n3.fits differ
```

Use following Python script to show the header of FITS files.

Python Code 11: ao2021_s02_11.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing astropy module
import astropy.io.fits

# construction of parser object
desc = ’Normalising an image’
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument (’fits’, nargs=1, help=’FITS file’)

# command-line argument analysis
args = parser.parse_args ()

# input parameters
file_input  = args.fits[0]

# opening FITS file
hdu_list = astropy.io.fits.open (file_input)

# primary HDU
hdu0 = hdu_list[0]

# header of primary HDU
header0 = hdu0.header

print (repr (header0))

# closing FITS file
hdu_list.close ()
```

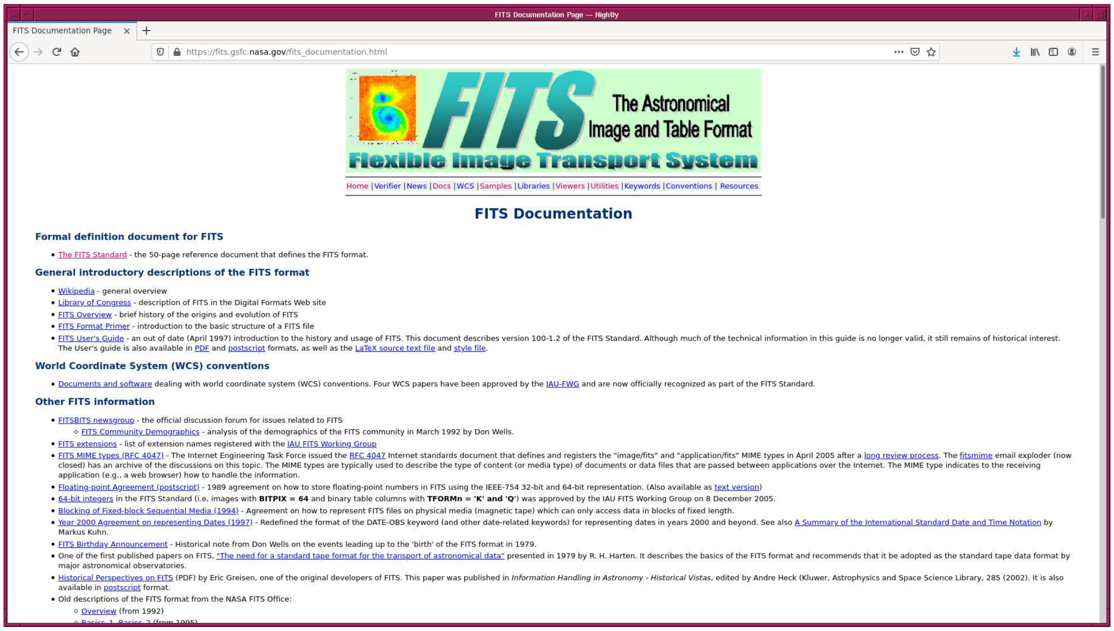
Compare the header of the file “`0452n2.fits`” and the file “`0452n3.fits`”.

```
% chmod a+x ao2021_s02_11.py
% ./ao2021_s02_11.py 0452n2.fits | tail
GAIN    = ’2       ’
SITEELEV=                 2862
RMSNOISE= ’8.5     ’
```

```
OPERATOR= 'lulin    '
CTYPE1  = 'RA---TAN' /          fastest changing axis name
CTYPE2  = 'DEC--TAN' /          slowest changing axis name
FOV     = '13'' 08" x 13'' 08"'
PIXSIZE =  0.39000000000000001
FLIPSTAT= '        '
SWOWNER = 'Ming-Hsin Chang' /   Licensed owner of software
% ./ao2021_s02_11.py 0452n3.fits | tail
RMSNOISE= '8.5     '
OPERATOR= 'lulin    '
CTYPE1  = 'RA---TAN' /          fastest changing axis name
CTYPE2  = 'DEC--TAN' /          slowest changing axis name
FOV     = '13'' 08" x 13'' 08"'
PIXSIZE =  0.39000000000000001
FLIPSTAT= '        '
SWOWNER = 'Ming-Hsin Chang' /   Licensed owner of software
HISTORY updated by Daisuke on 25/Feb/2021.
COMMENT image was normalised.
```

13 For your training

- Visit "FITS Documentation" page (Fig. 4), read relevant documents, and learn about FITS.
  - https://fits.gsfc.nasa.gov/fits_documentation.html
- Visit the section "FITS File Handling" of Astropy official document (Fig. 5), read it, and learn about FITS I/O related functions of Astropy.
  - https://docs.astropy.org/en/stable/io/fits/index.html


FITS Documentation Page — Nightly
FITS Documentation Page
https://fits.gsfc.nasa.gov/fits_documentation.html
FITS
The Astronomical Image and Table Format
Flexible Image Transport System
Home | Verifier | News | Docs | WCS | Samples | Libraries | Viewers | Utilities | Keywords | Conventions | Resources

FITS Documentation

Formal definition document for FITS

- The FITS Standard - the 50-page reference document that defines the FITS format.

General introductory descriptions of the FITS format

- Wikipedia - general overview
- Library of Congress - description of FITS in the Digital Formats Web site
- FITS Overview - brief history of the origins and evolution of FITS
- FITS Format Primer - introduction to the basic structure of a FITS file
- FITS User's Guide - an out of date (April 1997) introduction to the history and usage of FITS. This document describes version 100-1.2 of the FITS Standard. Although much of the technical information in this guide is no longer valid, it still remains of historical interest. The User's guide is also available in PDF and postscript formats, as well as the LaTeX source text file and style file.

World Coordinate System (WCS) conventions

- Documents and software dealing with world coordinate system (WCS) conventions. Four WCS papers have been approved by the IAU-FWG and are now officially recognized as part of the FITS Standard.

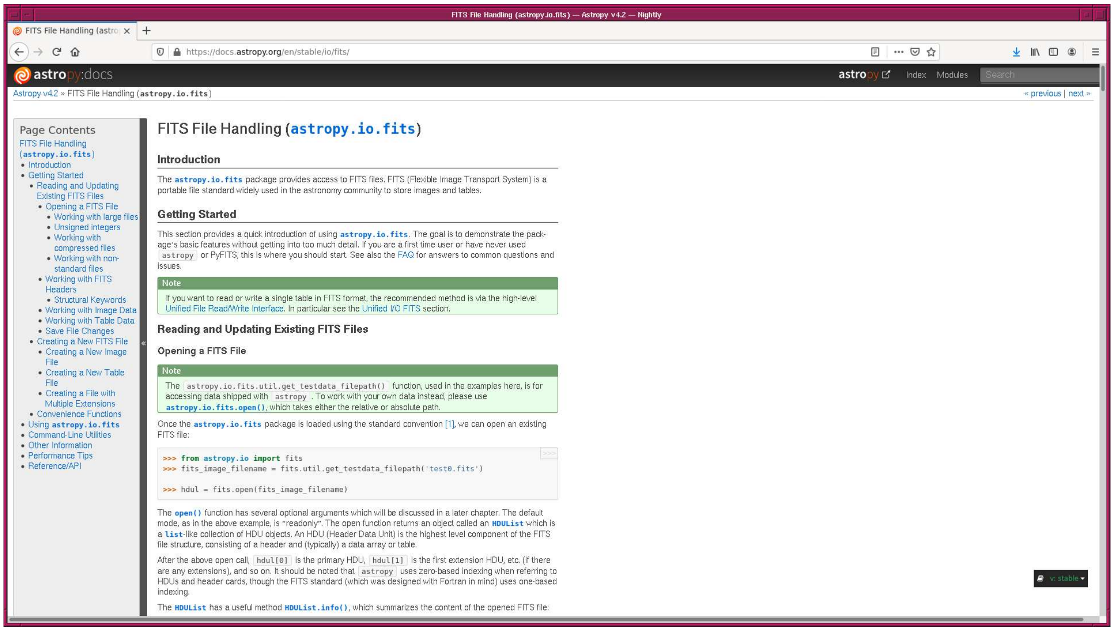
Other FITS information

- FITSBITS newsgroup - the official discussion forum for issues related to FITS
  - FITS Community Demographics - analysis of the demographics of the FITS community in March 1992 by Don Wells.
- FITS extensions - list of extension names registered with the IAU FITS Working Group
- FITS MIME types (RFC 4047) - The Internet Engineering Task Force issued the RFC 4047 Internet standards document that defines and registers the "image/fits" and "application/fits" MIME types in April 2005 after a long review process. The fitsmime email exploder (now closed) has an archive of the discussions on this topic. The MIME types are typically used to describe the type of content (or media type) of documents or data files that are passed between applications over the Internet. The MIME type indicates to the receiving application (e.g., a web browser) how to handle the information.
- Floating-point Agreement (postscript) - 1989 agreement on how to store floating-point numbers in FITS using the IEEE-754 32-bit and 64-bit representation. (Also available as text version)
- 64-bit integers in the FITS Standard (i.e, images with **BITPIX = 64** and binary table columns with **TFORMn = 'K' and 'Q'**) was approved by the IAU FITS Working Group on 8 December 2005.
- Blocking of Fixed-block Sequential Media (1994) - Agreement on how to represent FITS files on physical media (magnetic tape) which can only access data in blocks of fixed length.
- Year 2000 Agreement on representing Dates (1997) - Redefined the format of the DATE-OBS keyword (and other date-related keywords) for representing dates in years 2000 and beyond. See also A Summary of the International Standard Date and Time Notation by Markus Kuhn.
- FITS Birthday Announcement - Historical note from Don Wells on the events leading up to the 'birth' of the FITS format in 1979.
- One of the first published papers on FITS, "The need for a standard tape format for the transport of astronomical data" presented in 1979 by R. H. Harten. It describes the basics of the FITS format and recommends that it be adopted as the standard tape data format by major astronomical observatories.
- Historical Perspectives on FITS (PDF) by Eric Greisen, one of the original developers of FITS. This paper was published in *Information Handling in Astronomy - Historical Vistas*, edited by Andre Heck (Kluwer, Astrophysics and Space Science Library, 285 (2002). It is also available in postscript format.
- Old descriptions of the FITS format from the NASA FITS Office:
  - Overview (from 1992)
  - Basics 1, Basics 2 (from 1995)

Figure 4: The FITS documentation web page.

14 Assignment

1. Learn about FITS. Describe FITS. What is the overall design of FITS? What is the structure of a FITS file? What is an advantage of using FITS? How widely FITS is used in astronomy?


FITS File Handling (astropy.io.fits) — Astropy v4.2 — Nightly
FITS File Handling (astro
https://docs.astropy.org/en/stable/io/fits/
astropy:docs
astropy
Index
Modules
Search
Astropy v4.2 » FITS File Handling (astropy.io.fits)
« previous | next »
v: stable

Page Contents

- FITS File Handling (**astropy.io.fits**)
  - Introduction
  - Getting Started
    - Reading and Updating Existing FITS Files
      - Opening a FITS File
        - Working with large files
        - Unsigned integers
        - Working with compressed files
        - Working with non-standard files
      - Working with FITS Headers
        - Structural Keywords
      - Working with Image Data
      - Working with Table Data
      - Save File Changes
    - Creating a New FITS File
      - Creating a New Image File
      - Creating a New Table File
      - Creating a File with Multiple Extensions
    - Convenience Functions
  - Using **astropy.io.fits**
  - Command-Line Utilities
  - Other Information
  - Performance Tips
  - Reference/API

FITS File Handling (**astropy.io.fits**)

Introduction

The **astropy.io.fits** package provides access to FITS files. FITS (Flexible Image Transport System) is a portable file standard widely used in the astronomy community to store images and tables.

Getting Started

This section provides a quick introduction of using **astropy.io.fits**. The goal is to demonstrate the package's basic features without getting into too much detail. If you are a first time user or have never used `astropy` or PyFITS, this is where you should start. See also the FAQ for answers to common questions and issues.

Note

If you want to read or write a single table in FITS format, the recommended method is via the high-level Unified File Read/Write Interface. In particular see the Unified I/O FITS section.

Reading and Updating Existing FITS Files

Opening a FITS File

Note

The `astropy.io.fits.util.get_testdata_filepath()` function, used in the examples here, is for accessing data shipped with `astropy`. To work with your own data instead, please use **astropy.io.fits.open()**, which takes either the relative or absolute path.

Once the **astropy.io.fits** package is loaded using the standard convention [1], we can open an existing FITS file:

```
>>> from astropy.io import fits
>>> fits_image_filename = fits.util.get_testdata_filepath('test0.fits')

>>> hdul = fits.open(fits_image_filename)
```

The **open()** function has several optional arguments which will be discussed in a later chapter. The default mode, as in the above example, is "readonly". The open function returns an object called an **HDUList** which is a **list**-like collection of HDU objects. An HDU (Header Data Unit) is the highest level component of the FITS file structure, consisting of a header and (typically) a data array or table.

After the above open call, `hdul[0]` is the primary HDU, `hdul[1]` is the first extension HDU, etc. (if there are any extensions), and so on. It should be noted that `astropy` uses zero-based indexing when referring to HDUs and header cards, though the FITS standard (which was designed with Fortran in mind) uses one-based indexing.

The **HDUList** has a useful method **HDUList.info()**, which summarizes the content of the opened FITS file:

Figure 5: The FITS File Handling documentation of Astropy.

2. Make 3 Python scripts which use `astropy.io.fits` module. Describe the design of your Python scripts. Show the source code of your Python scripts. Take screenshots of your computer display to show the result of the execution of your Python scripts.

3. Choose one object frame from the data for this session. Read the header of the FITS file using `astropy.io.fits` module. Make Python scripts to do followings. Show source code of your Python scripts.

   (a) What is the focal length of the telescope written in the header?

   (b) What is the pixel size of the CCD imager written in the header?

   (c) Make a Python script to calculate pixel scale.

   (d) What is the pixel scale written in the header? Compare the pixel scale written in the header to the one you calculated.

   (e) What is the pixel number of the CCD imager?

   (f) Make a Python script to calculate the field-of-view of the CCD imager.

4. Choose one object frame from the data for this session. Read the header of the FITS file using `astropy.io.fits` module. Make Python scripts to do followings. Show source code of your Python scripts.

   (a) What is the date/time of the middle of the exposure?

   (b) Make a Python script to calculate JD corresponding the middle of the exposure (without using Astropy functions).

   (c) What is the JD written in the header? Compare the JD written in the header to the one you calculated.

5. Choose one object frame from the data for this session. Read the header of the FITS file using `astropy.io.fits` module. Make Python scripts to do followings. Show source code of your Python scripts.

   (a) What is the altitude written in the header?

   (b) Make a Python script to calculate the airmass corresponding to the altitude written in the header.

   (c) What is the airmass written in the header? Compare the airmass written in the header to the one you calculated.

6. Choose one dark frame from the data for this session. Read the data using `astropy.io.fits` module. Calculate mean, median, and standard deviation of the image. Describe the design of your Python script. Show the source code of your Python script. Take a screenshot of your computer display to show the result of the execution of your Python script.

7. Choose one bias frame from the data for this session. Read the data using `astropy.io.fits` module. Define four regions of 256 × 256 pixels in the image. Where are those four regions you defined? Visualise locations of four regions. Make a Python script to calculate mean, median, and standard deviation of four regions you defined. Compare the values, and discuss the uniformity of the bias frame. Describe the design of your Python script. Show the source code of your Python script. Take a screenshot of your computer display to show the result of the execution of your Python script.

8. Make a Python script to produce nicely formatted observing log. Think about which information should be included in the observing log, and design your own Python script. Execute the script, and generate the log. Describe the design of your Python script. Show the source code of your Python script. Take a screenshot of your computer display to show the result of the execution of your Python script.

9. Visit the official website of WCSTools (Fig. 6). Install WCSTools on your computer. Summarise the installation process.

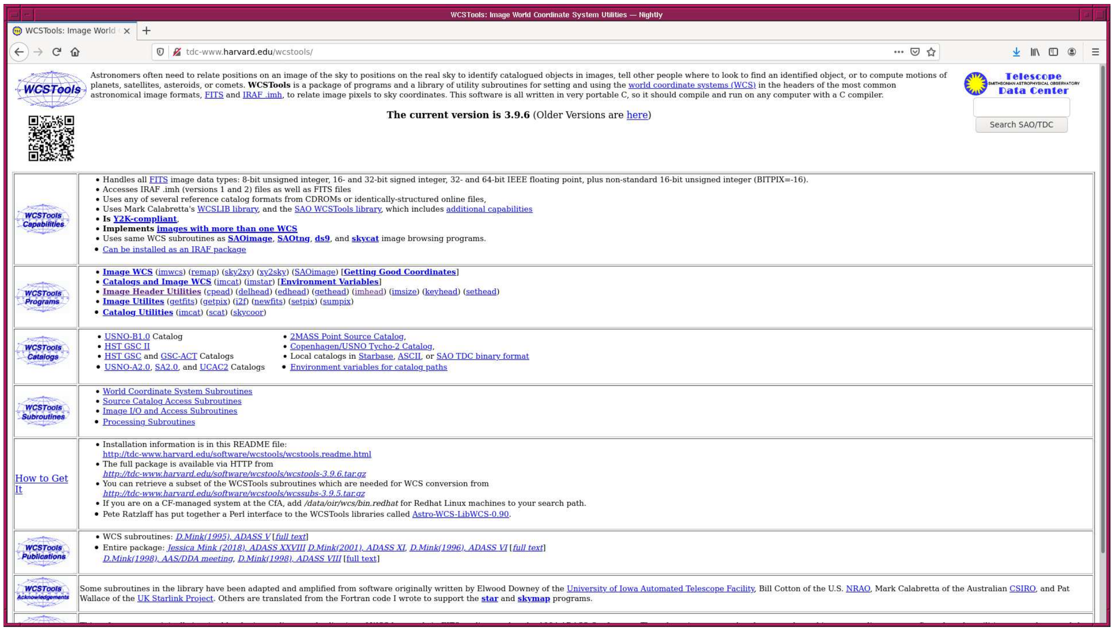
   - `http://tdc-www.harvard.edu/wcstools/`

10. Read the description of "`imhead`" command which is included in WCSTools. Play with it. Make a Python script which works like `imhead`. Describe the design of your Python script. Show the source code of your Python script. Take a screenshot of your computer display to show the result of the execution of your Python script.

    - `http://tdc-www.harvard.edu/software/wcstools/imhead.html`

11. Read the description of "`gethead`" command which is included in WCSTools. Play with it. Make a Python script which works like `gethead`. Describe the design of your Python script. Show the source code of your Python script. Take a screenshot of your computer display to show the result of the execution of your Python script.

    - `http://tdc-www.harvard.edu/software/wcstools/gethead/`



Astronomers often need to relate positions on an image of the sky to positions on the real sky to identify catalogued objects in images, tell other people where to look to find an identified object, or to compute motions of planets, satellites, asteroids, or comets. **WCSTools** is a package of programs and a library of utility subroutines for setting and using the world coordinate systems (WCS) in the headers of the most common astronomical image formats, FITS and IRAF .imh, to relate image pixels to sky coordinates. This software is all written in very portable C, so it should compile and run on any computer with a C compiler.

The current version is 3.9.6 (Older Versions are here)

- Handles all FITS image data types: 8-bit unsigned integer, 16- and 32-bit signed integer, 32- and 64-bit IEEE floating point, plus non-standard 16-bit unsigned integer (BITPIX=-16).
- Accesses IRAF .imh (versions 1 and 2) files as well as FITS files
- Uses any of several reference catalog formats from CDROMs or identically-structured online files,
- Uses Mark Calabretta's WCSLIB library, and the SAO WCSTools library, which includes additional capabilities
- **Is Y2K-compliant**,
- **Implements images with more than one WCS**
- Uses same WCS subroutines as **SAOimage**, **SAOtng**, **ds9**, and **skycat** image browsing programs.
- Can be installed as an IRAF package

- **Image WCS** (imwcs) (remap) (sky2xy) (xy2sky) (SAOimage) [**Getting Good Coordinates**]
- **Catalogs and Image WCS** (imcat) (imstar) [**Environment Variables**]
- **Image Header Utilities** (cpead) (delhead) (edhead) (gethead) (imhead) (imsize) (keyhead) (sethead)
- **Image Utilites** (getfits) (getpix) (i2f) (newfits) (setpix) (sumpix)
- **Catalog Utilities** (imcat) (scat) (skycoor)

- USNO-B1.0 Catalog
- HST GSC II
- HST GSC and GSC-ACT Catalogs
- USNO-A2.0, SA2.0, and UCAC2 Catalogs
- 2MASS Point Source Catalog,
- Copenhagen/USNO Tycho-2 Catalog,
- Local catalogs in Starbase, ASCII, or SAO TDC binary format
- Environment variables for catalog paths

- World Coordinate System Subroutines
- Source Catalog Access Subroutines
- Image I/O and Access Subroutines
- Processing Subroutines

- Installation information is in this README file:
  http://tdc-www.harvard.edu/software/wcstools/wcstools.readme.html
- The full package is available via HTTP from
  http://tdc-www.harvard.edu/software/wcstools/wcstools-3.9.6.tar.gz
- You can retrieve a subset of the WCSTools subroutines which are needed for WCS conversion from
  http://tdc-www.harvard.edu/software/wcstools/wcssubs-3.9.5.tar.gz
- If you are on a CF-managed system at the CfA, add */data/oir/wcs/bin.redhat* for Redhat Linux machines to your search path.
- Pete Ratzlaff has put together a Perl interface to the WCSTools libraries called Astro-WCS-LibWCS-0.90.

- WCS subroutines: *D.Mink(1995), ADASS V* [*full text*]
- Entire package: *Jessica Mink (2018), ADASS XXVIII* *D.Mink(2001), ADASS XI*, *D.Mink(1996), ADASS VI* [*full text*]
  D.Mink(1998), AAS/DDA meeting, *D.Mink(1998), ADASS VIII* [full text]

Some subroutines in the library have been adapted and amplified from software originally written by Elwood Downey of the University of Iowa Automated Telescope Facility, Bill Cotton of the U.S. NRAO, Mark Calabretta of the Australian CSIRO, and Pat Wallace of the UK Starlink Project. Others are translated from the Fortran code I wrote to support the **star** and **skymap** programs.

Figure 6: The official website of WCSTools.