SIC tutorial

Presentation by
Sébastien BARDEAU
(IRAM/Grenoble)

Current kernel developers: Jérôme PETY,
Sébastien BARDEAU, & Stéphane GUILLOTEAU
on behalf of the GILDAS developers over time

9th IRAM Millimeter Interferometry School
Oct. 10-14 2016, St Martin d'Hères

Table of contents

Table of contents (cont'd)

Basics

Basics: I. Need help?

GILDAS web page http://www.iram.fr/IRAMFR/GILDAS/

Online help

```
SIC> HELP ! Summary of all commands, gathered by language
SIC\
  @             BEGIN         BREAK         CONTINUE      EDIT          ELSE
...
LINEDB\
  USE           SELECT        LIST          INSERT        REMOVE
Additional help available (eg summaries):
  SIC\          VECTOR\       ADJUST\       LINEDB\
SIC> HELP SIC\ ! (with backslash) Language help with short command description
                     SIC\ Command Language Summary
ACCEPT         : Read variable in various format.
BEGIN          : Begin a sub-procedure, help file or data file.
BREAK          : Exit without error from a FOR-NEXT loop.
COMPUTE        : Execute non-arithmetic operations on variables.
...
SIC> HELP COMPUTE  ! Command help
    [SIC\]COMPUTE  OutVar  OPERATION  InVar [Parameters] [/BLANKING Bval
[Eval]]

Perform operations or transformations on variables that are not directly
supported by the array capabilities of the SIC command interpretor. Out-
Var is the output variable, InVar the input variable.
...
SIC> HELP HELP  ! All possible helps explained here
```

Helpdesk (Questions? Comments? Bug reports?): gildas@iram.fr

Basics: II. Getting started

```
shell-prompt> sic -h
GILDAS Version: dev (16mar16 10:33) (x86_64-redhat6.4-ifort) source tree
Usage: sic [OPTIONS] [COMMAND-LINE]

Options:
  -nl: No Log files
    No log and message files will be produced in ~/.gag/logs/
    for this session. For a permanent effect,  put  the  line
    "SIC_LOGFILES NO" in ~/.gag.dico
  -d: Debug
    Start the program with debug messages enabled.  They  can
    be disabled or reenabled later on with  the  command  SIC
    DEBUG MESSAGE
  -v: Version
    Display the version information
  -h: Help
    Display this help

Batch mode:
  shell> [nohup] sic < readfrom.pro > writeto.log 2>&1 &
  "nohup" should not be required if  you  really  redirect
  STDIN, STDOUT, and STDERR as in the above example.
  The command "exit" is not  needed  at  the  end  of  the
  file "readfrom.pro" since a EOF signal will terminate
  the process.
```

Basics: II. Getting started

```
shell-prompt> sic
GILDAS Version: dev (16mar16 10:33) (x86_64-redhat6.4-ifort) source tree

 * Welcome to SIC

 * Loaded modules
    sic (J.Pety, S.Bardeau, S.Guilloteau, E.Reynier)

 * In charge:               J.Pety, S.Bardeau
   Active developers:       S.Guilloteau, E.Reynier
   Main past contributors: F.Badia,D.Broguière,G.Buisson,G.Duvert,
                            T.Forveille,R.Gras,R.Lucas,G.Mella,P.Valiron

 * Questions? Comments? Bug reports? Mail to: gildas@iram.fr

SIC>


shell-prompt> sic help
GILDAS Version: dev (16mar16 10:33) (x86_64-redhat6.4-ifort) source tree
...
HELP should be available on the following commands :
SIC\
  @            BEGIN        BREAK        CONTINUE     EDIT         ELSE
...
SIC>
```

Basics: III. Abbreviations and ambiguities

Commands can be abbreviated:

```
SIC> examine pi
PI              =     3.141592653589793      ! Double  GLOBAL RO
SIC> exa pi
PI              =     3.141592653589793      ! Double  GLOBAL RO
SIC> ex pi
E-PARSE,  Ambiguous command could be :
SIC\EXIT           SIC\EXAMINE        SIC\EXECUTE
```

Options too:

```
SIC> exa pi /a
E-PARSE,  Ambiguous option for command EXAMINE could be :
/ADDRESS           /ALIAS
```

But not the variables!

```
SIC> exa p
E-EXAMINE,  No such variable P
W-SIC,  Ambiguous Name, Could be:
PI
```

Keywords: list and disambiguisation

```
SIC> define ?
I-DEFINE,  Choices are:
  REAL       DOUBLE     INTEGER    LONG       LOGICAL    CHARACTER  COMPLEX
  TABLE      STRUCTURE  IMAGE      HEADER     FUNCTION   COMMAND    FITS
  UVTABLE    ALIAS      LANGUAGE
SIC> define int a
```

Basics: IV. Explicitly or implicitly spawning shell commands

Commands can be spawned to the shell by prefixing them with $ (quick access):

```
SIC> $ls
pro  slides  tools
SIC> $date
Tue Mar 22 18:49:27 CET 2016
```

Commands can be spawned to the shell by prefixing them with SYSTEM (more flexible):

```
SIC> system "ls -l"
total 11184
-rw------- 1 bardeau astro       8 Apr  7 08:36 demo-greg-2d.aux
-rw------- 1 bardeau astro   22387 Apr  7 08:36 demo-greg-2d.log
...
SIC> define char*32 myfile
SIC> let myfile "demo-sic.pdf"
SIC> system "ls -l "'myfile'
-rw------- 1 bardeau astro 509698 Sep 29 14:20 demo-sic.pdf
```

Some SIC commands all perform operating system commands (more portable):

```
SIC> sic mkdir foo/bar  ! Create directory
SIC> sic dir foo        ! Change working directory
SIC> sic dir            ! Display working directory
I-SIC,  Default directory is /home/pety/gildas-git/tutorials/kernel/foo
SIC> sic rename bar bleu
SIC> sic delete bar
SIC> edit               ! Open an editor
I-EDIT,  Writing stack content on stack.sic
I-EDIT,  Using "emacsclient --no-wait" editor
```

Basics: V. Permanent customizations

Some customizations can be defined for all sessions. Two major ways:

$HOME/.gag.dico Put here logical name and value pairs, available to all programs and all sessions. For example:

```
home/user> cat .gag.dico
macro#dir:      ./;./pro;gag_pro:;
sic_logfiles    no
```

$HOME/.gag/init/init.sic (or init.greg, etc) Init files are real script files executed like any other one, for all sessions of the given program, and for programs depending on it. For example:

```
home/user> cat .gag/init/init.sic
sic log macro#dir: "./;./pro/;"‘macro#dir:‘   ! See slide "Procedures: II"
sic help scroll                               ! Scroll mode for HELP command
sic message * l+d                             ! Add debug messages in log file
symbol lrt "system ""ls --color=tty -lrt"""   ! Create a custom shortcut for shell "ls"
```

Basics: VI. Efficient loops

Examples

1. Defining all integers from 1 to N

```
define integer a[100]
let a[i] i
```

2. Computing a 2D gaussian model

```
! Define the gaussian parameters
define integer np
let np 1024
define real xc yc amp sig z[np,np]
let xc np*0.5
let yc np*0.5
let amp 1.0
let sig np/5.0
! Track computing time
sic cpu
! Compute the gaussian model
! for y 1 to np
!  for x 1 to np
    let z[x,y] amp*exp(-(((x-xc)^2+(y-yc)^2))/2/sig^2)
!  next x
! next y
! Report computing time
sic cpu
exa sic%cpu%raw%
```

Results

Non-vectorized

```
SIC%CPU%RAW%   ! Structure GLOBAL
SIC%CPU%RAW%ELAPSED =     33.757305
SIC%CPU%RAW%USER =     29.925451
SIC%CPU%RAW%SYSTEM =     3.484470
```

Vectorized ∼1000 times faster!

```
SIC%CPU%RAW%   ! Structure GLOBAL
SIC%CPU%RAW%ELAPSED =    3.066086E-02
SIC%CPU%RAW%USER =    2.799600E-02
SIC%CPU%RAW%SYSTEM =    2.999999E-03
```

Procedures

Procedures: I. Editing

Invoking the editor The procedure files can be edited with your favorite text/program editor in your working environment. You can also briefly open-edit-close a file in the session with the EDIT command:

```
SIC> edit procedure-file-name
I-EDIT,  Using "emacs" editor
```

The editor can be customized with the SIC EDIT command.

Syntax highlighting GILDAS provides syntax highlighting configuration files.

Emacs editor Put the following lines in your $HOME/.emacs configuration file.

```
; Customize emacs to work pretty well with GILDAS procedures
(setq load-path (cons
                 "~/gildas/gildas-src-sep16b/admin"
                 load-path))
(load "gildas")
```

This loads the file gildas.el in Emacs.

KDE editors (Kate or Kwrite) Copy (or link) the configuration file like this:

```
cp  /gildas/gildas-src-sep16b/admin/gildas.xml $HOME/.local/share/katepart5/syntax/
```

Procedures: II. Executing (1/2)

```
SIC> @ procedure-file-name
```

Procedures: III. Extensions

Standard extensions: procedures are usually suffixed with

.sic contains commands understood by Sic,

.greg contains commmands understood by Greg,

and so on.

Non-standard extensions: the procedure extension is free, e.g. myprocedure.sic~, myprocedure.sic.bak, myprocedure.cmd... can be executed.

Default extensions:

extension can be omitted. Sic will try several extensions from a predefined list:

```
SIC> sic extension
I-SIC,  Default macro extensions: .sic .pro
```

default list depends on program, e.g.

```
MRTCAL> sic extension
I-SIC,  Default macro extensions: .cal .class .greg .sic .pro
```

default list can be enriched:

```
SIC> sic extension .cmd
SIC> sic extension
I-SIC,  Default macro extensions: .cmd .sic .pro
```

Procedures: IV. Paths

Procedures can be prefixed with a path, absolute or relative, e.g.

```
SIC> @ /home/me/pro/hello.sic
SIC> @ pro/hello.sic
```

Default search paths can be omitted: search first in current working directory and then in GILDAS defined procedures. macro#dir: is a logical value storing the paths where to search for procedures ⇒ user customizable.

```
SIC> sic logical macro#dir:                              ! Display current value
macro#dir: = ./;gag_pro:;
SIC> sic logical macro#dir: "./;./pro/;"`macro#dir:`     ! Set new value
SIC> sic logical macro#dir:                              ! Check new value
macro#dir: = ./;./pro/;gag_pro:;
SIC> @ hello.sic
Hello
```

SIC WHICH ProcName tells which file will be executed when executing @ ProcName.

```
SIC> sic which hello.sic
./pro/hello.sic
```

Procedures: II. Executing (2/2)

```
SIC> @ hello          ! You can omit both path and extension
Hello

SIC> sic which hello  ! Check which one is found
./pro/hello.sic

SIC> type hello       ! Show the contents of the file
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
say "Hello"
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Procedures: V. Arguments - 1. The easy way

Procedure arguments are passed as character strings and retrieved as tokens &1, &2, ... (9 at most). Tokens are replaced before any other SIC interpretation

```
SIC> @ pro/abs2off 05:40:54 -02:28:35.00    05:40:54.270 -02:28:00.00

(-4.05,-35)

SIC> type abs2off.sic
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! @ abs2off
!   Convert from RA1,Dec1 coordinates to offsets from RA2,Dec2 projection
!   center assuming a RADIO projection
! &1 (1st argument): ra1
! &2 (2nd argument): dec1
! &3 (4rd argument): ra2
! &4 (4th argument): dec2
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
define double sec_per_rad
let sec_per_rad 180*3600/pi
define double ra1 dec1
define double ra2 dec2
let  ra1 "&1" /sexa r h ! Translate an input string from Hours  to Radian
let  ra2 "&3" /sexa r h
let dec1 "&2" /sexa r d ! Translate an input string from degree to Radian
let dec2 "&4" /sexa r d
! Answer to user, taking care of the projection and the unit conversion
say ""
say "("'nint(100*(ra1-ra2)*cos(dec1)*sec_per_rad)/100'","'(dec1-dec2)*sec_per_rad'")"
say ""
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Procedures: V. Arguments - 2. The flexible way

pro%narg Number of input arguments (up to 32)

pro%arg[iarg] iargth argument of procedure in character string format.

```
SIC> @ pro/abs2off
E-ABS2OFF  Usage: @ abs2off RA1 Dec1  RA2 Dec2
SIC> type abs2off.sic
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! @ abs2off
!   Convert from RA1,Dec1 coordinates to offsets from RA2,Dec2 projection
!   center assuming a RADIO projection
! 1st argument: ra1
! 2nd argument: dec1
! 4rd argument: ra2
! 4th argument: dec2
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
if (pro%narg.lt.4) then
   message e abs2off "Usage: @ abs2off RA1 Dec1  RA2 Dec2"
   return base
endif
define double sec_per_rad
let sec_per_rad 180*3600/pi
define double ra1 dec1
define double ra2 dec2
let  ra1 'pro%arg[1]' /sexa r h
let  ra2 'pro%arg[2]' /sexa r h
let dec1 'pro%arg[3]' /sexa r d
let dec2 'pro%arg[4]' /sexa r d
say ""
say "("'nint(100*(ra1-ra2)*cos(dec1)*sec_per_rad)/100'","'(dec1-dec2)*sec_per_rad'")"
say ""
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Procedures: VI. Local vs global scope for variables

```
SIC> type car2pol
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! @ car2pol
! Input &1: x0 coordinate along the x axis
! Input &2: y0 coordinate along the y axis
! Output stored in the polar structure (polar%rho and polar%theta)
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
if .not.exist(polar) then
   define structure polar /global
   define double polar%rho polar%theta /global
endif
define double cos sin ! Local variables
let polar%rho sqrt(&1^2+&2^2)
let cos &1/polar%rho
let sin &2/polar%rho
let polar%theta atan2(sin,cos)
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SIC>
SIC> @ car2pol 1.5 2.0
SIC> exa polar
POLAR           ! Structure GLOBAL
SIC> exa polar%
POLAR%          ! Structure GLOBAL
POLAR%THETA      =    0.9272952180016123      ! Double  GLOBAL
POLAR%RHO        =     2.500000000000000      ! Double  GLOBAL
SIC> exa cos sin
E-EXAMINE,  No such variable COS
```

Procedures: VII. Structuring in blocks - 1. Example

```
SIC> @ brightness-tools
I-BRIGHT,  Brightness tool initialization complete
SIC> exa bright%
BRIGHT%         ! Structure GLOBAL
BRIGHT%JYPERK    =     0.000000000000000      ! Double  GLOBAL
BRIGHT%KPERJY    =     0.000000000000000      ! Double  GLOBAL
SIC> @ bright-jansky 115.3  22.5 22.5 ! IRAM-30m values.
Wavelength: 2.6 mm, Beam: 13.5 x 13.5
K per Jy/Beam:  0.18167472815151
mJy/beam per K: 5.504342899944E+03
SIC> @ bright-jansky 115.3  1.2 0.8 ! NOEMA values.
Wavelength: 2.6 mm, Beam: 0.7 x 0.5
K per Jy/Beam:  95.805032423647
mJy/beam per K: 10.437865054709
SIC> @ bright-jansky 115.3  0.02 0.015 ! ALMA values.
Wavelength: 2.6 mm, Beam: 0.012 x 0.09
K per Jy/Beam:  3.0657610375567E+04
mJy/beam per K: 0.032618328295964
```

Procedures: VII. Structuring in blocks - 2. Howto

```
SIC> type brightness-tools ! List the content of tools/brightness-tools.sic
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
begin procedure bright-def
  if not.exist(bright) then
     define structure bright /global
     define double bright%kperjy bright%jyperk /global
  endif
end procedure bright-def
!
begin procedure bright-jansky
  ! Compute conversion factor between Jy/Beam and K.
  define double nu theta[2] lambda rad_per_sec
  let nu 'pro%arg[1]' ! [GHz]
  let nu nu*1e9 ! [Hz]
  let theta 'pro%arg[2]' 'pro%arg[3]'
  let rad_per_sec pi/(180*3600)
  let theta theta*rad_per_sec       ! Convert from arcsec to radian
  let theta theta/2./sqrt(log(2.0)) ! Convert from FWHM to 1/e width
  let lambda 2.99792458e8/nu
  let bright%jyperk 2.0*1.38e3*pi*theta[1]*theta[2]/lambda^2
  let bright%kperjy 1/bright%jyperk
  ! User Feedback
  say "Wavelength: "'nint(1e5*lambda)/100'" mm, Beam: "'nint(1000*theta[1]/rad_per_sec)/1000'" x "'nint(1
  say "K per Jy/Beam:  "'bright%kperjy'
  say "mJy/beam per K: "'1e3*bright%jyperk'
end procedure bright-jansky
!
@ bright-def
message i bright "Brightness tool initialization complete"
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Procedures: VIII. Dealing with errors

Errors can be raised for two mains reasons:

- syntax error (e.g. mispelled command, unknown option, missing argument),
- error during command execution.

```
SIC> type erroneous.sic
say "Entering"
let pi 1
say "Leaving"

SIC> @ erroneous.sic
Entering
E-LET,  Readonly variables cannot be modified
I-ERROR,  Occured in ./erroneous.sic at Line    2
SIC_3> let pi 1
```

In case of error:

- an error message is displayed (or a cascade of error messages),
- the backtrace is shown,
- the faulty command is recalled with a dedicated prompt

Your action is required:

1. edit the command line to fix the issue and execute it, then type C (CONTINUE) to resume procedure execution, OR
2. stop the execution with Q (QUIT) as many time as needed, and fix the issue in the procedure file.

Importing data from outside world

Importing data from outside world:
I. ASCII tables

Example file:

```
SIC> type ascii.dat  ! Display the file contents
! A comment line
11 1.23456
22 9.87654
```

If the number of lines is not known, use COMPUTE LINES:

```
SIC> define integer nl
SIC> compute nl lines ascii.dat
SIC> exa nl
NL    =       3        ! Integer GLOBAL
SIC> compute nl lines ascii.dat /blank  ! Ignore comment lines
SIC> exa nl
NL    =       2        ! Integer GLOBAL
```

Read file in array variables with command ACCEPT:

```
SIC> define real a[nl]
SIC> accept * a /column ascii.dat  ! Use * to skip columns to be ignored
SIC> exa a
A               is a real Array     of dimensions  2
  1.234560       9.876540
```

Comment lines are implicitly ignored. Column separator is a group of one or more blanks. Use option /FORMAT for columns with specific formatting.

Importing data from outside world:
II. FITS images, cubes, UV tables

FITS images (2D), cubes (3D and more), and UV tables have a standard structure and can be translated directly to the Gildas Data Format (see next slides) for use with Gildas tools:

```
SIC> sic mkdir tmp

SIC> fits fits/paws-pdbi+30m-12co10-1as-cube.fits to tmp/m51-12co10.lmv
W-FITS,  Reference system set by SPECSYS, ignoring VELREF
I-GIO_WIH,  GDFBIG re-allocation      2

SIC> fits fits/paws-pdbi+30m-12co10-1as-mom0.fits to tmp/m51-12co10.lm
W-FITS,  Reference system set by SPECSYS, ignoring VELREF
W-FITS_GILDAS,  Removing NaN on the fly, using DATAMIN/MAX
```

Remark: Direct support of the CLASS FITS files and HERSCHEL/HIFI FITS files is also available within CLASS (see documentation for details).

Importing data from outside world:
III. Other FITS files

All FITS files (in particular those with a structure which can not be transfered to the Gildas Data Format) can be read in a SIC structure mapping all their Header/Data Units (FITS HDU) and/or XTENSIONs:

```
SIC> define fits f myfile.fits
[Messages and warnings]
SIC> examine f%
...
```

Primary HDU components are available under F%, other HDU/XTENSION are available under F%XXX% (where XXX is the HDU/XTENSION name).

GDF $\simeq$ 1 header + 1 data block

GDF ≃ 1 header + 1 data block:
I. Looking at the header of a LM image

```
SIC> header tmp/m51-12co10.lm
File : tmp/m51-12co10.lm                                        REAL*4
Size        Reference Pixel           Value                  Increment
       935   468.000000000000       0.00000000000000     -1.4544411897077 43E-06
       601   301.000000000000       0.00000000000000      1.4544411897077 43E-06
Blanking value and tolerance      -445.51599       0.0000000
Source name          M51-1
Map unit             K.KM/S
Axis type            RA           DEC
Coordinate system    EQUATORIAL          Velocity    LSR
Right Ascension      13:29:52.532        Declination         47:11:41.98
Lii         104.8522142960191            Bii       68.56109465989114
Equinox             2000.0000
Projection type      AZIMUTHAL           Angle     0.000000000000000
Axis 1     A0        13:29:52.532        Axis 2     D0       47:11:41.98
Minimum             0.0000000      at        1        1
Maximum             445.51596      at      552      246
Axis 0 Line Name     12co10              Rest Frequency   115090.0037000000
Resolution in Velocity     0.0000000     in Frequency     0.000000000000000
Offset in Velocity         0.0000000     Doppler          0.000000000000000
Beam                    1.16               0.970               73.00
NO Noise level
NO Proper motion
```

GDF ≃ 1 header + 1 data block:
II. Looking at the header of a LMV cube

```
SIC> header tmp/m51-12co10.lmv
File : tmp/m51-12co10.lmv                                        REAL*4
Size        Reference Pixel          Value                  Increment
       935   468.000000000000       0.00000000000000     -1.454441189707743E-06
       601   301.000000000000       0.00000000000000      1.454441189707743E-06
       120   60.5000000000000       0.00000000000000       5.00000000000000
Blanking value and tolerance      16.533276      3.12593595E-09
Source name          M51-1
Map unit             K
Axis type            RA           DEC          VELOCITY
Coordinate system    EQUATORIAL          Velocity    LSR
Right Ascension      13:29:52.532        Declination         47:11:41.98
Lii         104.8522142960191            Bii       68.56109465989114
Equinox             2000.0000
Projection type      AZIMUTHAL           Angle     0.000000000000000
Axis 1      A0       13:29:52.532        Axis 2     D0       47:11:41.98
Minimum             0.0000000      at        1        1
Maximum             445.51596      at      552      246
Axis 3 Line Name     12co10              Rest Frequency   115090.0037000000
Resolution in Velocity     5.0000000     in Frequency    -1.919494647527124
Offset in Velocity         0.0000000     Doppler          0.000000000000000
Beam                    1.16               0.970               73.00
NO Noise level
NO Proper motion
```

GDF ≃ 1 header + 1 data block:

III. Loading the header of a LMV cube into a SIC structure

```
SIC> define header head tmp/m51-12co10.lmv read
SIC> exa head
HEAD            ! Header    GLOBAL RO
SIC> exa head%
HEAD%           ! Header    GLOBAL RO
HEAD%RDONLY      = T                          ! Logical GLOBAL RO
HEAD%TELE_SEC    =            0               ! Integer GLOBAL RO
HEAD%PARALLAX    =    0.000000                ! Real    GLOBAL RO
HEAD%MU          is a real Array     of dimensions  2
   0.000000       0.000000
HEAD%PROPER      =            0               ! Integer GLOBAL RO
HEAD%RMS         =    0.000000                ! Real    GLOBAL RO
HEAD%NOISE       =    0.000000                ! Real    GLOBAL RO
HEAD%SIGMA       =            0               ! Integer GLOBAL RO
HEAD%PA          =    1.274090                ! Real    GLOBAL RO
HEAD%MINOR       =   4.7026929E-06            ! Real    GLOBAL RO
HEAD%MAJOR       =   5.6238387E-06            ! Real    GLOBAL RO
HEAD%BEAM        =            3               ! Integer GLOBAL RO
HEAD%VTYPE       =            1               ! Integer GLOBAL RO
HEAD%DOPPLER     =    0.000000                ! Real    GLOBAL RO
HEAD%F_AXIS      =            3               ! Integer GLOBAL RO
HEAD%VELOFF      =    0.000000                ! Real    GLOBAL RO
HEAD%VELRES      =    5.000000                ! Real    GLOBAL RO
HEAD%RESTFRE     =     115090.0037000000      ! Double  GLOBAL RO
HEAD%IMAGFRE     =     0.000000000000000      ! Double  GLOBAL RO
HEAD%FREQRES     =    -1.919494647527124      ! Double  GLOBAL RO
HEAD%LINE        = 12co10                     ! Character*12 GLOBAL RO
HEAD%SPEC        =           14               ! Integer GLOBAL RO
...
```

GDF $\simeq$ 1 header + 1 data block:
III. Loading the header of a LMV cube into a SIC structure

```
...
HEAD%Y_AXIS     =            2              ! Integer GLOBAL RO
HEAD%X_AXIS     =            1              ! Integer GLOBAL RO
HEAD%ANGLE      =     0.000000000000000     ! Double  GLOBAL RO
HEAD%DO         =    0.8237080532122611     ! Double  GLOBAL RO
HEAD%AO         =     3.533748647002357     ! Double  GLOBAL RO
HEAD%PTYPE      =            3              ! Integer GLOBAL RO
HEAD%PROJ       =            9              ! Integer GLOBAL RO
HEAD%EQUINOX    =    2000.000               ! Real    GLOBAL RO
HEAD%BII        =     1.196616840586602     ! Double  GLOBAL RO
HEAD%LII        =     1.830016367472201     ! Double  GLOBAL RO
HEAD%DEC        =    0.8237080532122611     ! Double  GLOBAL RO
HEAD%RA         =     3.533748647002357     ! Double  GLOBAL RO
HEAD%SOURCE     = M51-1                     ! Character*12 GLOBAL RO
HEAD%POSI       =           15              ! Integer GLOBAL RO
HEAD%SYSTEM     = EQUATORIAL                ! Character*12 GLOBAL RO
HEAD%UNIT7      =                           ! Character*12 GLOBAL RO
HEAD%UNIT6      =                           ! Character*12 GLOBAL RO
HEAD%UNIT5      =                           ! Character*12 GLOBAL RO
HEAD%UNIT4      =                           ! Character*12 GLOBAL RO
HEAD%UNIT3      = VELOCITY                  ! Character*12 GLOBAL RO
HEAD%UNIT2      = DEC                       ! Character*12 GLOBAL RO
HEAD%UNIT1      = RA                        ! Character*12 GLOBAL RO
HEAD%UNIT       = K                         ! Character*12 GLOBAL RO
HEAD%DESC       =           24              ! Integer GLOBAL RO
...
```

GDF ≃ 1 header + 1 data block:
III. Loading the header of a LMV cube into a SIC structure

```
...
HEAD%MAXLOC     is a long integer Array     of dimensions  7
                   503                   363                    48
                     0                     0                     0
                     0
HEAD%MINLOC     is a long integer Array     of dimensions  7
                   350                   580                    52
                     0                     0                     0
                     0
HEAD%MAX        =    16.53327               ! Real    GLOBAL RO
HEAD%MIN        =   -10.31831               ! Real    GLOBAL RO
HEAD%EXTREMA    =            6              ! Integer GLOBAL RO
HEAD%BLANK      is a real Array     of dimensions  2
   16.53328      3.1259360E-09
HEAD%BLAN       =            2              ! Integer GLOBAL RO
HEAD%CONVERT    is a double precision Array     of dimensions  3  7
   468.000000000000000       0.00000000000000000     -1.45444118970774347E-06
   301.000000000000000       0.00000000000000000      1.45444118970774347E-06
   60.5000000000000000       0.00000000000000000       5.00000000000000000
   0.00000000000000000       0.00000000000000000       1.00000000000000000
   1.00000000000000000       0.00000000000000000       1.00000000000000000
   1.00000000000000000       0.00000000000000000       1.00000000000000000
   1.00000000000000000       0.00000000000000000       1.00000000000000000
HEAD%DIM        is a long integer Array     of dimensions  7
                   935                   601                   120
                     1                     1                     1
                     1
HEAD%NDIM       =            3              ! Integer GLOBAL RO
HEAD%GENE       =           16              ! Integer GLOBAL RO
```

GDF $\simeq$ 1 header + 1 data block:
IV. Modifying, e.g., the source name

```
SIC> define header head tmp/m51-12co10.lmv read
SIC> exa head%source
HEAD%SOURCE     = M51-1                     ! Character*12 GLOBAL RO
SIC> let head%source "M51"
E-LET,  Readonly variables cannot be modified
SIC> delete /var head
SIC> define header head tmp/m51-12co10.lmv write ! Note the WRITE status
SIC> exa head%source
HEAD%SOURCE     = M51-1                     ! Character*12 GLOBAL
SIC> let head%source "M51"
SIC> header tmp/m51-12co10.lmv
...
Source name         M51-1       ! It did not change on disk yet!
...
SIC> delete /var head           ! The file is only updated on disk at this point!
SIC> header tmp/m51-12co10.lmv
...
Source name         M51         ! Now it did change on disk!
...
```

GDF $\simeq$ 1 header + 1 data block:
V. Loading the header and the data block of a LMV cube into a SIC structure

```
SIC> define image cube tmp/m51-12co10.lmv read
SIC> exa cube /address
CUBE             is a real Array     of dimensions  935  601  120
-11  47709906011152 3  935  601  120  1  1  1  1  67432200  1 T 0
SIC> exa cube[1] /ad
CUBE[1]          is a real Array     of dimensions  935  601
   -11  47709906011152 2  935  601  0  0  0  0  0  561935  1 T 0
SIC> exa cube[1,1] /ad
CUBE[1,1]        is a real Array     of dimensions  935
   -11  47709906011152 1  935  0  0  0  0  0  0  935  1 T 0
SIC> exa cube[1,1,1] /ad
CUBE[1,1,1]      is a real Array     of dimensions
   -11  47709906011152 0  0  0  0  0  0  0  0  1  1 T 0
SIC> exa cube[1,1,1]
CUBE[1,1,1]      is a real Array     of dimensions
   16.53328
SIC> exa cube%
CUBE%            is a real Array     of dimensions  935  601  120
CUBE%RDONLY     = T                          ! Logical GLOBAL RO
CUBE%TELE_SEC   =            0               ! Integer GLOBAL RO
CUBE%PARALLAX   =    0.000000                ! Real    GLOBAL RO
...
SIC> exa cube[1,11:20,1]
CUBE[1,11:20,1]  is a real Array     of dimensions  10
   16.53328       16.53328       16.53328       16.53328       16.53328
   16.53328       16.53328       16.53328       16.53328       16.53328
```

GDF ≃ 1 header + 1 data block:
VI. Changing, e.g., the blanking value

```
SIC> define image cube tmp/m51-12co10.lmv read
SIC> exa cube[1,11:20,1]
CUBE[1,11:20,1]  is a real Array     of dimensions  10
   16.53328       16.53328       16.53328       16.53328       16.53328
   16.53328       16.53328       16.53328       16.53328       16.53328
SIC> let cube -1000 /where abs(cube-cube%blank[1]).le.cube%blank[2]
E-LET,  Readonly variables cannot be modified
SIC> delete /var cube
SIC> define image cube tmp/m51-12co10.lmv write ! Note the WRITE status
SIC> exa cube[1,11:20,1]
CUBE[1,11:20,1]  is a real Array     of dimensions  10
   16.53328       16.53328       16.53328       16.53328       16.53328
   16.53328       16.53328       16.53328       16.53328       16.53328
SIC> let cube -1000 /where abs(cube-cube%blank[1]).le.cube%blank[2]
SIC> let cube%blank[1] -1000
SIC> let cube%blank[2]     0
SIC> header tmp/m51-12co10.lmv
...
Blanking value and tolerance      16.533276      3.12593595E-09 ! It did not change on disk yet!
...
SIC> delete /var cube                          ! The file is only updated on disk at this point!
SIC> header tmp/m51-12co10.lmv
...
Blanking value and tolerance      -1000.0000       0.0000000    ! Now it did change on disk!
...
SIC> define image cube tmp/m51-12co10.lmv read
SIC> exa cube[1,11:20,1]
   -1000.000      -1000.000      -1000.000      -1000.000      -1000.000
   -1000.000      -1000.000      -1000.000      -1000.000      -1000.000
SIC> delete /var cube
```

GDF $\simeq$ 1 header + 1 data block:
VII. Importing data, changing the blanking value, and modifying the header in a single procedure

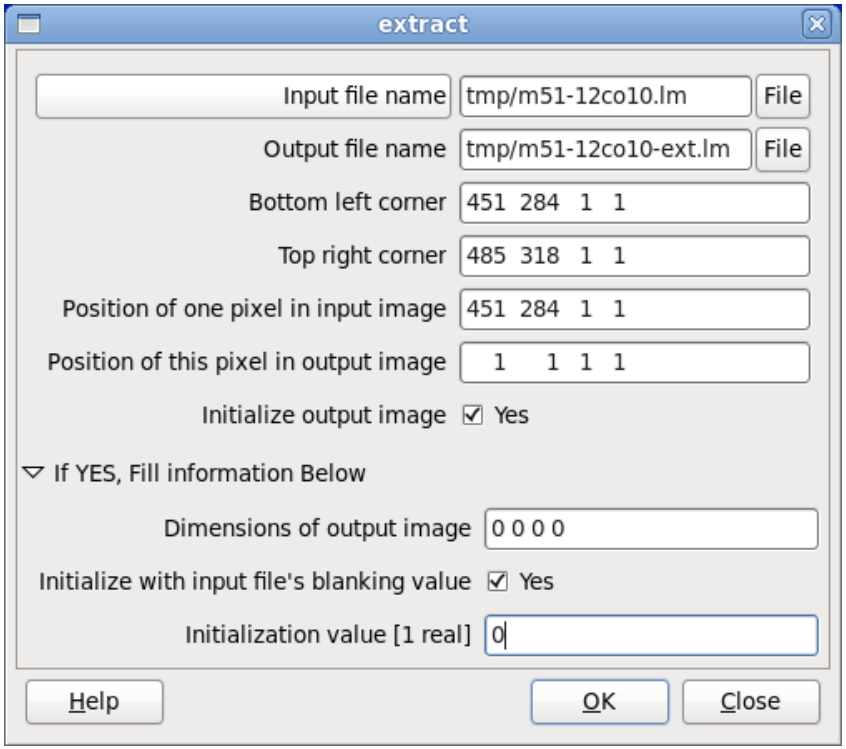
```
SIC> type import ! List the content of pro/import.sic
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
begin procedure import-modify
  define real blank[2]
  let blank[1] &3
  let blank[2] &4
  define image ima "tmp/&1" write
  let ima blank[2] /where abs(ima-ima%blank[1]).le.ima%blank[2]
  let ima%blank blank ! Directly copy the full array
  let ima%source "&2"
  delete /var ima
  header "tmp/&1"
end procedure import-modify
!
begin procedure import-fits
  sic mkdir tmp
  fits "fits/&1.fits" to "tmp/&2"
end procedure import-fits
!
begin procedure import-data
  @ import-fits &1 &2
  @ import-modify &2 M51 -1000 0
end procedure import-data
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
@ import-data  paws-pdbi+30m-12co10-1as-cube  m51-12co10.lmv
@ import-data  paws-pdbi+30m-12co10-1as-mom0  m51-12co10.lm
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
SIC> @ import
...
```

GILDAS Tasks

GILDAS Tasks: I. Generalities

Definition Standalone program with standardized input parameters so that the program can be called through the RUN command.

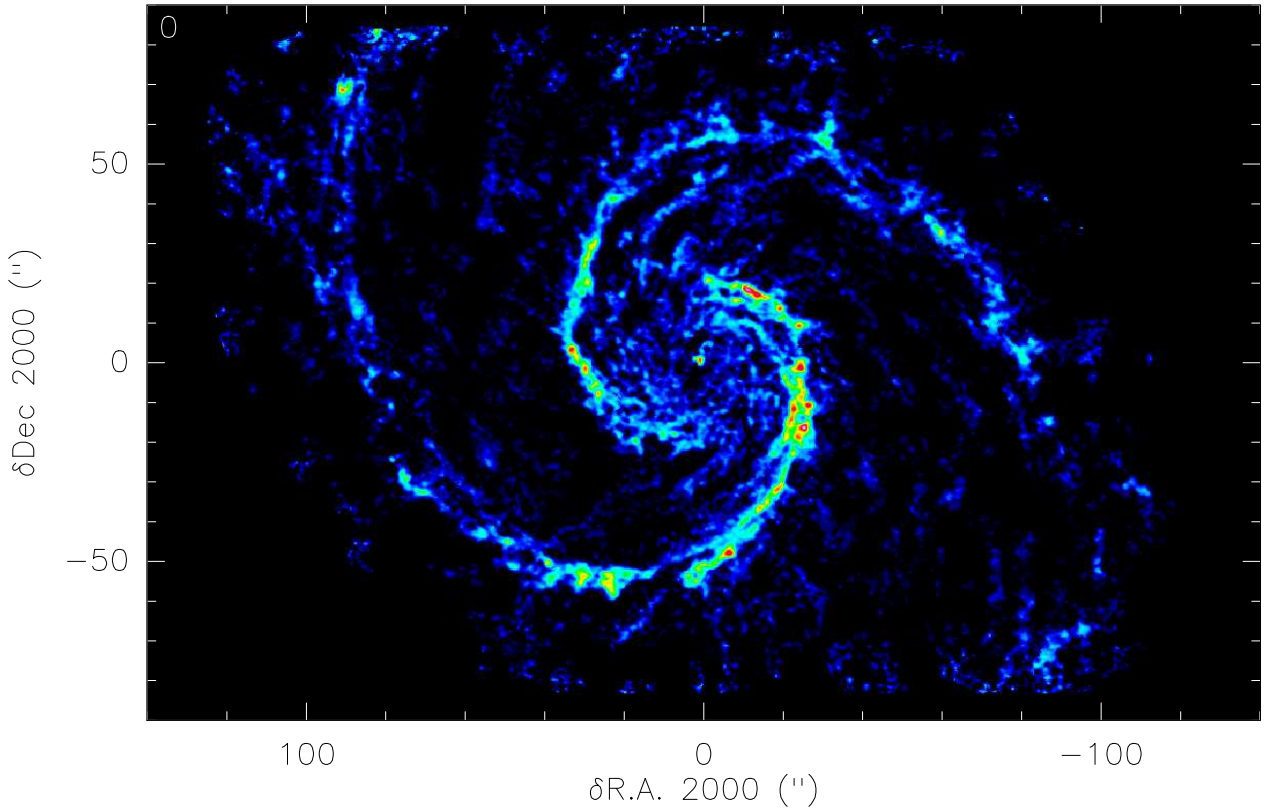
Example Extracting a given subset of an image.



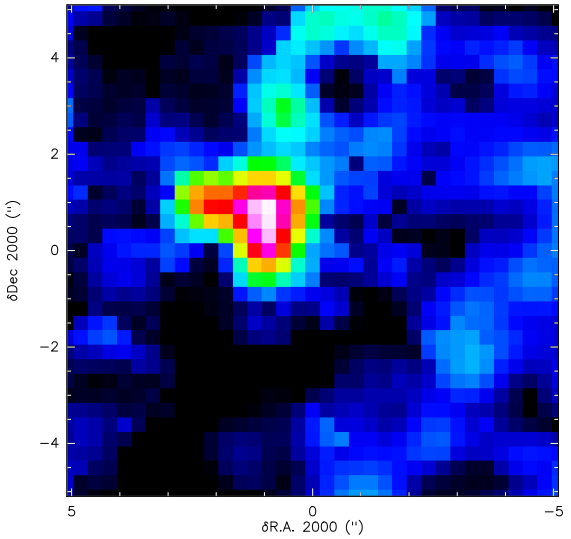

```
SIC> run extract
Waiting ...
I-RUN,  Task extract running, logfile is
I-RUN,  /home/pety/.gag/logs/extract.gildas

I-GDF_STBL,  Setting  2 starting blocks
I-GIO_RIH,  GDFBIG re-allocation      2
 Old dimensions            2                    601
S-EXTRACT,  Successful completion


I-RUN,  Elapsed .0, User .0, System .0
I-RUN,  Task extract completed successfully
SIC>
```

GILDAS Tasks: II. Results

Full image before extraction



Zoom image after extraction



GILDAS Tasks: III. What is available, where?

Displaying the list of tasks with a one-line description per task.

```
SIC> help task
...
correlation-analysis

CORRELATE         Correlation image of two input images or data cubes
HISTO_CLOUD       Cross histogram of two input images (Table output)
HISTO_CROSS       Cross histogram of two input images/cubes (Image output)
HISTO_DOUBLE      Histogram of an image as a function of another one
HISTO_SIMPLE      Histogram of an image or of a table
HISTO_TABLE       Cross histogram of column tables
MINIMIZE          Find best linear correlation between two images
REGRESSION        Compute the linear regression from a cross histogram
CORRELATE         Correlation image of two input images or data cubes
TABLE_DENSITY     Density of points from 2 columns of a table
...
```

Tasks are grouped to simplify their listing. A single group can be listed.

```
SIC> help task mapping ! Try "help task ?" for the available groups
mapping

UV_MOSAIC         Gather or Split Mosaic UV Tables
CLEAN             CLEAN deconvolution of an input data cube
MX                MX imaging and deconvolution algorithm starting from a UV table
PRIMARY           Correct for the interferometer primary beam attenuation an input cube
MAKE_PRIMARY      Linear mosaicing tool from individual cleaned fields
MAKE_MOSAIC       Build a dirty mosaic from individual fields (suited only to MAPPING)
MAKE_MOSAIC       Build a dirty mosaic from individual fields (suited only to MAPPING)
UV_AVERAGE        Channel averaging of UV data
UV_CAL            Apply an input table of gain (from UV_GAIN) to a table of uncalibrated visibilities
...
```

Tasks can be launched from any GILDAS program They are grouped only for bookkeeping purposes and to give the user a rough idea of their actions.

GILDAS Tasks: IV. Init file

```
SIC> $ls ! GILDAS remembers your last input for this task in extract.init
extract.init  fits  pro  tmp tools
SIC> type extract.init
TASK\FILE "Input file name" Y_NAME$ "tmp/m51-12co10.lm"
TASK\FILE "Output file name" X_NAME$ "tmp/m51-12co10-ext.lm"
TASK\INTEGER "Bottom left corner" BLC$[4]  451 284 1 1
TASK\INTEGER "Top right corner" TRC$[4]  485 318 1 1
TASK\INTEGER "Position of one pixel in input image" PIXEL_IN$[4]  451 284 1 1
TASK\INTEGER "Position of this pixel in output image" PIXEL_OUT$[4]  1 1 1 1
TASK\LOGICAL "Initialize output image" INITIALIZE$ YES
SAY "If YES, Fill information Below"
TASK\INTEGER "Dimensions of output image" X_DIM$[4]  0 0 0 0
TASK\LOGICAL "Initialize with input file's blanking value" BLANKOUT$ YES
TASK\REAL "Initialization value [1 real]" INITVAL$ 0
TASK\GO
SIC> run extract ! Next time you will run this task, the widget will already be filled for you
Waiting ...
```

GILDAS Tasks: V. Debugging

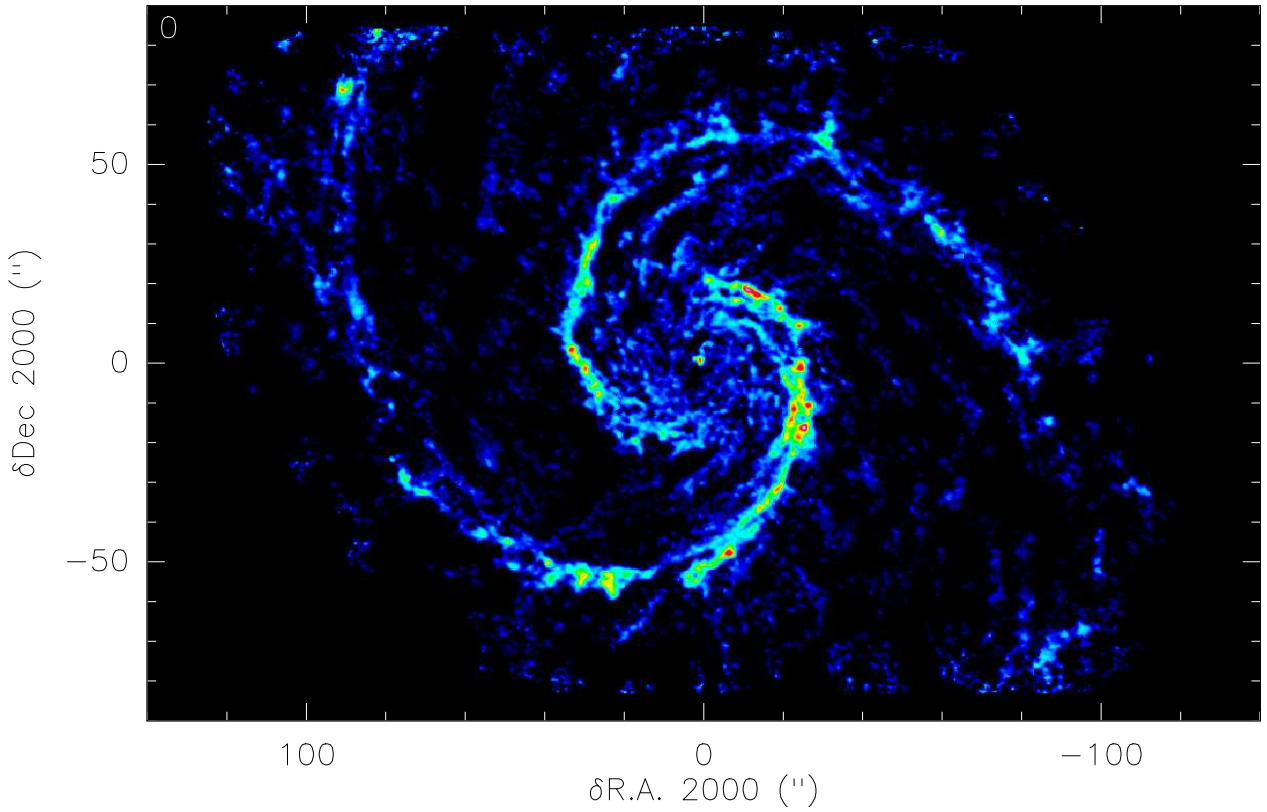
```
SIC> spy extract
::::::::::::::
/home/pety/.gag/scratch/124729/extract.par
::::::::::::::
Y_NAME$
tmp/m51-12co10.lm
X_NAME$
tmp/m51-12co10-ext.lm
BLC$
                  451                   284                     1                     1
TRC$
                  485                   318                     1                     1
PIXEL_IN$
                  451                   284                     1                     1
PIXEL_OUT$
                    1                     1                     1                     1
INITIALIZE$
 T
X_DIM$
                    0                     0                     0                     0
BLANKOUT$
 T
::::::::::::::
/home/pety/.gag/logs/extract.gildas
::::::::::::::
I-GDF_STBL,  Setting  2 starting blocks
I-GIO_RIH,  GDFBIG re-allocation      2
 Old dimensions            2                   601
S-EXTRACT,  Successful completion
```

A more user-friendly interface to the EXTRACT task

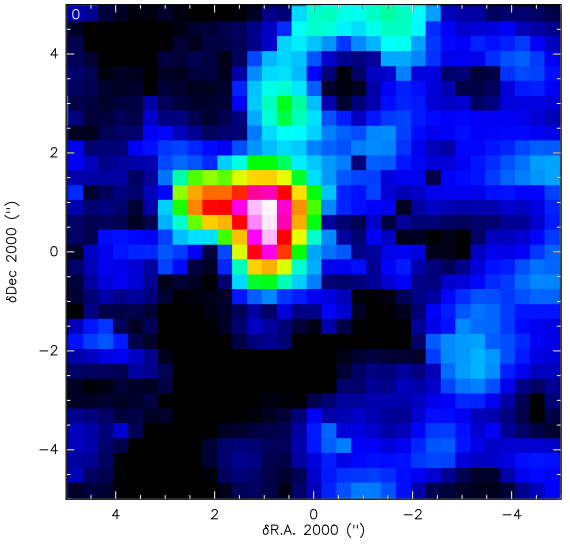
A more user-friendly interface to the EXTRACT task:
I. Results

```
SIC> @ extract-tools
SIC> let name tmp/m51-12co10
SIC> let type lm
SIC> let center 0 ! arcsecond
SIC> let size 10 ! arcsecond
SIC> @ do-extract
```

Full image before extraction



Zoom image after extraction



A more user-friendly interface to the EXTRACT task:
II. Defining global variables of many different kinds

```
begin procedure extract-def
  if .not.exist(name)   define character name*256 /global
  if .not.exist(type)   define character type*16 /global
  if .not.exist(size)   define double size[2] /global
  if .not.exist(center) define double center[2] /global
  if .not.exist(extract) then
     define structure extract /global
     define integer extract%blc[4] extract%trc[4] extract%axis[4] /global
     define double extract%min[4] extract%max[4] /global
  endif
end procedure extract-def
SIC> @ extract-tools
SIC> exa name
NAME                 ! Character*  256 GLOBAL
SIC> exa type
TYPE            =                            ! Character*16 GLOBAL
SIC> exa size
SIZE            is a double precision Array     of dimensions  2
   0.00000000000000000       0.00000000000000000
SIC> exa center
CENTER          is a double precision Array     of dimensions  2
   0.00000000000000000       0.00000000000000000
SIC> exa extract%
  EXTRACT%       ! Structure GLOBAL
  EXTRACT%MAX     is a double precision Array     of dimensions  4
  EXTRACT%MIN     is a double precision Array     of dimensions  4
  EXTRACT%AXIS    is a long integer Array     of dimensions  4
  EXTRACT%OUT     is a long integer Array     of dimensions  4
  EXTRACT%TRC     is a long integer Array     of dimensions  4
  EXTRACT%BLC     is a long integer Array     of dimensions  4
```

A more user-friendly interface to the EXTRACT task:
III. Parsing the input variables ⇒ Lot's of bookkeeping

```
if exist(extract%head) delete /var extract%head
define double rad_per_sec
let rad_per_sec pi/(180*3600)
define header extract%head 'name'"."'type' read /global
! Handle the X axis
if (extract%head%x_axis.ne.0) then
   ! Found the x_axis => assume that a conversion from second to radian is needed
   let extract%axis[1] extract%head%x_axis
   let extract%min[1] (center[1]-0.5*size[1])*rad_per_sec
   let extract%max[1] (center[1]+0.5*size[1])*rad_per_sec
else
   ! Assume that center[1] and size[1] refers to first axis in cube units
   let extract%axis[1] 1
   let extract%min[1] (center[1]-0.5*size[1])
   let extract%max[1] (center[1]+0.5*size[1])
endif
! Handle the Y axis
if (extract%head%y_axis.ne.0) then
   ! Found the y_axis => assume that a conversion from second to radian is needed
   let extract%axis[2] extract%head%y_axis
   let extract%min[2] (center[2]-0.5*size[2])*rad_per_sec
   let extract%max[2] (center[2]+0.5*size[2])*rad_per_sec
else
   ! Assume that center[2] and size[2] refers to second axis in cube units
   let extract%axis[2] 2
   let extract%min[2] (center[2]-0.5*size[2])
   let extract%max[2] (center[2]+0.5*size[2])
endif
```

A more user-friendly interface to the EXTRACT task:
IV. Playing with coordinate axes to comput BLC and TRC

```
! Compute blc and trc for the LM axes
let extract%blc 1
let extract%trc 1
define integer kaxis  ! Intermediate local variable to simplify code
define double min max ! Intermediate local variable to simplify code
for iaxis 1 to 2 ! 1 = x and 2 = y
   let kaxis extract%axis[iaxis]
   if (extract%head%convert[3,kaxis].ne.0) then
      ! Compute: ix = xref+(x-xval)/xinc
      let min extract%head%convert[1,kaxis]+(extract%min[iaxis]-extract%head%convert[2,kaxis])
      let min min/extract%head%convert[3,kaxis]
      let max extract%head%convert[1,kaxis]+(extract%max[iaxis]-extract%head%convert[2,kaxis])
      let max max/extract%head%convert[3,kaxis]
   else
      message e extract "Axis "'kaxis'" has a zero valued increment"
      return base ! Stop command flow after explaining why to the user
   endif
   let extract%blc[kaxis] min(min,max) ! Depending on inc sign min and max could be inverted
   let extract%trc[kaxis] max(min,max) ! Depending on inc sign min and max could be inverted
next iaxis
delete /var extract%head ! Done with the header => Delete to avoid memory leaks
```

A more user-friendly interface to the EXTRACT task:
V. User feedback on what will be done

```
! The following code will produce something like
!    I-EXTRACT,  Will extract [451:485,284:318,1:1,1:1]
define character mess*80
let mess "Will extract ["
for iaxis 1 to 4
   let mess 'mess''extract%blc[iaxis]'":"'extract%trc[iaxis]'
   if (iaxis.lt.4) let mess 'mess'","
next iaxis
let mess 'mess'"]"
message i extract 'mess'
```

A more user-friendly interface to the EXTRACT task:
VI. Silently calling the task from inside a procedure

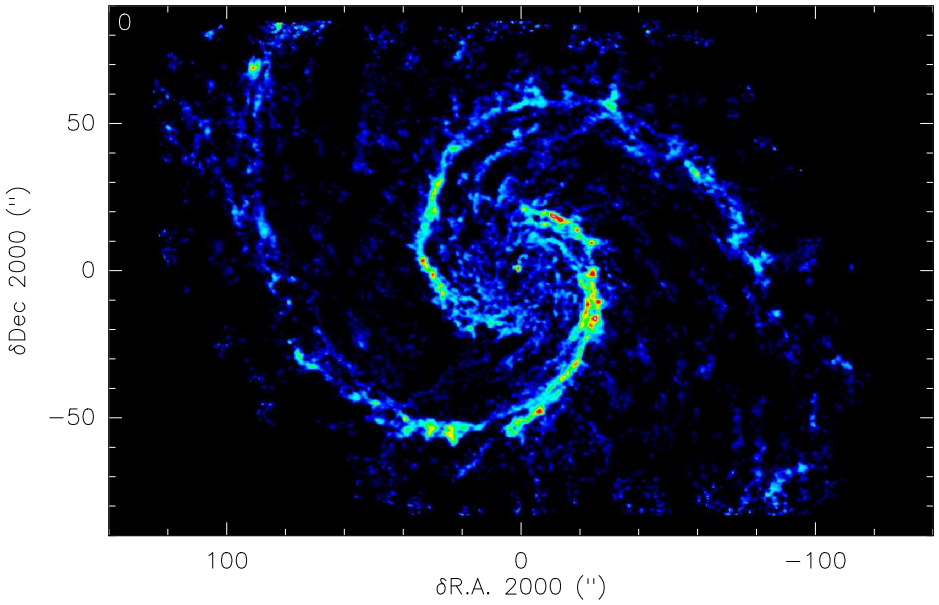
```
begin data gag_scratch:extract.init
  ! Parameter file for EXTRACT task
  task\file "Input file name"  Y_NAME$ 'name'"."'type'
  task\file "Output file name" X_NAME$ 'name'"-ext."'type'
  task\integer "Bottom left corner" BLC$[4] 'extract%blc[1]' 'extract%blc[2]' 'extract%blc[3]' 'extract%bl
  task\integer "Top right corner"   TRC$[4] 'extract%trc[1]' 'extract%trc[2]' 'extract%trc[3]' 'extract%tr
  task\integer "Position of one pixel in input image"   PIXEL_IN$[4]  'extract%blc[1]' 'extract%blc[2]' '
  task\integer "Position of this pixel in output image" PIXEL_OUT$[4] 1 1 1 1
  task\logical "Initialize output image" INITIALIZE$ YES
  SAY "If YES, Fill information Below"
  task\integer "Dimensions of output image" X_DIM$[4]  0 0 0 0
  task\logical "Initialize with input file's blanking value" BLANKOUT$ YES
  task\real "Initialization value [1 real]" INITVAL$ 0
  task\go
end data gag_scratch:extract.init
!
begin procedure extract-do
  @ extract-parse-input ! Code from previous slides
  run extract gag_scratch:extract.init /nowindow
  let name 'name'"-ext"
  say " "
  message i extract "Sub-cube "'name'"."'type'" successfully extracted"
  message w extract "Name set to "'name'
  say " "
end procedure extract-do
```

A more user-friendly interface to the EXTRACT task: VII. How to plunge the initial image into a square grid?

⇒ Easy!

```
SIC> @ extract-tools
SIC> let name tmp/m51-12co10
SIC> let type lm
SIC> let center 0 ! arcsecond
SIC> let size 300 ! arcsecond
SIC> @ do-extract
```

Full image before extraction



Zoom image after extraction