fitsutils

February 1, 2016

Abstract

Library of Fortran 90 and Perl utilities used to handle FITS files.

1 Description

1.1 Introduction

This library will consist of a series of Fortran 90 modules useful for handling FITS files. At the first delivery (Ver. 1.0), there is one Fortran library and one Perl library are available. Each module is described in a separate section below. Each module contains a separate file and will be compiled separately. The resulting object files will be combined into a single library file.

1.2 Module index

- Section 2: `fits_utils`:
- Section 3: `Fitsplutils (Perl)`:
- Section 4: `WcsKey (Perl)`
- Section 5: `MultipleCoordsSys (Perl)`
- Section 6: `FitsCelCoordsSys (Perl)`

2 General-purpose FITS-processing utilities

Module name: `fits_utils`

2.1 Statistical information

2.1.1 getFitsVecStatInfo???

This function is a wrapper for the function `ssclib/array_utils/getAryStatInfo` for a 1-dimensional array, and so returns the structure `aryStatInfo???T` (See the documentation of `ssclib`), which contains the statistical information of the array.

The part '???' in the function names is either Double, Single, Int32/16/8.

The following is an example interface for the Double-type one. In other types, only the difference is the type of the returned variable (`aryStatInfo???T`).

```
interface getFitsVecStatInfoDouble

  function getFitsVecStatDoubleName(setTabName, colName &
     , minAreaIndices, maxAreaIndices, valLower, valUpper, extNum, arMaskIn &
     , flagInfo) result(aryStatInfo)

    ! integer, parameter :: rankArin = 1
    type(aryStatInfoDoubleT) :: aryStatInfo     ! defined in array_utils
    character(*), intent(in) :: setTabName      ! "FITS.ds" or "FITS.ds:TAB"
    character(*), intent(in) :: colName
    integer(int32), intent(in), optional :: minAreaIndices(rankArin), maxAreaIndices(rankArin)
    real(double), intent(in), optional :: valLower, valUpper
    integer, intent(in), optional :: extNum
    logical, intent(in), optional :: arMaskIn(:)
    type(aryStatInfoFlagT), intent(in), optional :: flagInfo
  end function getFitsVecStatDoubleName

  function getFitsVecStatDoubleNameSimp(fitsSetTabName, colName &
     , minAreaIndex, maxAreaIndex, valLower, valUpper, extNum, arMaskIn &
     , flagInfo) result(aryStatInfo)

    ! integer, parameter :: rankArin = 1
    type(aryStatInfoDoubleT) :: aryStatInfo     ! defined in array_utils
    character(*), intent(in) :: fitsSetTabName  ! "FITS.ds" or "FITS.ds:TAB"
    character(*), intent(in) :: colName
    integer(int32), intent(in) :: minAreaIndex, maxAreaIndex
    real(double), intent(in), optional :: valLower, valUpper
    integer, intent(in), optional :: extNum
    logical, intent(in), optional :: arMaskIn(:)
    type(aryStatInfoFlagT), intent(in), optional :: flagInfo
  end function getFitsVecStatDoubleNameSimp

  function getFitsVecStatDoubleSet(inSet, colName, tabName, extNum &
     , minAreaIndices, maxAreaIndices, valLower, valUpper, arMaskIn &
     , flagInfo, strFileInfo) result(aryStatInfo)

    !integer, parameter :: rankArin = 1
    type(aryStatInfoDoubleT) :: aryStatInfo     ! defined in array_utils
    type(DataSetT), intent(in) :: inSet
    character(*), intent(in) :: colName
    character(*), intent(in), optional :: tabName       ! or extNum
    integer, intent(in), optional :: extNum             ! or tabName
    integer(int32), intent(in), optional :: minAreaIndices(rankArin), maxAreaIndices(rankArin)
    real(double), intent(in), optional :: valLower, valUpper
    logical, intent(in), optional :: arMaskIn(:)
    type(aryStatInfoFlagT), intent(in), optional :: flagInfo
    character(*), intent(in), optional :: strFileInfo
  end function getFitsVecStatDoubleSet

  function getFitsVecStatDoubleSetSimp(inSet, colName, tabName, extNum &
```

```
     , minAreaIndex, maxAreaIndex, valLower, valUpper, arMaskIn &
     , flagInfo, strFileInfo) result(aryStatInfo)

   !integer, parameter :: rankArin = 1
   type(aryStatInfoDoubleT) :: aryStatInfo     ! defined in array_utils
   type(DataSetT), intent(in) :: inSet
   character(*), intent(in) :: colName
   character(*), intent(in), optional :: tabName       ! or extNum
   integer, intent(in), optional :: extNum             ! or tabName
   integer(int32), intent(in) :: minAreaIndex, maxAreaIndex
   real(double), intent(in), optional :: valLower, valUpper
   logical, intent(in), optional :: arMaskIn(:)
   type(aryStatInfoFlagT), intent(in), optional :: flagInfo
   character(*), intent(in), optional :: strFileInfo
 end function getFitsVecStatDoubleSetSimp

 function getFitsVecStatDoubleTab(inTab, colName &
    , minAreaIndices, maxAreaIndices, valLower, valUpper, arMaskIn &
    , flagInfo, strFileInfo) result(aryStatInfo)

   !integer, parameter :: rankArin = 1
   type(aryStatInfoDoubleT) :: aryStatInfo     ! defined in array_utils
   type(TableT), intent(in) :: inTab
   character(*), intent(in) :: colName
   integer(int32), intent(in), optional :: minAreaIndices(rankArin), maxAreaIndices(rankArin)
   real(double), intent(in), optional :: valLower, valUpper
   logical, intent(in), optional :: arMaskIn(:)
   type(aryStatInfoFlagT), intent(in), optional :: flagInfo
   character(*), intent(in), optional :: strFileInfo
 end function getFitsVecStatDoubleTab

 function getFitsVecStatDoubleTabSimp(inTab, colName &
    , minAreaIndex, maxAreaIndex, valLower, valUpper, arMaskIn &
    , flagInfo, strFileInfo) result(aryStatInfo)

   !integer, parameter :: rankArin = 1
   type(aryStatInfoDoubleT) :: aryStatInfo     ! defined in array_utils
   type(TableT), intent(in) :: inTab
   character(*), intent(in) :: colName
   integer(int32) :: minAreaIndex, maxAreaIndex
   real(double), intent(in), optional :: valLower, valUpper
   logical, intent(in), optional :: arMaskIn(:)
   type(aryStatInfoFlagT), intent(in), optional :: flagInfo
   character(*), intent(in), optional :: strFileInfo
 end function getFitsVecStatDoubleTabSimp

end interface
```

Note that the ranks of arMaskIn (if specified) and that of the input FITS file have to be identical.

2.1.2 getFitsImgStatInfo???

This function is a wrapper for the function `ssclib/array_utils/getAryStatInfo` for a 2-dimensional array, and so returns the structure `aryStatInfo???T` (See the documentation of `ssclib`), which contains the statistical information of the array.

The part '???' in the function names is either Double, Single, Int32/16/8.

The following is an example interface for the Double-type one. In other types, only the difference is the type of the returned variable (`aryStatInfo???T`).

```
interface getFitsImgStatInfoDouble
  function getFitsImgStatDoubleName(imageSetName, arMaskIn &
     , minAreaIndices, maxAreaIndices, valLower, valUpper &
     , flagInfo) result(aryStatInfo)

    ! integer, parameter :: rankArin = 2
    type(aryStatInfoDoubleT) :: aryStatInfo ! defined in array_utils
    character(*), intent(in) :: imageSetName
    logical, intent(in), optional :: arMaskIn(:,:)
    integer(int32), intent(in), optional :: minAreaIndices(rankArin), maxAreaIndices(rankArin)
    real(double), intent(in), optional :: valLower, valUpper
    type(aryStatInfoFlagT), intent(in), optional :: flagInfo
  end function getFitsImgStatDoubleName

  function getFitsImgStatDoubleSet(imageSet, arMaskIn &
     , minAreaIndices, maxAreaIndices, valLower, valUpper &
     , flagInfo) result(aryStatInfo)

    ! integer, parameter :: rankArin = 2 ! = size(lbound(arin))
    type(aryStatInfoDoubleT) :: aryStatInfo ! defined in array_utils
    type(DataSetT), intent(in) :: imageSet
    logical, intent(in), optional :: arMaskIn(:,:)
    integer(int32), intent(in), optional :: minAreaIndices(rankArin), maxAreaIndices(rankArin)
    real(double), intent(in), optional :: valLower, valUpper
    type(aryStatInfoFlagT), intent(in), optional :: flagInfo
  end function getFitsImgStatDoubleSet
end interface
```

Note that the ranks of arMaskIn (if specified) and that of the input FITS image have to be identical.

2.1.3 getAnnularMask

This function is a wrapper for the function `ssclib/array_utils/getAnnularMaskAry` (See the documentation of `ssclib`), accepting an input 2-dimensional FITS image, and returns a 2-dimensional logical mask array. The input FITS image just provides the size and axes of the returned mask.

```
interface getAnnularMask
  function getAnnularMaskDoubleName(imageSetName, centX, centY &
     , rOuter, rInner, extNum) result(arMask)
    logical, allocatable :: arMask(:,:) ! result
```

```
    character(*), intent(in) :: imageSetName
    real(double), intent(in) :: centX, centY, rOuter
    real(double), intent(in), optional :: rInner ! 0 in default.
    integer(int32), intent(in), optional :: extNum ! Extention number of the image array in the input
  end function getAnnularMaskDoubleName

  function getAnnularMaskDoubleSet(imageSet, centX, centY &
     , rOuter, rInner, extNum) result(arMask)
    logical, allocatable :: arMask(:,:) ! result
    type(DataSetT), intent(in) :: imageSet
    real(double), intent(in) :: centX, centY, rOuter
    real(double), intent(in), optional :: rInner ! 0 in default.
    integer(int32), intent(in), optional :: extNum ! Extention number of the image array in the input
  end function getAnnularMaskDoubleSet

  function getAnnularMaskSingleName(imageSetName, centX, centY &
     , rOuter, rInner, extNum) result(arMask)
    logical, allocatable :: arMask(:,:) ! result
    character(*), intent(in) :: imageSetName
    real(single), intent(in) :: centX, centY, rOuter
    real(single), intent(in), optional :: rInner ! 0 in default.
    integer(int32), intent(in), optional :: extNum ! Extention number of the image array in the input
  end function getAnnularMaskSingleName

  function getAnnularMaskSingleSet(imageSet, centX, centY &
     , rOuter, rInner, extNum) result(arMask)
    logical, allocatable :: arMask(:,:) ! result
    type(DataSetT), intent(in) :: imageSet
    real(single), intent(in) :: centX, centY, rOuter
    real(single), intent(in), optional :: rInner ! 0 in default.
    integer(int32), intent(in), optional :: extNum ! Extention number of the image array in the input
  end function getAnnularMaskSingleSet
end interface
```

3 Wrapper utilities to handle FITS files in Perl

Module name: `Fitsplutils`

See the header of the library code for detail. You may want to read it by, for example,
`cd /YOUR/DIR; pod2man Fitsplutils.pm | tbl | neqn | nroff -h -man | less`

4 Module to define the FITS WCS keywords

Module name: `WcsKey`

See the header of the library code for detail. You may want to read it by, for example,
`cd /YOUR/DIR; pod2man WcsKey.pm | tbl | neqn | nroff -h -man | less`

5 Class to represent multiple coordinate systems

Module name: `MultipleCoordsSys`

See the header of the library code for detail. You may want to read it by, for example,
`cd /YOUR/DIR; pod2man MultipleCoordsSys.pm | tbl | neqn | nroff -h -man | less`

6 Class representing the celestial coordinate system in a FITS file

Module name: `FitsCelCoordsSys`

See the header of the library code for detail. You may want to read it by, for example,
`cd /YOUR/DIR; pod2man FitsCelCoordsSys.pm | tbl | neqn | nroff -h -man | less`

References