GDL (IDL) for beginners

The programming language GDL – a free variant of IDL – uses library functions that were specially designed for the analysis of astrophysical data and includes routines to read and write FITS files.

The interpreter is started in the console with:

```
gdl
```

Now enter the GDL program code line by line and the interpreter executes it right away.

However, it's more conveniant to write the program as a file. Open a text editor of your choice (kate, emacs, nedit, vi,...) and write/edit the program file (its file extension should be .pro), then save it and compile it in GDL (in contrast to other languages like Fortran or C this will not create an executable file, because IDL/GDL works differently) using the command

```
.compile myprog.pro
```

Depending on the name given to the program (inside the file as first line: *pro*), e.g.

```
pro prog1
```

the compiling will make the program “prog1” available for use. Execute it by calling

```
prog1
```

in GDL.

Syntax

Attention, in contrast to the regular Linux console capitalization does not matter for commands in GDL, i.e.

```
print, "Hello World!"
PRINT, "Hello World!"
```

and even

```
PrInT, "Hello World!"
```

return the same output. However, for strings and filenames it's different, as usual.

Basics

Calling the interpreter:	gdl

Starting GDL help:	?
Show all variables:	help
Compile a script:	.compile myprogram.pro
Execute the script in gdl:	myprogram
Leave gdl:	exit
First line of a script:	pro myprogram
Last line of a script (end of the program):	end

(Note: Replace “myprogram” with useful names that describe the program.)

FITS files

Read the content of a fits file into the variable “image1”:

```
image1 = readfits('filename.fits')
```

If the image is not in the current directory, give the full (or relative) path (i.e. '../data/filename.fits').

Depending on the internal computer architecture, data consisting of more than one byte are saved in either high or low byte order. In rare cases the default byte order is different between the source system (where the fits file was created) and the reduction system. This can be seen in (seemingly) absurd values and can be noticed when viewing the image of the fits file. If that happens, convert the byte order:

```
byteorder, image1
```

Save an array (e.g. *image2*) as fits file:

```
writefits, 'newfilename.fits', image2
```

Note: All FITS commands require the (freely distributed) AstroLib for GDL/IDL (which is installed for the praktikum account).

Input/Output

If the variable *namevar* has a content (e.g. namevar='world'), create output by writing:

```
print, "Hello ", namevar
```

Any number of values/strings/variables and be attached, separted by commas. At the end there will be a line break.

Write into a file:

```
openw, handle, 'filename.dat'
printf,handle,i,variable1,"some text",variable2
```

```
close, handle
```

handle needs to be an arbitrary but unique integer value. Alternative:

```
openw,out,'filename.dat',/get_lun
printf,out,i,variable1,"some text",variable2
close,out
free_lun,out
```

GDL will assign a handle then (/get_lun). The command free_lun is needed to unassign that handle again.

Variables and arrays

Indexed variables start with the index 0. The array with indices 0, 1,..., MMAX has MMAX+1 entries, thus.

Dimensioning for a two dimensional array for integer numbers:

```
variablename = intarr(MMAX+1,NMAX+1)
```

If the variable shall contain larger numbers than Integer allows ($i>2^15-1=32767$), use Long or even Long64 (32 bit or 64 bit) arrays:

```
variablename = lonarr(MMAX+1,NMAX+1)
variablename = lon64arr(MMAX+1,NMAX+1)
```

For floating point numbers (Float) use

```
variablename = fltarr(MMAX+1,NMAX+1)
```

For double-precision numbers (Double) use

```
variablename = dblarr(MMAX+1,NMAX+1)
```

For strings use

```
variablename = strarr(MMAX+1,NMAX+1)
```

The number of entries in an array (e.g. *variablename*) can be checked with the command

```
size(variablename,/dimensions)
```

which returns an array itself (data type, size etc.). To obtain the number, extract the first value (index 0!) of that array:

```
number = (size(variablename,/dimensions))(0)
```

Control structures

Loop syntax:

```
for j=n1,n2 do begin
  ...
endfor
```

Conditional syntax:

```
if a gt b then begin
  ...
endif
```

See the help of GDL (type “?”) for more, see e.g. the Documentation page of GDL.