GNU Data Language (GDL)
a free and open-source implementation of IDL

Sylwester Arabas
University of Warsaw, Poland

Alain Coulais
CNRS LERMA, Observatoire de Paris, France

Takeshi Enomoto
JAMSTEC, Yokohama, Japan

GDL development is led by Marc Schellens, the project founder
Plan of the talk

• About GDL
  • what’s GDL (and IDL)
  • why to use GDL, who uses it and what for
  • how to get GDL
  • how to help and contribute

• Hands-on demo
  • running the GDL interpreter
  • syntax basics
  • opening an example dataset
  • simple array manipulation
  • basic plotting functionalities
before we begin with GDL: what’s IDL

http://www.ittvis.com/

The IDL Programming Language

When you need to transform complex scientific data from numbers into visualizations to convey meaningful information – such as 2 and 3-dimensional lines, surface and contour plots, or high-quality images – you need a programming language that is intuitive and powerful at the same time, and one that doesn’t require excessive time and effort to produce expert-level results.

IDL is the programming language choice of scientists and engineers because it’s easy to learn, easy to use, and requires fewer lines of code than other programming languages, so getting from data to discovery is easier and faster.

What Makes IDL so Easy and Effective?

- Dynamic Type System
- Intuitive Rules and Conventions
- Access Virtually any Type of Data

The IDL programming language requires fewer lines of code than many other languages (bottom). Five lines of IDL code were used to create a contour plot of coastline topography (top).

- IDL is a product and registered trade mark of ITT Visual Information Solutions
- IDL is a tool for data analysis and visualisation
- IDL is a programming language
- IDL package contains: interpreter/compiler, library routines, documentation and a development environment
- IDL is related with GDL as Matlab with Octave/Scilab, M$ Office with OO.org etc
• The situation
  • **IDL is widely used in astronomy**, since 1977, including a lot of Space Missions (official pipelines)
  • **Several well known libraries** (Astron, MPFIT, Solar Soft, ...) used as commodity by Scientists
  • Code is rather well **independant of OS’s** (VMS, many Unix, OSX, Linux and also MS-win) and can live during very long time
  • **Fast coding**, nice for testing idea/modles/processing during data analysis

• The problems?
  • Dependence.
    • IDL is not open source free software. We had serious warning in the past, some platforms (Unix, VMS) had desapear. Despite support by NASA, editor may desapear. Some choices by editor (widgets, limited graphical outputs, ...) upset users. Because of the large codebase, we cannot afford disparition!
  • No general purpose alternative.
    • Now Python in Astronomy become a real alternative, but it was not the case 3 or 5 years ago.

• Solution? Having a free clone.
The situation

- **IDL is widely used in astronomy**, since 1977, including a lot of Space Missions (official pipelines)
- **Several well known libraries** (Astron, MPFIT, Solar Soft, ...) used as commodity by Scientists
- Code is rather well **independant of OS’s** (VMS, many Unix, OSX, Linux and also MS-win) and can live during very long time
- **Fast coding**, nice for testing idea/modles/processing during data analysis

The problems?

- **Dependence**. IDL is not open source free software. We had serious warning in the past, some platforms (Unix, VMS) had desapear. Despite support by NASA, editor may desapear. Some choices by editor (widgets, limited graphical outputs, ...) upset users. Because of the large codebase, we cannot afford disparition!
- **No general purpose alternative**. Now Python in Astronomy become a real alternative, but it was not he case 3 or 5 years ago.
The situation

- **IDL is widely used in astronomy**, since 1977, including a lot of Space Missions (official pipelines)
- **Several well known libraries** (Astron, MPFIT, Solar Soft, ...) used as commodity by Scientists
- Code is rather well **independant of OS's** (VMS, many Unix, OSX, Linux and also MS-win) and can live during very long time
- **Fast coding**, nice for testing idea/modles/processing during data analysis

The problems?

- **Dependence**. IDL is not open source free software. We had serious warning in the past, some platforms (Unix, VMS) had disappear. Despite support by NASA, editor may disappear. Some choices by editor (widgets, limited graphical outputs, ...) upset users. Because of the large codebase, we cannot afford disparition!
- **No general purpose alternative**. Now Python in Astronomy become a real alternative, but it was not he case 3 or 5 years ago.

Solution? **Having a free clone.**
About GDL: the project and its authors

- aim: develop an open-source drop-in replacement for IDL
About GDL: the project and its authors

- aim: develop an open-source drop-in replacement for IDL
- project founder and leader: Marc Schellens
About GDL: the project and its authors

- **aim:** develop an open-source drop-in replacement for IDL
- **project founder and leader:** Marc Schellens
- **project team:**
  - volunteers from around the world including:
    - Joel Gales, Pierre Chanial, Peter Messmer, . . .
  - Alain’s students including:
  - users sending bug-reports, patches, comments, . . .
  - packagers including Takeshi Enomoto from JAMSTEC

Current status:
- compiler/interpreter: generally complete
- library routines: partially implemented (ca. 400 routines)
- documentation: virtually nothing (IDL docs are on the web)
- development environment: no plans

About GDL: the project and its authors

• aim: develop an open-source drop-in replacement for IDL
• project founder and leader: Marc Schellens
• project team:
  • volunteers from around the world including:
    Joel Gales, Pierre Chanial, Peter Messmer, . . .
  • Alain’s students including:
  • users sending bug-reports, patches, comments, . . .
  • packagers including Takeshi Enomoto from JAMSTEC
• current status:
  • compiler/interpreter: generally complete
  • library routines: partially implemented (ca. 400 routines)
  • documentation: virtually nothing (IDL docs are on the web)
  • development environment: no plans

Sylwester Arabas, Alain Coulais & Takeshi Enomoto
About GDL: the project and its authors

• aim: develop an open-source drop-in replacement for IDL
• project founder and leader: Marc Schellens
• project team:
  • volunteers from around the world including:
    Joel Gales, Pierre Chanial, Peter Messmer, ...
  • Alain’s students including:
  • users sending bug-reports, patches, comments, ...
  • packagers including Takeshi Enomoto from JAMSTEC
• current status:
  • compiler/interpreter: generally complete
  • library routines: partially implemented (ca. 400 routines)
  • documentation: virtually nothing (IDL docs are on the web)
  • development environment: no plans
• more info: http://gnudatalanguage.sourceforge.net/
About GDL: some pros and cons

- **inherited from IDL:**
  - one can process data and get customized plots of them with a few (dozens) lines of code
  - opens many data formats: text, binary, netCDF, HDF, FITS, GRIB...

- **GDL specific:**
  - it's free! and you can modify and expand it to suit your needs
  - numerics based on reliable open-source libraries as (GSL, FFTW, ...)
  - enhanced CLI thanks to readline, SVG & GRIB support, Python bridge
  - compiles on many platforms (Linux, BSDs, OSX, Solaris, Cygwin, ...)

- **inherited from IDL:**
  - bit archaic syntax with constructs from numerous languages from Fortran to C++ and inconsistent naming of library routines...
  - but who cares, it's meant to be more a bash than a Java of scientific computing – it's intended for short scripts and interactive usage

- **GDL specific:**
  - not all library routines are implemented yet
  - some useful keywords are missing
  - lack of documentation (but IDL docs are on the web)
About GDL: some **pros and cons**

- **inherited from IDL:**
  - one can process data and get customized plots of them with a few (dozens) lines of code
  - opens many data formats: text, binary, netCDF, HDF, FITS, GRIB...

- **GDL specific:**
  - it’s free! and you can modify and expand it to suit your needs
  - numerics based on reliable open-source libraries as (GSL, FFTW, ...)
  - enhanced CLI thanks to readline, SVG & GRIB support, Python bridge
  - compiles on many platforms (Linux, BSDs, OSX, Solaris, Cygwin, ...)

- bit archaic syntax with constructs from numerous languages from Fortran to C++ and inconsistent naming of library routines . . .
- but who cares, it’s meant to be more a bash than a Java of scientific computing – it’s intended for short scripts and interactive usage

- not all library routines are implemented yet
- some useful keywords are missing
- lack of documentation (but IDL docs are on the web)
About GDL: some **pros** and **cons**

- **inherited from IDL:**
  - one can process data and get customized plots of them with a few (dozens) lines of code
  - opens many data formats: text, binary, netCDF, HDF, FITS, GRIB...

- **GDL specific:**
  - it’s free! and you can modify and expand it to suit your needs
  - numerics based on reliable open-source libraries as (GSL, FFTW, ...)
  - enhanced CLI thanks to readline, SVG & GRIB support, Python bridge
  - compiles on many platforms (Linux, BSDs, OSX, Solaris, Cygwin, ...)

- **inherited from IDL:**
  - bit archaic syntax with constructs from numerous languages from Fortran to C++ and inconsistent naming of library routines ...
  - but who cares, it’s meant to be more a bash than a Java of scientific computing – it’s intended for short scripts and interactive usage
About GDL: some **pros** and **cons**

- **inherited from IDL:**
  - one can process data and get customized plots of them with a few (dozens) lines of code
  - opens many data formats: text, binary, netCDF, HDF, FITS, GRIB...

- **GDL specific:**
  - it’s free! and you can modify and expand it to suit your needs
  - numerics based on reliable open-source libraries as (GSL, FFTW, ...)
  - enhanced CLI thanks to readline, SVG & GRIB support, Python bridge
  - compiles on many platforms (Linux, BSDs, OSX, Solaris, Cygwin, ...)

- **inherited from IDL:**
  - bit archaic syntax with constructs from numerous languages from Fortran to C++ and inconsistent naming of library routines ... but who cares, it’s meant to be more a bash than a Java of scientific computing – it’s intended for short scripts and interactive usage

- **GDL specific:**
  - not all library routines are implemented yet
  - some useful keywords are missing
  - lack of documentation (but IDL docs are on the web)
About GDL: known users

- people at the Paris Observatory, Lyon Observatory, Strasbourg Observatory
- people at NASA (GSFC, JPL, ... ) and MIT
- people in Max Planck’s labs
- people at the Lockheed Martin Solar and Astrophysics Laboratory and Stanford University (2 conf. papers on GDL)
- contacts in Australia, Brazil, India, Israel, Japan, Spain, Switzerland, UK, USA, Venezuela... (astro- and geophysicists)
- some guys in the medical imaging community (the GDLffDICOM package)
- Sylwester’s students at the University of Warsaw
- students at the Dept of Physics, Montana State Univ.
- and many more... (assuming from the number of downloads, packages, forum posts)
About GDL: known users

- people at the Paris Observatory, Lyon Observatory, Strasbourg Observatory
- people at NASA (GSFC, JPL, ...) and MIT
- contacts in Australia, Brazil, India, Israel, Japan, Spain, Switzerland, UK, USA, Venezuela... (astro- and geophysicists)
- some guys in the medical imaging community (the GDLffDICOM package)
- Sylwester's students at the University of Warsaw
- students at the Dept of Physics, Montana State Univ.
- and many more... (assuming from the number of downloads, packages, forum posts)
About GDL: known users

- people at the Paris Observatory, Lyon Observatory, Strasbourg Observatory
- people at NASA (GSFC, JPL, ...) and MIT
- people in Max Planck's labs

...?
About GDL: known users

- people at the Paris Observatory, Lyon Observatory, Strasbourg Observatory
- people at NASA (GSFC, JPL, ...) and MIT
- people in Max Planck’s labs
- people at the Lockheed Martin Solar and Astrophysics Laboratory and Stanford University (2 conf. papers on GDL)

...?
About GDL: known users

- people at the Paris Observatory, Lyon Observatory, Strasbourg Observatory
- people at NASA (GSFC, JPL, ... ) and MIT
- people in Max Planck’s labs
- people at the Lockheed Martin Solar and Astrophysics Laboratory and Stanford University (2 conf. papers on GDL)
- contacts in Australia, Brazil, India, Israel, Japan, Spain, Switzerland, UK, USA, Venezuela... (astro- and geophysicists)
About GDL: known users

- people at the Paris Observatory, Lyon Observatory, Strasbourg Observatory
- people at NASA (GSFC, JPL, . . . ) and MIT
- people in Max Planck’s labs
- people at the Lockheed Martin Solar and Astrophysics Laboratory and Stanford University (2 conf. papers on GDL)
- contacts in Australia, Brazil, India, Israel, Japan, Spain, Switzerland, UK, USA, Venezuela... (astro- and geophysicists)
- some guys in the medical imaging comunity (the GDLffDICOM package)
About GDL: known users

- people at the Paris Observatory, Lyon Observatory, Strasbourg Observatory
- people at NASA (GSFC, JPL, ...) and MIT
- people in Max Planck’s labs
- people at the Lockheed Martin Solar and Astrophysics Laboratory and Stanford University (2 conf. papers on GDL)
- contacts in Australia, Brazil, India, Israel, Japan, Spain, Switzerland, UK, USA, Venezuela... (astro- and geophysicists)
- some guys in the medical imaging comunity (the GDLffDICOM package)
- Sylwester’s students at the University of Warsaw

...?
About GDL: known users

- people at the Paris Observatory, Lyon Observatory, Strasbourg Observatory
- people at NASA (GSFC, JPL, ...) and MIT
- people in Max Planck’s labs
- people at the Lockheed Martin Solar and Astrophysics Laboratory and Stanford University (2 conf. papers on GDL)
- contacts in Australia, Brazil, India, Israel, Japan, Spain, Switzerland, UK, USA, Venezuela... (astro- and geophysicists)
- some guys in the medical imaging comunity (the GDLffDICOM package)
- Sylwester’s students at the University of Warsaw
- students at the Dept of Physics, Montana State Univ.

...?
About GDL: known users

- people at the Paris Observatory, Lyon Observatory, Strasbourg Observatory
- people at NASA (GSFC, JPL, ...) and MIT
- people in Max Planck’s labs
- people at the Lockheed Martin Solar and Astrophysics Laboratory and Stanford University (2 conf. papers on GDL)
- contacts in Australia, Brazil, India, Israel, Japan, Spain, Switzerland, UK, USA, Venezuela... (astro- and geophysicists)
- some guys in the medical imaging community (the GDLffDICOM package)
- Sylwester’s students at the University of Warsaw
- students at the Dept of Physics, Montana State Univ.
- and many more... (assuming from the number of downloads, packages, forum posts)
About GDL: how we use it

• @ the Observatoire de Paris:
  • analysing time series (several satellites)
  • modelization and inversion of time responses of various detectors: IR detectors, bolometers
  • 2D deconvolution methods for Radio Interferometry (Clean, MEM, and alternative)
  • teaching various topics, at different levels: FFT, image processing ...
About GDL: how we use it

• @ the Observatoire de Paris:
  • analysing time series (several satellites)
  • modelization and inversion of time responses of various detectors: IR detectors, bolometers
  • 2D deconvolution methods for Radio Interferometry (Clean, MEM, and alternative)
  • teaching various topics, at different levels: FFT, image processing ...

• @ the University of Warsaw:
  • aircraft & LIDAR measurements data analysis and visualisation
  • teaching meteorological data processing:
    • geosci. model output data: interpolation, subsetting, statistics
    • signals processing: FFT, wavelets, principal components, filtering
    • satellite image and weather-forecast display/processing
    • plotting, data formats: CVS, netCDF, HDF4, HDF5, GRIB
    • ”operational” weather forecast visualisation (GRIB+SVG)
About GDL: how others use it

- **At Strasbourg and Lyon:**
  doing heavy computations without blocking IDL licences
  (GDL cited in an MNRAS article!)

- **at LESIA, Paris Obs.:**
  substituting IDL by GDL in an automatic pipeline from a
given format (PDS used in Planetary mission) to a new one
(Virtual Observatory, XML ...). Gaining perenity on very long
time scale (space missions: 10 to 30 years).

- **at Lockheed Martin/Stanford:**
  deconvolution of images from solar telescopes using
  a CUDA-enhanced GDL (poster at NVIDIA conference!)

- **at Montana State University:**
  web-based remote access to data analysis environment for
  students of an IDL course
About GDL: how others use it

- **At Strasbourg and Lyon:**
  doing heavy computations without blocking IDL licences
  (GDL cited in an MNRAS article!)

- **at LESIA, Paris Obs.:**
  substituting IDL by GDL in an automatic pipeline from a
  given format (PDS used in Planetary mission) to a new one
  (Virtual Observatory, XML ...). Gaining perenity on very long
  time scale (space missions: 10 to 30 years).
About GDL: how others use it

- **At Strasbourg and Lyon:**
  doing heavy computations without blocking IDL licences
  (GDL cited in an MNRAS article!)

- **at LESIA, Paris Obs.:**
  substituting IDL by GDL in an automatic pipeline from a
given format (PDS used in Planetary mission) to a new one
(Virtual Observatory, XML ...). Gaining perenity on very long
time scale (space missions: 10 to 30 years).

- **at Lockheed Martin/Stanford:**
  deconvolution of images from solar telescopes using
a CUDA-enhanced GDL (poster at NVIDIA conference!)
About GDL: how others use it

- **At Strasbourg and Lyon:**
  doing heavy computations without blocking IDL licences
  (GDL cited in an MNRAS article!)

- **at LESIA, Paris Obs.:**
  substituting IDL by GDL in an automatic pipeline from a
  given format (PDS used in Planetary mission) to a new one
  (Virtual Observatory, XML ...). Gaining perenity on very long
time scale (space missions: 10 to 30 years).

- **at Lockheed Martin/Stanford:**
  deconvolution of images from solar telescopes using
  a CUDA-enhanced GDL (poster at NVIDIA conference!)

- **at Montana State University:**
  web-based remote access to data analysis environment for
  students of an IDL course
GDL - GNU Data Language, a free IDL (Interactive Data Language, see http://ittvis.com/idl/) compatible incremental compiler.

**Download Now!**
gdl-0.9rc4.tar.gz (1.5 MB) OR View all files

**Project Reviews**

- **That's a very good work.**
  posted by astroste 313 days ago

- **An incomplete but functional and very useful Interactive Data Language interpreter.**
  posted by anonymous 359 days ago

- **Extremely useful!**
  posted by anonymous 288 days ago

- **High funcionality!**
  posted by anonymous 134 days ago

- **GDL is a great chance to introduce IDL to students. Thanks to all developers!**
  posted by anonymous 197 days ago

- **Bit of a pain to compile, but generally works well and makes it quick and easy to do sophisticated data analysis.**
  posted by anonymous 244 days ago
GDL> zad10_2D
% Compiled module: ZAD10_2D.
% Compiled module: MEAN.
% Compiled module: LOADCT.
% LOADCT: Loading table RED TEMPERATURE
GDL>
```
1 pro periodogram, czas, sqnl, _extra = ex
2 n = n_elements(czas)
3 dt = czas[1]-czas[0]
4 f = findgen(n/2+1) / n / dt
5 amp2 = abs((fft(sqnl))[0:n/2])^2
6 amp2[1:n/2-1] *= 2 ; uwzgl. częstotliwości ujemnych
7 moc_dB = 10*log10(amp2/max(amp2))
8 ; norm. do max więc bez norm. do sumy wag w oknie
9 oplot, f, moc_dB, p_sym=10, _extra = ex
10 end

12 pro hann_demo
13 n = 512 ; dt = 1./(n-1) * 2 * 1PI
14 nyquist = .5 / dt
15
16 czas = findgen(n) * dt
17 sqnl = sin(nyquist/11 * 2*1PI * czas) +
       cos(nyquist/3 * 2*1PI * czas) / 4
18 hann = hann(n)
19
20 !P.MULTI = [0,1,2] & !X.STYLE = 1
21
22 oplot, linestyle=4, czas, sqnl,
23 xtitle="czas [s]", ytitle="amplituda",
24 title="y=sin(Nyq/11 * 2pi x) + cos(Nyq/3 * 2pi x)/4"
25 oplot, linestyle=0, czas, sqnl*hann
26 oplot, czas, hann
27
28 oplot, [0, nyquist], [0, -150], /nodata,
29 xtitle="f [Hz]", /xlog,
30 ytitle="widmo mocy (periodogram) [dB]"
31 periodogram, linestyle=4, czas, sqnl
32 periodogram, linestyle=0, czas, sqnl * hann
33 end
```
Sylwester Arabas, Alain Coulais & Takeshi Enomoto
GNU Data Language 17/ 34

```gdl
\[ \text{cmd} = \text{"wget --continue --output-document=": + \text{plik} + \text{" http://nomads.ncep.noaa.gov/cgi-bin/filter_gfs_hd.pl?file=gfs.t06z.mastergrb2f" + hh_str + \text{"&lev_500_mb=on&var_HGT=on&var_UGRD=on&var_VGRD=on&var_ABSV=on&subregion=&dir=%2Fgfs." + data + \text{"&master" + \text{"&bottomlat=" + strtrim(string(maplimit[0]), 2) + \"&leftlon=" + strtrim(string(maplimit[1]), 2) + \"&" + \text{\"&toplat=" + strtrim(string(maplimit[2]), 2) + \"" + \text{\"&" + \"+ spawn, cmd, output, exit_status=status
}
if status ne 0 then begin
  message, /continue, \"pobranie pliku nie powiodło się\"
  continue
endif

; pobranie danych z plików GRIB (w pierwszym krzywym) grib_f = gribapi_open_file(plik)
n_msgs = gribapi_count_in_file(grib_f)
for m = 0, n_msgs - 1 do begin
  grib_m = gribapi_new_from_file(grib_f)
  if h eq h_prwsz and m eq 0 then begin
    gribapi_get_data, grib_m, lats, lons, tmp
    gribapi_get, grib_m, \"numberOfPointsAlongAP\"
    gribapi_get, grib_m, \"numberOfPointsAlongAW\"
    n_stps = 1 + (h_oostn - h_prwsz) / h_perstp
    lons = (temporary(lons))[indgen(n_lons)]
    lats = (temporary(lats))[indgen(n_lats) * n]
    vals = fltarr(n_lons, n_lats, n_stps, n_msg)
  endif else gribapi_get, grib_m, \"values\", tmp
  vals[*], *, h/h_perstp, m = temporary(tmp)
  gribapi_release, grib_m
endfor ; m
```

WPDM2009 Zad. B.1 : Przegnozy GFS 2010012706+00
- No startup file read (GDL_STARTUP/IDL_STARTUP env. var. not set).
- Please report bugs, feature or help requests and patches at: http://sourceforge.net/projects/gnudatalanguage/

GDL> set plot, 'svg'
GDL> v = fittarr(1024) & v[9] = 1
GDL> plot, wt(v, 4, /inverse)
GDL> device, /close
GDL> help, /version, /stru
** Structure !VERSION, 8 bags, data length=28:
  ARCH STRING 'i86pc'
  OS STRING 'SunOS'
  OS FAMILY STRING 'unix'
  OS NAME STRING 'SunOS'
  RELEASE STRING '6.0'
  BUILD DATE STRING 'Jan 27 2010'
  MEMORY_BITS INT 32
  FILE_OFFSET_BITS INT 32

GDL>
GDL works under Cygwin! :)
Sylwester Arabas, Alain Coulais & Takeshi Enomoto

GNU Data Language
About GDL: how to get GDL

- **pre-compiled packages** *(no trouble, fast, no options, releases only):*
  - Fedora: 32/64 bit, up-to-date, even experimental features available
  - Debian: ca. 15 architectures!, bit outdated (2009-09)
  - ArchLinux: up-to-date, small feature set
  - Ubuntu: obsolete (2008-04)
  - unofficial up-to-date packages for Ubuntu & Debian by Lea Noreskal
  - HMUG (OSX): up-to-date
  - hpc.sourceforge.net: obsolete (2008-04)
About GDL: how to get GDL

- pre-compiled packages (no trouble, fast, no options, releases only):
  - Fedora: 32/64 bit, up-to-date, even experimental features available
  - Debian: ca. 15 architectures!, bit outdated (2009-09)
  - ArchLinux: up-to-date, small feature set
  - Ubuntu: obsolete (2008-04)
  - unofficial up-to-date packages for Ubuntu & Debian by Lea Noreskal
  - HMUG (OSX): up-to-date
  - hpc.sourceforge.net: obsolete (2008-04)

- pre-configured packages (no trouble, some options, releases only):
  - Gentoo: up-to-date, rich feature set
  - Macports: up-to-date, rich feature set (the only one with CMSVLIB!)
  - Fink: up-to-date
  - FreeBSD: bit outdated (2009-09)
About GDL: how to get GDL

- pre-compiled packages (no trouble, fast, no options, releases only):
  - Fedora: 32/64 bit, up-to-date, even experimental features available
  - Debian: ca. 15 architectures!, bit outdated (2009-09)
  - ArchLinux: up-to-date, small feature set
  - Ubuntu: obsolete (2008-04)
  - unofficial up-to-date packages for Ubuntu & Debian by Lea Noreskal
  - HMUG (OSX): up-to-date
  - hpc.sourceforge.net: obsolete (2008-04)

- pre-configured packages (no trouble, some options, releases only):
  - Gentoo: up-to-date, rich feature set
  - Macports: up-to-date, rich feature set (the only one with CMSVLIB!)
  - Fink: up-to-date
  - FreeBSD: bit outdated (2009-09)

- source distribution (all options up to you, troubles happen, releases only)
About GDL: how to get GDL

- pre-compiled packages (**no trouble, fast, no options, releases only**):
  - Fedora: 32/64 bit, up-to-date, even experimental features available
  - Debian: ca. 15 architectures!, bit outdated (2009-09)
  - ArchLinux: up-to-date, small feature set
  - Ubuntu: obsolete (2008-04)
  - unofficial up-to-date packages for Ubuntu & Debian by Lea Noreskal
  - HMUG (OSX): up-to-date
  - hpc.sourceforge.net: obsolete (2008-04)

- pre-configured packages (**no trouble, some options, releases only**):
  - Gentoo: up-to-date, rich feature set
  - Macports: up-to-date, rich feature set (the only one with CMSVLIB!)
  - Fink: up-to-date
  - FreeBSD: bit outdated (2009-09)

- source distribution (**all options up to you, troubles happen, releases only**)

- CVS (**all options up to you, newest features/fixes, expect troubles**)

Sylwester Arabas, Alain Coulais & Takeshi Enomoto
About GDL: the Macports GDL package

eyrie:~ slayoo$ port info gnudatalanguage
gnudatalanguage @0.9rc4, Revision 4 (math, science)
Variants: mpich, openmp, python, universal, wxWidgets
Description: A free IDL (Interactive Data Language) compatible incremental compiler (ie. runs IDL programs).
Homepage: http://gnudatalanguage.sourceforge.net/
Build Dependencies: autoconf, automake, libtool
Library Dependencies: zlib, gsl, ncurses, readline, plplot, netcdf, hdf4, hdf5-18, grib_api, libproj4, ImageMagick, xorg-libX11, udunits2, fftw-3, fftw-3-single, cmsvlib
Platforms: darwin
License: GPLv2
Maintainers: takeshi@macports.org, slayoo@igf.fuw.edu.pl

http://www.macports.org/
About GDL: how to help and contribute

- Try it! Let us know what you use GDL for. Or if you don’t, why not.
- Report feature/improvement requests and bugs (with example codes).
- Send patches (GPL code) with new features, bugfixes or test routines.
- Port or package GDL for new platforms.
- Report problems to packagers.
- Let others know about GDL!

We need help!
About GDL: how to help and contribute

- Try it! Let us know what you use GDL for. Or if you don’t, why not.
- Report feature/improvement requests and bugs (with example codes).
- Send patches (GPL code) with new features, bugfixes or test routines.
- Port or package GDL for new platforms.
- Report problems to packagers.
- Let others know about GDL!

We need help!

- What is really important now is that GDL is User-wish driven project.
- We cannot check all the cases: please report!
- We don’t know all keywords: please report if missing or misbehaving.
- We don’t know or haven’t checked all (sometime crazy) interactions between keywords: please report problems (IDL Documentation is itself not always without ambiguities).
Hands-on: let’s start

$ gdl
GDL> print, 'Hello world!'
Hello world!
GDL> $ echo "Hello world!"
Hello world!
GDL> hello
% Compiled module: HELLO.
Hello world!
GDL> hello
Hello world!
GDL> exit
$ gdl -quiet -e hello
% Compiled module: HELLO.
Hello world!

hello.pro:

1 pro hello
2 print, 'Hello world!'
3 end
Hands-on: syntax basics

- assignment & function call with one positional argument:
  GDL> a = findgen(11)

- procedure call & variable size inquiry
  GDL> help, a

- keyword argument & some Fortran heritage...
  GDL> print, a, format='(4G)'

- ... and some relief for C users
  GDL> print, a, format='('%''%4.3g'')'

- simplest plot (x-axis: array indices) & math ops. applied to arrays
  GDL> plot, sin(a*!PI/10)^2

- array addressing
  GDL> print, a[0]
  GDL> print, a[4:5]
  GDL> print, a[[9,10]]
  GDL> print, a[a]
  GDL> print, a[indgen(3)]
Hands-on: CReSS-SDM example

```
1  pro cressanim, ncfile
2    nc = ncdf_open(ncfile)                                       ; opening the netCDF file
3  ncdf_varget, nc, 'X', x  nx = n_elements(x)                    ; \ | grid-box->x/z ; time-step->time
4  ncdf_varget, nc, 'Z', z  nz = n_elements(z)
5  ncdf_varget, nc, 'T', t  nt = n_elements(t)
6
7  loadct, 1
8  qclev = findgen(11) / 1000.                                    ; [0, .001, ... .01] kg/kg
9
10  for i = 0, nt - 1 do begin
11     ncdf_varget, nc, 'qc', qc, offset=[0,0,0,i], count=[nx,1,nz,1]
12     contour, reform(qc), x, z, /fill, levels=qclev, title=t[i] ; filled-contour plots of cloud water
13
14     ncdf_varget, nc, 'qr', qr, o=[0,0,0,i], c=[nx,1,nz,1]        ; any unambiguous keyword name is OK
15     wh = where(qr gt 0, cnt)
16     if cnt gt 0 then begin
17        ai = array_indices(qr, wh)
18        plots, x[ai[0,*]], z[ai[2,*]], psym=7                    ; rain water plotted with "X" symbols
19     endif
20
21    wait, 1                                                          ; wating one seconds before next frame
22 endfor
23 end
```
NCDF_OPEN() opens a netCDF file

returns: a ”handle” to the file (long integer)
arguments: filename
keywords: /nowrite, /write

NCDF_VARGET retrieves data from an open netCDF file

arguments: handle, variable id or name, output variable
keywords: count=array, offset=array, stride=array
LOADCT  loads a color palette

  argument:  palette id (0...40)

CONTOUR  plots data with contours

  arguments:  values array, [x array, y array]
  keywords:  /fill, levels=array, title=string, ...

PLOTS  plots data with symbols or lines

  arguments:  x array, y array
  keywords:  pSYM=integer, ...
WHERE() returns the (1d) indices of array elems that evaluate to true

arguments: array, output var with count of "true" elems
keywords: ...

ARRAY_INDICES() converts 1-d indices to n-d indices

returns: array with n-d indices for given 1-d indices
arguments: n-d array, 1-d indices
keywords: ...

REFORM() changes the dimensions of an array

returns: the modified array (copy or not)
arguments: array, [new dimensions]
keywords: /overwrite
.COMPILE triggers recompilation of a file/routine
(interpreter directive)

STOP stops execution allowing interactive debugging
(to be inserted in the code)

节约 continues execution after a STOP
(interpreter directive)
ご清聴ありがとうございました。