A short tutorial on
GrADS – Grid Analysis and Display System
# Table of Contents

1. THE SOFTWARE ................................................................. 4
   1.1 What is GrADS? ............................................................ 4
   1.2 Downloading GrADS ....................................................... 4
   1.3 Support and Discussion List ............................................. 4

2. BACKGROUND AND BASIC COMMANDS ........................................ 4
   2.1 Installing GrADS ........................................................... 4
   2.2 The data and descriptor (.ctl) files ................................... 5
   2.3 Running GrADS (initiation session) ..................................... 6
   2.4 The “set” command ........................................................ 10
   2.5 Manipulating Dimensions ................................................. 10
   2.6 Other Basic Command .................................................... 11
   2.7 Examples and Basic Exercises .......................................... 12

3. PLOTTING GRAPHICS ................................................................. 14
   3.1 Graphics types ............................................................ 14
   3.2 Projections maps ........................................................... 16
   3.3 Inserting Titles, Texts, Forms and Symbols .............................. 17
   3.4 Controlling Graphical Options .......................................... 18
   3.5 Page Control ............................................................... 23
   3.6 Application examples and exercises ..................................... 24

4. GENERATING GRAPHICS OUTPUT FILES ......................................... 26
   4.1 GrADS metafile (.gmf) archives ........................................ 26
   4.2 GrADS Metafile Viewer for Windows ................................. 26
   4.3 gxtran application ........................................................ 26
   4.4 Applications gxps and gxeps ........................................... 27
   4.5 printim and wi commands ............................................... 27
   4.6 Application examples and exercises ..................................... 28

5. VARIABLES, EXPRESSIONS AND FUNCTIONS .................................... 30
   5.1 Names of Variables ....................................................... 30
   5.2 Defining New Variables: define command ................................ 31
   5.3 Expressions ............................................................... 31
   5.4 Functions ................................................................. 31
   5.5 Application examples and exercises ..................................... 36

6. SCRIPTING LANGUAGE (script.gs) ................................................. 37
   6.1 Basic Concepts ........................................................... 37
   6.2 Language Elements in Scripts .......................................... 40
   6.3 Application examples and exercises ..................................... 45

7. ADDITIONAL TOPICS ............................................................... 46
7.1 The Template Option ........................................................................................................46
7.2 Generating binary files with fwrite .................................................................................46
7.3 Creating a Mask ...............................................................................................................46
Have fun using GrADS in your weather research and operation tasks!!!

1. THE SOFTWARE

1.1 What is GrADS?
GrADS - Grid Analysis and Display System- is interactive software used in the tasks of accessing, manipulation and visualization of geophysical data. GrADS works with data sets in binary, GRIB, NetCDF or HDF-SDS formats, in which the variables can have up to 5 dimensions (longitude, latitude, vertical levels, time, and ensemble) as specified by a descriptor (control) file. Currently, GrADS is one of the most widely used software by the operational and meteorological research communities around the world. This software was originally developed by researcher Brian Doty at COLA (http://cola.gmu.edu/cola.php) within in the late 1980s. Its distribution is totally free through its official website: http://cola.gmu.edu/grads/grads.php. Data matrices may contain one or more variables arranged in a regular grid, either nonlinear, or Gaussian, or at station or variable resolution points. Variables can be plotted and combined using various types of graphics, which can be recorded in PostScript format or various graphic image formats (PNG, GIF, JPEG, etc). GrADS has a scripting language with which it is possible to develop sophisticated analyzes, derived variable calculations and automatic visualization applications (graphical interfaces with buttons and dropmenus clickable). Within the scripts it is possible to develop interactivity with functions, expressions or external routines written with other programming languages (FORTRAN, C++, UNIX Shell, etc.) and also with operating system command lines (MS-DOS, Windows, LINUX, UNIX). Current versions bring a wide variety of intrinsic functions (GrADS own functions), but the user can also add their own function using external routines developed in FORTRAN or another language. GrADS can be run in batch mode and therefore scripts can be used to perform automatic tasks without the need for direct user presence.

1.2 Downloading GrADS
On the official GrADS download page (http://cola.gmu.edu/grads/downloads.php) you will find precompiled executable files (binary files), source code and supplementary data sets and utilities (Map files, source files, etc.) required for GrADS installation and execution. Documentation Online documentation and all manuals are available at http://cola.gmu.edu/grads/gadoc/gadoc.php.

1.3 Support and Discussion List
There is a list of effectively active GrADS users, where you can share information, learn about recent refinements and developments, new versions, as well as mostly help in troubleshooting GrADS users in general. To be on the GrADS list, send an email to the address gradsusr-request@list.cineca.it and provide your address, institution, etc. To see the online file from the GrADS list go to the address http://dao.gsfc.nasa.gov/grads_listserv/

2 BACKGROUND AND BASIC COMMANDS

2.1 Installing GrADS

The GrADS executables are typically placed in /usr/local/bin/grads/. If you do not have write permission for this directory, you can put them in a subdirectory of your home
directory (e.g. ~/bin) or anywhere else in your path. The font and map files are supplementary data sets that are required in order to run GrADS. Their default location is: /usr/local/lib/grads/. If you do not have write permission for this directory you can place the files elsewhere, but you must also change the environment variable GADDIR so the GrADS executables will know where to find these files. You can download the data files separately by clicking here: data2.tar.gz.

cd <dirname>
tar xvfz data2.tar.gz
setenv GADDIR <dirname>

An additional supplementary tar file contains a sample gridded data set along with an example session that reviews basic GrADS capabilities. This data set is not required to run GrADS. If you have not used GrADS before, you are strongly encouraged to obtain this file and go through the sample session. You can download it directly by clicking here: example.tar.gz.

2.2 The data and descriptor (.ctl) files

Basically, GrADS works with two main files:
- the data file (for example, data.dat, data.grib, data.bin …)
- and the descriptor file (for example, descriptor.ctl)

The data file must be in the BINARY, GRIB, NetCDF, or HDF-SDS formats. The descriptor.ctl is a text-type file, in which all specifications of the dimension of data file are described. A simple example descriptor file is below:

Significations of the lines of the descriptor file (model.ctl):

| DSET ^model.dat | Specifies the name of the data file (^means the data are in the current directory) |
| OPTIONS little_endian | This entry controls various aspects of the way GrADS interprets the raw data file and can take The keyword uses here describe the byte ordering of the data file |
| UNDEF -2.56E33 | Missing values (Ignored in the plot) |
### Title
5 Days of Sample Model Output

#### XDEF
72 LINEAR 0.0 5.0

- **Zonal (longitude) grid specifications:**
  - number of grid boxes, increment type, minimum, resolution

#### YDEF
46 LINEAR -90.0 4.0

- **Meridional (latitude) grid specifications:**
  - number of grid boxes, increment type, minimum, resolution

#### ZDEF
7 LEVELS 1000 850 700 500 300 200 100

- **Vertical grid specifications:**
  - number of levels, increment type, pressure levels

#### TDEF
5 LINEAR 02JAN1987 1DY

- **Time grid:**
  - number of time periods, increment type, minimum, resolution

#### VARS 8

- **Number of Variables in the file**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name</th>
<th>Number</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ps</td>
<td>Surface Pressure</td>
<td>0</td>
<td>99</td>
<td></td>
</tr>
<tr>
<td>u</td>
<td>U Winds</td>
<td>7</td>
<td>99</td>
<td></td>
</tr>
<tr>
<td>v</td>
<td>V Winds</td>
<td>7</td>
<td>99</td>
<td></td>
</tr>
<tr>
<td>hgt</td>
<td>Geopotential Heights</td>
<td>7</td>
<td>99</td>
<td></td>
</tr>
<tr>
<td>tair</td>
<td>Air Temperature</td>
<td>7</td>
<td>99</td>
<td></td>
</tr>
<tr>
<td>q</td>
<td>Specific Humidity</td>
<td>5</td>
<td>99</td>
<td></td>
</tr>
<tr>
<td>tsfc</td>
<td>Surface Temperature</td>
<td>0</td>
<td>99</td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>Precipitation</td>
<td>0</td>
<td>99</td>
<td></td>
</tr>
</tbody>
</table>

#### ENDVARS
End of variable listing

**Note:**
The full description of the descriptor file components for the various data formats is in Appendix A1 of this manual. You can find the online documentation on descriptor file at the address [http://cola.gmu.edu/grads/gadoc/descriptorfile.html](http://cola.gmu.edu/grads/gadoc/descriptorfile.html)

### 2.3 Running GrADS (initiation session)

This section will give you a guidance on how to: run GrADS, set the graphics windows, open data file, know the content of the file, display a variable, and exit grads.

- **In the terminal type** `grads` **and press enter**

GrADS will prompt you with a landscape vs. portrait question (as illustrate):

Just press `enter`.

At this point, referring to two figures below, a graphics output window (on the left) should open on your console (on the right). You may wish to move or resize this window. **Keep in mind that you will be entering GrADS commands from the window () where you first started GrADS** -- this window will need to be made the 'active' window and you will not want to entirely cover that window with the graphics output window.
In the text window (console, where you started grads from), you should now see a prompt:

```
ga->
```

You will enter GrADS commands at this prompt and see the results displayed in the graphics output window.

✓ Set the graphic window

**Tip:** The GrADS preview screen always opens with the black background, which sometimes makes it difficult to interpret certain graphics. To change the background of the preview screen to white, in the console where you have grads prompt (ga->) type following command:

```
ga-> set display color white
```

```
ga-> clear
```

What happened?

✓ Open a data file

Within the GrADS prompt, the command to open the descriptor file (which in turn controls the data file) is done as follows:

```
ga-> open model.ctl
```

Informations that appears at the opening of the .ctl file.

```
ga-> open model.ctl
Scanning description file:  model.ctl
Data file model.dat is open as file 1
LAT set to -90 90
LEV set to 1000 1000
```

✓ You may want to see what is in this file, so enter:
This data contains surface pressure, represented by a variable name, ps, display this variable by entering:

By default, GrADS will display a lat/lon plot at the first time and at the lowest level in the data set.

Now you may want to produce a hard copy of the plot. So enter the command:

```
printim myfirstplot.png
```

Now you may want to take a look at your GrADS output file. To do so you may need to leave the GrADS session. Enter the command `quit`.

Now, you have left the GrADS session, and went back to the Linux environment. You are expected to use Linux commands (not GrADS commands), while in the Linux environment!

- List the content of the current directory (`GrADSTutorial`) and look for a file with `.png` extension, and you should be able to see the file you have created while you were in GrADS environment.
- Which Linux command did you use to open this file?
Note for this initiation section:
Other opening commands are listed in the following table:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>grads -l</code></td>
<td>Opens GrADS in landscape mode</td>
</tr>
<tr>
<td><code>grads -p</code></td>
<td>Opens GrADS in portrait mode</td>
</tr>
<tr>
<td><code>grads -b</code></td>
<td>Runs GrADS in batch mode (No window opens)</td>
</tr>
<tr>
<td><code>grads -c &quot;GrADS command line &quot;</code></td>
<td>Open GrADS and run the quoted command</td>
</tr>
</tbody>
</table>

These options can be used in combinations, such as:

<table>
<thead>
<tr>
<th>Combination</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>grads -lc &quot;open model.ctl&quot;</code></td>
<td>Opens GrADS in landscape mode and run the quoted command (open the file model.ctl)</td>
</tr>
<tr>
<td><code>grads -bpc &quot;run scripts.gs&quot;</code></td>
<td>Opens GrADS in portrait mode, in batch mode (No graphical window opens) run the command in the <code>grads</code> script file script.gs</td>
</tr>
</tbody>
</table>

Hand on tools: See lab2, a sample of GrADS Session (it takes about 30 minutes to complete this session).
2.4 The “set” command
The set command specifies "when", "where" and "how" variables will be plotted. For example:

<table>
<thead>
<tr>
<th>When</th>
<th>where</th>
<th>how</th>
</tr>
</thead>
<tbody>
<tr>
<td>ga-&gt; set t 1</td>
<td>ga-&gt; set lat -20 -10</td>
<td>ga-&gt; set gxout line</td>
</tr>
</tbody>
</table>

2.5 Manipulating Dimensions
The dimensions are manipulated using the set command, according to the examples below:

- `ga-> set lat valofLAT1 valofLAT2`: Specifies the grid between latitudes valofLAT1 and valofLAT2; If valofLAT2 is not specified, we have the latitude fixed at the point of the valofLAT1.
- `ga-> set y valofY1 valofY2`: Same as above.
- `ga-> set lon valofLON1 valofLON2`: Specifies the grid between the lengths valofLON1 and valofLON2; If valofLON2 is not specified, we have the longitude fixed at the point of valofLON1.
- `ga-> set x valofX1 valofX2`: Same as above.
- `ga-> set lev valofLev1 valofLev2`: Specifies the grid between the vertical levels valofLev1 and valofLev2; If valofLev2 is not specified, we have the vertical level fixed in valofLev1.
- `ga-> set z valofZ1 valofZ2`: Same as above.
- `ga-> set t valofT1 valofT2`: Specifies the grid between the times valofT1 and valofT2; If valofT2 is not specified, we have the fixed time in valofT1.
- `ga-> set time valofT1 valofT2`: Same as above, but the syntax of valofT1 and valofT2 must be in the form: 00z09feb2004.
Comments:

• The LAT values of the Southern Hemisphere and LON of the Western Hemisphere are preceded by the negative sign.
• GrADS consider the Y dimension ranging from south to north and the X dimension ranging from west to east. Therefore, when specifying the same, it is necessary to make the first set of LAT (LON) further south (west).

For example:

\texttt{ga-> set lat -30 -5}
\texttt{ga-> set lon -80 -20}

2.6 Other Basic Command

The query or q command is used to obtain information about data files (names of variables, etc.), dimensions, screen and geographical positions, statistics in general, etc. For example:

\texttt{ ga-> q file } \quad \text{Specifies general information for the descriptor file}

<table>
<thead>
<tr>
<th>File 1 : 5 Days of Sample Model Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descriptor: model.ctl</td>
</tr>
<tr>
<td>Binary: model.dat</td>
</tr>
<tr>
<td>Type = Gridded</td>
</tr>
<tr>
<td>Xsize = 72  Ysize = 46  Zsize = 7</td>
</tr>
<tr>
<td>Tsize = 5  Esize = 1</td>
</tr>
<tr>
<td>Number of Variables = 8</td>
</tr>
<tr>
<td>ps 0 99 Surface Pressure</td>
</tr>
<tr>
<td>u 7 99 U Winds</td>
</tr>
<tr>
<td>v 7 99 V Winds</td>
</tr>
<tr>
<td>hgt 7 99 Geopotential Heights</td>
</tr>
<tr>
<td>tair 7 99 Air Temperature</td>
</tr>
<tr>
<td>q 5 99 Specific Humidity</td>
</tr>
<tr>
<td>tsfc 0 99 Surface Temperature</td>
</tr>
<tr>
<td>p 0 99 Precipitation</td>
</tr>
</tbody>
</table>

\textbf{Note:} If multiple descriptor files are open, use the following:

\texttt{ga-> q files} \quad \text{Specifies general informations for all the descriptors files opened}

\texttt{ga-> q file n} \quad \text{to have information about the opened descriptor file number n}

\texttt{ ga-> q dims } \quad \text{Specifies the current dimensions}

<table>
<thead>
<tr>
<th>Default file number is: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>X is varying Lon = 0 to 360 X = 1 to 73</td>
</tr>
<tr>
<td>Y is varying Lat = -90 to 90 Y = 1 to 46</td>
</tr>
<tr>
<td>Z is fixed Lev = 1000 Z = 1</td>
</tr>
<tr>
<td>T is fixed Time = 00202JAN1987 T = 1</td>
</tr>
<tr>
<td>E is fixed Ens = 1 E = 1</td>
</tr>
</tbody>
</table>

\textbf{Note:} If multiple descriptor files are open, use the following:

\texttt{ga-> clear} \quad \text{Clear the preview screen Same as above}

\texttt{ga-> reinit} \quad \text{Restart GrADS; Close all the opened .ctl}

\texttt{ga-> reset} \quad \text{Restart GrADS; But without closing .ctl}

\texttt{ga-> !command-line} \quad \text{Run operating system command line}

\texttt{ga-> help} \quad \text{Basic help}
2.7 Examples and Basic Exercises
The examples and basic exercises below are based on gfs_sample.grb2 and its control file gfs_sample.ctl. The assumption is that the data is available in ~/GrADSTutorial directory.

<table>
<thead>
<tr>
<th>Example 1:</th>
<th>Proposed exercise 1:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open GrADS in Portrait mode and plot the pressure variable at the mean sea level.</td>
<td><strong>Open GrADS in Landscape and plot the precipitation field</strong></td>
</tr>
<tr>
<td>At the GrADS prompt, type:</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; set display color white</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; c</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; open gfs_sample.ctl</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; q file</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; d prmslmsl</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example 2:</th>
<th>Proposed exercise 2:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plotting two overlapping variables (pressure and horizontal wind).</td>
<td><strong>Plot the precipitation field superimposed on the horizontal wind field</strong></td>
</tr>
<tr>
<td>At the GrADS prompt, type:</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; c</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; d prmslmsl</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; d ugrdprs;vgrdprs</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; d skip(ugrdprs,20); vgrdprs</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example 3:</th>
<th>Proposed exercise 3:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plot of surface temperature for African Countries.</td>
<td><strong>Plot the map of specific humidity over your country</strong></td>
</tr>
<tr>
<td>At the GrADS prompt, type:</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; c</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; set mpdset hires bmap</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; q dims</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; set lat -40 40</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; set lon -20 55</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; d tmpsfc</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example 4:</th>
<th>Proposed exercise 4:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map of geopotential at 500 hPa</td>
<td><strong>Plot the horizontal wind at 200 hPa</strong></td>
</tr>
<tr>
<td>At the GrADS prompt, type:</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; c</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; set lev 500</td>
<td></td>
</tr>
</tbody>
</table>
**Example 5:**
Vertical temperature profile on the center point in Ndjamena

At the GrADS prompt, type:
```plaintext
ga-> c
ga-> set lat 12.15
nga-> set lon 15.06
ga-> set z 1 7
ga-> set zlog on
ga-> d tmpprs
```

**Proposed exercise 5:**
Plot the vertical profile of specific humidity on the center point in Dakar.

**Example 6:**
Zonal vertical profile of temperature along the equator (longitude vs altitude section)

At the GrADS prompt, type:
```plaintext
ga-> reset
ga-> set lat 0
ga-> set z 1 7
ga-> set zlog on
ga-> d tmpprs
```

**Proposed exercise 6:**
Plot the vertical meridional section (altitude vs latitude) of geopotential height along the longitude of Accra

The following two examples are performed based on the `rain_arc_month.ctl` files (ARC monthly precipitation from 1983 to 2016).

**Example 7:**
Temporal animation of the rain in Africa from January to June 1992

At the GrADS prompt, type:
```plaintext
ga-> reinit
ga-> open rain_arc_month.ctl
nga-> set lat -40 40
nga-> set lon -20 55
nga-> set time jan1992 jun1992
nga-> d rain
```

**Proposed exercise 7:**
Make the animation of the observed rain over Africa between the months of July to December of 1988

**Example 8:**
Hovmöller diagram of the rainfall observed during the year 1992 along the globe and on the equator line.

**Proposed exercise 8:**
Make the hovmöller diagrams of rain observed in 1998 along the longitudes of Africa specifically over the latitude of:
- Sahel band
At the GrADS prompt, type:

```
 ga-> c
 ga-> set time 00Z01jan1992 00Z31dec1992
 ga-> set lat 0
 ga-> d rain
```

### 3 PLOTTING GRAPHICS

#### 3.1 Graphics types

There are several graphics options. By default, if the user does not specify graphics output type, the output will be line type (for 1-dimensional data) and contour type (for 2-dimensional graphs).

The command line to select the graphics output type is:

```
 ga-> set gxout graphic_type
```

The following examples summarize different graphics output options:

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>contours (Isolines)</td>
<td>ga-&gt; open gfs_sample.ctl ga-&gt; set display color white ga-&gt; c ga-&gt; set t 3 ga-&gt; set mpdset hires ga-&gt; set lat -40 40 ga-&gt; set lon -20 55 ga-&gt; set gxout contour ga-&gt; d tmpprs-273</td>
</tr>
<tr>
<td>10</td>
<td>shaded contours (colors bands)</td>
<td>ga-&gt; c ga-&gt; set gxout shaded ga-&gt; d tmpprs-273</td>
</tr>
<tr>
<td>11</td>
<td>same as Example 10, but here shading a made on grid points.</td>
<td>ga-&gt; c ga-&gt; set gxout grfill ga-&gt; d tmpprs-273</td>
</tr>
<tr>
<td>12</td>
<td>Values in the grid points</td>
<td>ga-&gt; c ga-&gt; set gxout grid ga-&gt; d tmpprs-273</td>
</tr>
<tr>
<td>Example 13: Vectors (arrows)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------------</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ga-&gt; set gxout vector</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ga-&gt; d ugrdprs;vgrdprs</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example 14: Streamlines</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ga-&gt; c</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; set gxout stream</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; d ugrdprs;vgrdprs</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example 15: Wind with Barb (synoptic chart)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ga-&gt; c</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; set gxout barb</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; d ugrdprs; vgrdprs</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example 16: Shaded in the grid points of the values specified by the set fgvals value col value col ...</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ga-&gt; c</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; set gxout fgrid</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; set fgvals 20 4 23 8 26 2</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; d tmpprs-273</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example 17: Bar graph and error bar graph</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ga-&gt; c</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; set t 3</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; set lat 0</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; set gxout bar</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; set gxout errbar</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; d pratesfc</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example 18: Line Graph</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ga-&gt; c</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; set gxout line</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; d pratesfc</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example 19: Scatter plot</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ga-&gt; c</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; set gxout scatter</td>
<td></td>
</tr>
<tr>
<td>ga-&gt; d tmpsfc-273; tmpprs-273</td>
<td></td>
</tr>
</tbody>
</table>
### Example 20: Statistics (information) on the data (without graph)
```
ga-> c
nga-> set gxout stat
nga-> d tmpprs
```

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ga-&gt; set gxout fwrite</td>
<td>Write (generates) grads fwrite file with binary data (no graph)</td>
</tr>
<tr>
<td>ga-&gt; set gxout linefill</td>
<td>Lines with color filling between two lines</td>
</tr>
<tr>
<td>ga-&gt; set gxout value</td>
<td>Station value (station points)</td>
</tr>
<tr>
<td>ga-&gt; set gxout wxsym</td>
<td>Symbols of the Synoptic map (weather conditions)</td>
</tr>
<tr>
<td>ga-&gt; set gxout findstn</td>
<td>Find the nearest station</td>
</tr>
</tbody>
</table>

### 3.2 Projections maps
The following examples summarise different projection options:

#### Example 21: *latlon* (default) aspect ratio maintained on the screen
```
ga-> reinit
nga-> open gfs_sample.ctl
nga-> set map 1 1 10
nga-> set mproj latlon
nga-> d pratesfc (t=2)
```

#### Example 22: *scaled*, same as *latlon*, but with aspect ratio not maintained on the screen
```
ga-> reset
nga-> set mproj scaled
nga-> d pratesfc (t=2)
```

#### Example 23: Polar stereographic: *sps* (HS) or *nps* (HN)
```
ga-> c
nga-> set mproj sps
nga-> set lon –100 0
nga-> set lat –90 0
nga-> d pratesfc (t=2)
```

#### Example 24: *robinson*
```
ga-> reset
nga-> set mproj robinson
nga-> set lon –180 180
nga-> set lat –90 90
```
Example 25: Orthographic (**orthogr**)  
**ga-** > reset  
**ga-** > set mproj **orthogr**  
**ga-** > d pratesfc (t=2)

Example 26: **mollweide**  
**ga-** > reset  
**ga-** > set mproj **mollweide**  
**ga-** > d pratesfc (t=2)

Example 27: **lambert** – Conical Lambert Conformal  
**ga-** > reset  
**ga-** > set mproj **lambert**  
**ga-** > set lat -90 0  
**ga-** > d pratesfc (t=2)

Example 28: **off** same as **scaled**, but does not plot map and plot labels without lat and lon sign  
**ga-** > reset  
**ga-** > set mproj **off**  
**ga-** > d pratesfc (t=2)

### 3.3 Inserting Titles, Texts, Forms and Symbols

The command lines for entering titles, texts, shapes and symbols are as follows:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ga-</strong> &gt; draw title <strong>graphic-title</strong></td>
<td>Write title at the top of the picture</td>
</tr>
<tr>
<td><strong>ga-</strong> &gt; draw xlab <strong>X-Title</strong></td>
<td>Write title on x-axis</td>
</tr>
<tr>
<td><strong>ga-</strong> &gt; draw ylab <strong>Y-Title</strong></td>
<td>Write title on y-axis</td>
</tr>
<tr>
<td><strong>ga-</strong> &gt; draw string <strong>x</strong> <strong>y</strong> <strong>text</strong></td>
<td>Write text at the point (<strong>x</strong>, <strong>y</strong>)</td>
</tr>
<tr>
<td><strong>ga-</strong> &gt; draw line <strong>x1</strong> <strong>y1</strong> <strong>x2</strong> <strong>y2</strong></td>
<td>Draw a line between (<strong>x1</strong>,<strong>y1</strong>) and (<strong>x2</strong>,<strong>y2</strong>)</td>
</tr>
<tr>
<td><strong>ga-</strong> &gt; draw rec <strong>xlo</strong> <strong>ylo</strong> <strong>xhi</strong> <strong>yhi</strong></td>
<td>Draw a rectangle</td>
</tr>
<tr>
<td><strong>ga-</strong> &gt; draw recf <strong>xlo</strong> <strong>ylo</strong> <strong>xhi</strong> <strong>yhi</strong></td>
<td>Draw a solid (fill) rectangle</td>
</tr>
<tr>
<td><strong>ga-</strong> &gt; draw polyf <strong>x1</strong> <strong>y1</strong> <strong>x2</strong> <strong>y2</strong> ... <strong>xn</strong> <strong>yn</strong></td>
<td>Draws a polygon between (<strong>x1</strong>,<strong>y1</strong>), (<strong>x2</strong>,<strong>y2</strong>) ... (<strong>xn</strong>,<strong>yn</strong>)</td>
</tr>
<tr>
<td><strong>ga-</strong> &gt; draw mark <strong>marktype</strong> <strong>x</strong> <strong>y</strong> <strong>size</strong></td>
<td>Draw a mark on point (<strong>x</strong>,<strong>y</strong>)</td>
</tr>
<tr>
<td><strong>ga-</strong> &gt; draw wxsym <strong>symbol</strong> <strong>x</strong> <strong>y</strong> <strong>size</strong> <strong>color</strong> <strong>thickness</strong></td>
<td>Draw a weather symbol on point (<strong>x</strong>,<strong>y</strong>)</td>
</tr>
</tbody>
</table>
3.4 Controlling Graphical Options

➢ Color coding:

<table>
<thead>
<tr>
<th>Color</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>White</td>
<td>0</td>
</tr>
<tr>
<td>Black</td>
<td>1</td>
</tr>
<tr>
<td>Red</td>
<td>2</td>
</tr>
<tr>
<td>Green</td>
<td>3</td>
</tr>
<tr>
<td>Cyan</td>
<td>5</td>
</tr>
<tr>
<td>Magenta</td>
<td>6</td>
</tr>
<tr>
<td>Yellow</td>
<td>7</td>
</tr>
<tr>
<td>Orange</td>
<td>8</td>
</tr>
<tr>
<td>Purple</td>
<td>9</td>
</tr>
<tr>
<td>Yellow/Green</td>
<td>10</td>
</tr>
<tr>
<td>Medium Blue</td>
<td>11</td>
</tr>
<tr>
<td>Dark Yellow</td>
<td>12</td>
</tr>
<tr>
<td>Aqua</td>
<td>13</td>
</tr>
<tr>
<td>Dark Purple</td>
<td>14</td>
</tr>
<tr>
<td>Grey</td>
<td>15</td>
</tr>
</tbody>
</table>

Note: For the rainbow Colors Sequence (9 14 4 11 5 13 3 10 7 12 8 2 6), you can use the following commands:

```
 ga-> set ccolor rainbow
 ga-> set ccolor revrain
```
*here you reverse the colors of the rainbow*

➢ Line style coding

Usage: `ga-> set line color style thickness`

```
    0 = none            1  = solid
    2 = long dash       3  = Short dash
    4 = Long short dash 5  = dots
    6 = dot dash        7  = dot dot dash
```

➢ Mark style coding

Usage: `ga-> set cmark marktype`

```
  1  = plus sign     5  = closed square     10 = open circle with vertical bar
  6  = multiplication sign  11 = closed circle with vertical bar
  7  = open diamond    12 = closed diamond (GrADS version 2.1+)
```

0 - none  1 - plus sign  2 - open circle (default) 3 - closed circle 4 - open square
Weather Symbol code (from 1 to 41, as shown below):

Usage: \texttt{ga-> draw wxsym symbol x y size color thickness}

Command to get the screen coordinates of the point (x,y)

\texttt{ga-> q pos} \quad \text{ (Click the screen on the desired point)}

or

\texttt{ga-> q ll2xy lon lat} \quad \text{ (No need to click the screen)}

Command to control text (string):

\texttt{ga-> set string color justification thickness rotation}

This command sets attributes for strings drawn with the \texttt{draw string} command.

<table>
<thead>
<tr>
<th>Justification coding:</th>
</tr>
</thead>
<tbody>
<tr>
<td>tl = top left</td>
</tr>
<tr>
<td>tc = top center</td>
</tr>
<tr>
<td>tr = top right</td>
</tr>
<tr>
<td>l = left</td>
</tr>
<tr>
<td>c = center</td>
</tr>
<tr>
<td>r = right</td>
</tr>
<tr>
<td>bl = bottom left</td>
</tr>
<tr>
<td>bc = bottom center</td>
</tr>
<tr>
<td>br = bottom right</td>
</tr>
</tbody>
</table>

\texttt{ga-> set strsiz hsiz vsiz}

This command sets the string character size, where \texttt{hsiz} is the width of the characters; \texttt{vsiz} is the height of the characters, in virtual page inches. If \texttt{vsiz} is not specified, it will be set the the same value as \texttt{hsiz}.

\texttt{ga-> set font number}

This command allows the user to select the font for subsequent text operations. With font type \texttt{(number)} from 0 to 5.
 Commands to control the plots in the various types of graphs

- **Graphs 1-D (gxout = line):**
  - `ga> set ccolor color#` Set the color of the line
  - `ga> set cthick thickness` Set the thickness of lines (thickness from 1 to 10)
  - `ga> set cstyle linestyle` Set the line style
  - `ga> set cmark markertype` Set the style of the marker
  - `ga> set vrange v1 v2` Set the range of the values for the scale on the Y-axis
  - `ga> set missconn on|off (default off)` Connects or not lines in missing data

- **Graphic type (gxout = bar or errbar):**
  - `ga> set bargap value` Set the gap between bars (value from 0 to 100)
  - `ga> set barbase value|bottom|top` Plots bars above or below the value
  - `ga> set baropts filled|outline` Filled the bars or not
  - `ga> set cthick values` Set the thickness of line (values from 1 to 10)

- **Graphic type (gxout = linefill):**
  - `ga> set lfcols col1 col2` Fill the space between two isolines with colors col1 and col2

- **Graphic type (gxout = contour):**
  - `ga> set ccolor color#` Set the color of the isolines
  - `ga> set cthick thickness` Set the thickness of isolines (thickness from 1 to 10)
  - `ga> set cstyle linestyle` Set the isolines style
  - `ga> set cterp on|off` Turns spline smoothing on or off
  - `ga> set cint value` Sets the contour interval to the specified value
  - `ga> set cmax value` Controls the maximum value of the isolines
  - `ga> set cmin value` Controls the minimum value of the isolines
  - `ga> set black val1 val2` Omits contours between val1 and val2
  - `ga> set clevs val1 val2 ...` Plot specified values
  - `ga> set ccols col1 col2 ...` Specifies colors for clevs
  - `ga> set rbrange val1 val2` Sets the range of values used to determine which values acquire which rainbow color
  - `ga> set rbcols col1 col2 ...` Specifies a new rainbow color sequence
  - `ga> set rbcols auto` Sets colors in rainbow sequence
  - `ga> set clab on|off|forced` Controls contour labeling
  - `ga> set clskip number` Specify the number of contour lines to skip when labeling
  - `ga> set clopts color# thickness size` controls the look of the contour labels drawn on contour lines
  - `ga> set csmooth on|off` Apply smoothing. If on, the grid is interpolated to a finer grid using cubic interpolation before contouring
o Graphic type (gxout = shaded or grfill):

    ga-> set cint value
    ga-> set cmax value
    ga-> set cmin value
    ga-> set black val1 val2
    ga-> set clevs val1 val2 ...
    ga-> set ccols col1 col2 ...
    ga-> set rbrange val1 val2
    ga-> set rbcols col1 col2 ...
    ga-> set csMOOTH on|off

Sets the contour interval to the specified value
Controls the maximum value of the isolines
Controls the minimum value of the isolines
Omits contours between val1 and val2
Specifies colors for clevs
Controls the maximum value of the isolines
Controls the minimum value of the isolines
Plot specified values
Omits contours between val1 and val2
Specifies a new rainbow color sequence
Apply smoothing. If on, the grid is interpolated
to a finer grid using cubic interpolation before
contouring

o Graphic type (gxout = grid):

    ga-> set dignum number
    ga-> set digsiz size

Number of digits after the decimal place
Size (in inches, or plotter units) of numbers. 0.1 to 0.15 is usually a good range to use

o Graphic type (gxout = vector ou barb):

    ga-> set ccolor color#
    ga-> set cthick thickness
    ga-> set arrlab on|off
    ga-> set arrscl size magnitude
    ga-> set arrowhead value
    ga-> set cint value
    ga-> set cmax value
    ga-> set cmin value
    ga-> set black val1 val2
    ga-> set clevs val1 val2 ...
    ga-> set ccols col1 col2 ...
    ga-> set rbrange val1 val2
    ga-> set rbcols col1 col2 ...

Set the color of the vectors
Set the thickness of vectors (thickness from 1 to 10)
Shows or not the reference vector below the plot
Specifies arrow length scaling. Length of the vector
according to magnitude
Set the size of the arrowhead
Sets the vectors interval to the specified value
Controls the maximum magnitude of the vectors
Controls the maximum magnitude of the vectors
Omits vectors of magnitudes between val1 and val2
Plot specified values
Specifies colors for clevs
Sets the range of values used to determine which values
acquire which rainbow color
Specifies a new rainbow color sequence

o Graphic type (gxout = scatter):

    ga-> set cmark markertype
    ga-> set digsiz size
    ga-> set ccolor color#
    ga-> set vrange v1 v2
    ga-> set vrange2 v1 v2

Set the style of the marker
Size (in inches, or plotter units) of numbers. 0.1 to 0.15 is usually a good range to use
Set marker’s colors
Set the range of values for the scale on the X-axis
Set the range of values for the scale on the Y-axis

o Graphic type (gxout = fgrid):

    ga-> set fgvals val col <val col> <val col> ...

Specifies values and colors for fgrid
Graphic type (gxout = stream):

- `ga> set strmden density` Controls the appearance of the streamlines (values from -10 to 10)
- `ga> set ccolor color#` Set the color of the isolines
- `ga> set cint value` Sets the contour interval to the specified value
- `ga> set cmax value` Controls the maximum value of the isolines
- `ga> set cmin value` Controls the minimum value of the isolines
- `ga> set cthick thickness` Set the thickness of isolines (thickness from 1 to 10)
- `ga> set black val1 val2` Omits contours between val1 and val2
- `ga> set clevs val1 val2 ...` Plot specified values
- `ga> set ccols col1 col2 ...` Specifies colors for clevs
- `ga> set rbrange val1 val2` Sets the range of values used to determine which values acquire which rainbow color
- `ga> set rbcols col1 col2 ...` Specifies a new rainbow color sequence

Stations data; Graphic type (gxout = value):

- `ga> set digsiz size` Size (in inches, or plotter units) of value. 0.1 to 0.15 is usually a good range to use
- `ga> set ccolor color#` Set the color of the value
- `ga> set stid on|off2` Turns on/off display of station ID next to the data values
- `ga> set cthick thickness` Set the thickness of value (thickness from 1 to 10)

Stations data; Graphic type (gxout = barb):

- `ga> set digsiz size` Size (in inches, or plotter units) of numbers. 0.1 to 0.15 is usually a good range to use
- `ga> set ccolor color#` Set the color of barbs
- `ga> set cthick thickness` Set the thickness of barbs (thickness from 1 to 10)

Stations data; Graphic type (gxout = wxsym):

- `ga> set ccolor color#` Set the color of symbols
- `ga> set cthick thickness` Set the thickness of symbols (thickness from 1 to 10)
- `ga> set digsiz size` Size (in inches, or plotter units) of numbers. 0.1 to 0.15 is usually a good range to use
- `ga> set wxcols col1 col2 ...` Specifies the colors of symbols

Stations data; Graphic type (gxout = model):

- `ga> set ccolor color#` Set the color
- `ga> set cthick thickness` Set the thickness (thickness from 1 to 10)
- `ga> set digsiz size` Size (in inches, or plotter units) of numbers. 0.1 to 0.15 is usually a good range to use
- `ga> set wxcols col1 col2 ...` Specifies the colors of symbols
- `ga> set mdlopts noblank|blank|dig3|nodig3` Model options
Commands to control axes, maps, etc:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ga-&gt; set grid status style color# thickness</td>
<td>Specifies the characteristics of the displayed grid lines. Valid options for status are: on - both latitude and longitude lines drawn, off - grid lines are drawn, horizontal - only latitude grid lines are drawn, vertical - only longitude grid lines are drawn.</td>
</tr>
<tr>
<td>ga-&gt; set zlog on</td>
<td>off</td>
</tr>
<tr>
<td>ga-&gt; set xaxis start end &lt;increment&gt;</td>
<td>Range x-axis from start to end with increment.</td>
</tr>
<tr>
<td>ga-&gt; set yaxis start end &lt;increment&gt;</td>
<td>Range y-axis from start to end with increment.</td>
</tr>
<tr>
<td>ga-&gt; set xlevs lev1 lev2 ...</td>
<td>Specify individual labeled tick mark for the X-axis.</td>
</tr>
<tr>
<td>ga-&gt; set ylevs lev1 lev2 ...</td>
<td>Specify individual labeled tick mark for the Y-axis.</td>
</tr>
<tr>
<td>ga-&gt; set xlabs lab1</td>
<td>lab2</td>
</tr>
<tr>
<td>ga-&gt; set ylabs lab1</td>
<td>lab2</td>
</tr>
<tr>
<td>ga-&gt; set xlint interval</td>
<td>Specifies the interval between labeled tick marks on the X-axis.</td>
</tr>
<tr>
<td>ga-&gt; set ylint interval</td>
<td>Specifies the interval between labeled tick marks on the Y-axis.</td>
</tr>
<tr>
<td>ga-&gt; set xyrev on</td>
<td>off</td>
</tr>
<tr>
<td>ga-&gt; set xflip on</td>
<td>off</td>
</tr>
<tr>
<td>ga-&gt; set yflip on</td>
<td>off</td>
</tr>
<tr>
<td>ga-&gt; set xlopts color# thickness size</td>
<td>Controls the appearance of the tick labels on the X-axis.</td>
</tr>
<tr>
<td>ga-&gt; set ylopts color# thickness size</td>
<td>Controls the appearance of the tick labels on the Y-axis.</td>
</tr>
<tr>
<td>ga-&gt; set annot color# thickness</td>
<td>Controls the look of the plot annotations (draw title, the frame around the plot, any additional axes that are drawn alongside the frame, the axis labels, etc).</td>
</tr>
<tr>
<td>ga-&gt; set mdpset lowres</td>
<td>mres</td>
</tr>
<tr>
<td>ga-&gt; set map color# style thickness</td>
<td>Controls the appearance of the map lines.</td>
</tr>
<tr>
<td>ga-&gt; set mpdraw on</td>
<td>off</td>
</tr>
<tr>
<td>ga-&gt; set grads on</td>
<td>off</td>
</tr>
</tbody>
</table>

3.5 Page Control

Screen Display standard sizes are:

grads -l (landscape: 11 x 8.5)

![Diagram of landscape view]

grads -p (portrait: 8.5 x 11)

![Diagram of portrait view]
Page can be controlled using the following commands:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ga-&gt; set vpage off</code></td>
<td>Default setting, virtual page is equal to real page</td>
</tr>
<tr>
<td><code>ga-&gt; set vpage xmin xmax ymin ymax</code></td>
<td>Defines a “virtual page” that fits within the specified limits of the real page. All the graphics output will be drawn until another <code>set vpage</code> is entered.</td>
</tr>
<tr>
<td><code>ga-&gt; set parea off</code></td>
<td>Default setting, plotting area is chosen depending on the type of the graphics output.</td>
</tr>
<tr>
<td><code>ga-&gt; set parea xmin xmax ymin ymax</code></td>
<td>Specifies the area for plotting contour plots, maps, or lines graphs. This area does not include axis labels, titles, etc.</td>
</tr>
</tbody>
</table>

### 3.6 Application examples and exercises

**Example 29: Maps of Africa**

```
ga-> reinit
ga-> open gfs_sample.ctl
ga-> set display color white
ga-> c
ga-> set mpdset hires
ga-> set map 1 1 10
ga-> set grid off
ga-> set xlopts 1 1 0.15
ga-> set ylopts 1 1 0.15
ga-> set lat -40 40
ga-> set lon -20 55
ga-> set t 2
ga-> set gxout shaded
ga-> set cmin 1
ga-> set cint 5
ga-> d pratesfc*86400
ga-> set gxout contour
ga-> set cmin 1
ga-> set cint 5
ga-> set ccolor 1
ga-> set clab on
ga-> set clskip 3
ga-> d pratesfc*86400
ga-> draw title Precipitation (mm/day)
ga-> draw xlab Longitude
ga-> draw ylab Latitude```

**Proposed exercise 29:**

Over the whole grid of Africa, plot:

- Plot pressure field at sea level highlighting in shaded only the high pressures (prmslmsl> 1015),
- Plot vector wind in barb (remember to skip)
- Display the title of map,
- Write strings A and B on the center of the low and high pressure.
**Example 30:**
Two figures on the same portrait page, Rain and Outgoing Long-wave Radiation in Africa

```
ga-> set mpsdset hires
ga-> set map 1 1 10
ga-> set grid off
ga-> set grds off
ga-> set xlopts 1 1 0.15
ga-> set ylopts 1 1 0.15
ga-> set lon -20 55
ga-> set parea 0.5 8 6 10.8
ga-> set gxout shaded
ga-> set cmin 1
ga-> set cint 5
ga-> d pratesfc
ga-> set gxout contour
ga-> set cmin 1
ga-> set cint 5
ga-> set ccolor 1
ga-> d pratesfc
ga-> set parea 0.5 8 0.5 5.5
ga-> set gxout shaded
ga-> set cmax 230
ga-> set cint 10
ga-> d ulwrftoa
ga-> set gxout contour
ga-> set cmax 230
ga-> set cint 10
ga-> set ccolor 1
ga-> d ulwrftoa
```

**Proposed exercise 30:**
Plot 4 figures using the vpage option on the same landscape page.
The variables to be plotted on each of the figures are:
- Wind vector at 850 hPa
- Streamlines at 200 hPa
- Surface temperature
- Geopotential at 500 hPa

**PS:** don’t forget to put titles on each figure
4 GENERATING GRAPHICS OUTPUT FILES

4.1 GrADS metafile (.gmf) archives
* Generating a GrADS metafile file (*.gmf)

   The example below plots the temperature field and generates a .gmf file

<table>
<thead>
<tr>
<th>Example 31: Procedure to generate an .gmf file</th>
</tr>
</thead>
<tbody>
<tr>
<td>ga-&gt; enable print archive1.gmf</td>
</tr>
<tr>
<td>ga-&gt; d tmpprs</td>
</tr>
<tr>
<td>ga-&gt; print</td>
</tr>
<tr>
<td>ga-&gt; disable print</td>
</tr>
</tbody>
</table>

Notes:
✓ If the user does not disable print; the file is terminated with reinit or quit
✓ It is possible to generate several separate graphics (frames) within the same .gmf file

4.2 GrADS Metafile Viewer for Windows

GrADS metafile Viewer (GV) is an application in Windows environment that is used to make the visualization and manipulation of the generated .gmf files by GrADS.

Graphics opened within the GV can be copied and pasted into your documents (Word, PowerPoint, etc.). There are also other options, such as: printing, cutting a piece of the figure, etc.

4.3 gxtran application

The gxtran utility application is used to manipulate and view .gmf files. It is most commonly used in LINUX environment. The syntax is described below:

```
  ga-> ! gxtran option -i filemane.gmf
```

The option can be:
- a  Animate the frames without giving the enter on each frame change
- r  Reverts background colors
- g  pixel size

Note: Press <enter> to exit gxtran

<table>
<thead>
<tr>
<th>Example 32: Generating a .gmf and viewing with gxtran</th>
</tr>
</thead>
<tbody>
<tr>
<td>ga-&gt; c</td>
</tr>
<tr>
<td>ga-&gt; enable print archive2.gmf</td>
</tr>
<tr>
<td>ga-&gt; d tmpprs (z=1)</td>
</tr>
<tr>
<td>ga-&gt; print</td>
</tr>
<tr>
<td>ga-&gt; c</td>
</tr>
<tr>
<td>ga-&gt; d tmpprs (z=3)</td>
</tr>
<tr>
<td>ga-&gt; print</td>
</tr>
<tr>
<td>ga-&gt; c</td>
</tr>
<tr>
<td>ga-&gt; d tmpprs (z=5)</td>
</tr>
</tbody>
</table>
You will better use GV and you will the manipulations are easy

4.4 Applications gxps and gxeps

The **gxps** utility application (both windows and linux versions) converts `.gmf` files to *PostScript* (.ps) format images. To do so the syntax is:

```
! gxps option -i archive.gmf -o archive.ps
```

-option can be:
- -c color format
- -r black background
- -d puts CTRL-D at the end of file

The **gxeps** utility application (both windows and linux versions) also converts `.gmf` files to *PostScript* (.eps) formatted images, with additional options, according to the syntax below:

```
! gxeps option -i archive.gmf -o archive.eps
```

-option can be:
- -c color format
- -r black background
- -d puts CTRL-D at the end of file
- -1 PostScript Level 1
- -2 PostScript Level 2
- -a A4-size page
- -l Letter-size page
- -L Prompt for a label to be placed in the figure
- -n Prompt for a note to be included in the file
- -v verbose mode

**NOTE:** In both **gxps** and **gxeps**, if you do not specify `-c` the image will be in grayscale on the white background.

4.5 printim and wi commands

The **printim** command is used to convert the graphic content of the window into an image type file (GIF or PNG), according to the syntax below:

```
printim archive.out option
```

-option can be:
- gif generates GIF image (default: png image)
- Black background black
- White background white
- XNNN horizontal pixel size
- YNNN vertical pixel size

The **wi** command uses the ImageMagick library interface converts the graphic content of the window into an image type file (several format), according to the syntax below:

```
wi archive.out
```

The ImageMagick formatting options to be chosen in the .out extension are: *gif, bmp, cgm, eps, fax, ico, jpeg, pcx, hdf and others* ...

Notes:
✓ printim also works in batch mode, but only in GrADS version 1.8 or higher
✓ wi does not run in batch mode, as it requires an X-server. Some ImageMagick formats (TIFF, PNG, MPEG, etc.) do not work in GrADS. In this case, the generated image will be MIFF type. If no extension is specified, GIF is the default format.

4.6 Application examples and exercises

<table>
<thead>
<tr>
<th>Example 33:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertical section (Longitude x Height) of UR and Wind (Uvel; Omega) with generation of .gmf to be placed in Word document as figure</td>
</tr>
</tbody>
</table>

```
$ ga> open gfs_sample.ctl
$ ga> set lon -100 0
$ ga> set lat 0
$ ga> set z 1 7
$ ga> enable print ex33.gmf
$ ga> set gxout shaded
$ ga> set cmin 0.5
$ ga> set cint 0.1
$ ga> d rhprs
$ ga> set gxout contour
$ ga> set ccolor 0
$ ga> set cmin 0.5
$ ga> set cint 0.1
$ ga> d rhprs
$ ga> set gxout vector
$ ga> set ccolor 1
$ ga> set arrsc1 1.5 50
$ ga> set arrowhead -0.5
$ ga> set cthick 10
$ ga> d ugrdprs; vvelprs*(−100)
$ ga> draw title Vertical section of Rh and wind
$ ga> draw xlab Longitude
$ ga> draw ylab Altitude (Pressure Levels)
$ ga> print
$ ga> disable print
```

After generating ex33.gmf, open it in GV and put (copy; paste) in your Word document as figure.
Example 34:
Graph lines with generation .gmf to be placed in Word as figure

```
# Example code for generating graph lines

# Enable print
ga> enable print ex34.gmf

# Set parea
ga> set parea 2 8.5 1 7.7

# Set lon
ga> set lon -100 0

# Set lat
ga> set lat 0

# Set grid
ga> set grid off

# Set grads
ga> set grads off

# Set xaxis
ga> set xaxis 1 11 1

# Set xlopts
ga> set xlopts 1 1 0.2

# Set gxout
ga> set gxout line

# Set ylopts
ga> set ylopts 2 1 0.12

# Set t
ga> set t 3

# Draw pratesfc
ga> d pratesfc

# Set ccolor
ga> set ccolor 4

# Set ylopts
ga> set ylopts 4 1 0.12

# Draw tcdcclm
ga> d tcdcclm

# Set ccolor
ga> set ccolor 3

# Set ylopts
ga> set ylopts 3 1 0.12

# Draw ulwrftoa
ga> d ulwrftoa

# Set strsiz
ga> set strsiz 0.4 0.3

# Set string
ga> set string 2

# Draw string
ga> draw string 2.5 8 Precipitation

# Set string
ga> set string 4

# Draw string
ga> draw string 4.5 8 Cloud Cover

# Set string
ga> set string 3

# Draw string
ga> draw string 6.5 8 OLR

# Draw xlab
ga> draw xlab time

# Print
ga> print

# Disable print
ga> disable print
```
5 VARIABLES, EXPRESSIONS AND FUNCTIONS

5.1 Names of Variables

The complete specification for a variable name is:

<table>
<thead>
<tr>
<th>abbrev.file# (dimexpr, dimexpr,...)</th>
<th>abbrev</th>
<th>Abbreviation for the variable as specified in the .ctl file</th>
</tr>
</thead>
<tbody>
<tr>
<td>file#</td>
<td>file#</td>
<td>The reference number of the opened files containing the variable. The default is 1 (first file to be opened). The command set dfile file # change the default file.</td>
</tr>
<tr>
<td>dimexpr</td>
<td>dimexpr</td>
<td>Expression of the dimension that locally modifies the environment of the current dimension only for the variable in question. Only fixed dimensions can be used.</td>
</tr>
</tbody>
</table>

Absolute dimensions are:
X | Y | Z | T | Lon | Lat | Time = value

The relative dimensions are, for example:
X | Y | Z | T | Lon | Lat | Lev | Time + __/ valor

Here are some examples of variable specifications:
- zgeo.3(lev=500) zgeo in file 3 , taken at the level 500 hPa (absolute dimension)
- prec(time-12hr) precipitation 12 hr before the current time (relative dimension)
- uvel.2(t-1,lev=850) expression using both relative and absolute dimensions

Note:
Lat, lon, lev are predefined by GrADS variables, i.e. they are implicitly contained within each .ctl file. When used, they provide the lat, lon, lev in the respective grid points, for example lat.2 specifies the latitude of the second open grid .ctl.

Example 35:
Using Expressions ...

```
   ga-> set map 3 1 10
   ga-> set lon -90 -30
   ga-> set lat -35 10
   ga-> set lev 1000
   ga-> set cthick 10
   ga-> set arrscl 1 10
   ga-> set arrowhead -0.5
   ga-> d skip(ugrdprs,2); vgrdprs
   ga-> set gxout stream
   ga-> set ccolor 2
   ga-> set strmden 2
   ga-> d ugrdprs (lev=200); vgrdprs (lev=200)
```


5.2 Defining New Variables: define command

The *define* command allows the interactive creation of new variables, according to the syntax:

```
define new-variable-name = expression
```

The new variable is stored in memory and can be used in subsequent commands. It is possible to use *define* with dimensions ranging from 0 to 4. When Z and / or T are varying, *define* evaluates the expression for each Z and T.

To clear the memory and undefining your new variable use the *undefine* command, according to the syntax:

```
undefine new-variable-name
```

**Example 36:**

Defining a variable for several vertical levels

```
ga -> set lon -90 -30
nga -> set lat -35 10
nga -> set lev 1000 200
nga -> define tempc = tmpprs - 273
nga -> set lev 1000
ga -> d tempc
nga -> set lev 500
ga -> d tempc
```

5.3 Expressions

Similarly to FORTRAN, expressions in GrADS consist of operators, operands, and parentheses, which are used to control the order of calculations in operations. The operators are: + (addition), - (subtraction), * (multiplication), / (division) The operands can be: variable specifications, functions and constants

**Note:** The operations are performed for each grid point and therefore the grids must have the same dimensions.

Example:

```
hgtps - hgtprs (t-1)
tmpprs (lev=500) - tmpps (lev=850)
```

5.4 Functions

Grad has a wide range of intrinsic functions. The list below enumerates some of them according to their specific assignments, as well as the syntax of each one.
### Mathematical operations:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs(expr)</td>
<td>Provides the absolute value of expr. Missing data values do not participate.</td>
</tr>
</tbody>
</table>
| cdiff(expr,dim) | Performs a centered difference operation on expr in the direction specified by dim. The difference is done in the grid space, and no adjustment is performed for unequally spaced grids. The result value at each grid point is the value at the grid point plus one minus the value at the grid point minus one. Result values at the grid boundaries are set to missing. Example: Calculation of the temperature advection  
  \[
  \text{define } dx = \text{cdiff(temp,x)} \\
  \text{define } dy = \text{cdiff(temp,y)} \\
  \text{define } dx = \text{cdiff(lon,x)}*3.1416/180 \\
  \text{define } dy = \text{cdiff(lat,y)}*3.1416/180 \\
  \text{d } = 1*(uvel*dtx)/(cos(lat*3.1416/180)*dx) + vvel*dty/dy )/6.37e6
  \]
| exp(expr)    | Provide the exponential of expr                                              |
| gint (expr,dim1, dim2) | Provide the general integral of expr (similar to the ave, but not divided by the total area). dim1 and dim2 represents the start and the end point of the integral respectively. |
| log(expr)    | Takes the natural logarithm of expr. Values less than or equal to zero are set to missing in the result. |
| log10(expr)  | Same as above, but for the logarithm to the base 10                          |
| pow(expr1,expr2) | Raises the values of expr1 to the power of expr2                             |
| sqrt(expr)   | Takes the square root of the result of the expr. Values in expr that are less than zero are set to missing in the result |
| vint(psexpr,expr,top) | Performs a mass-weighted vertical integral in mb pressure coordinates  
  \[
  \text{psexpr } \text{surface pressure, in mb, which bounds the integral on the bottom} \\
  \text{expr expression representing the quantity to be integrated} \\
  \text{top top pressure, in mb. This value must be a constant and cannot be provided as an expression} \\
  \]
  Example: calculation of precipitable water in mm  
  \[
  \text{vint(psnm,umcs,275)}
  \]

### Trigonometric Functions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cos(expr)</td>
<td>Takes the cosine of the expr. Values are assumed to be in radians</td>
</tr>
<tr>
<td>acos(expr)</td>
<td>Applies the inverse cosine function to the result of expr. Values from expr that exceed 1 or are less than -1 are set to missing. The result is expressed in radians.</td>
</tr>
<tr>
<td>sin(expr)</td>
<td>Takes the sin of the provided expression. It is assumed the expression is in radians. Result values are in the range -1 to 1</td>
</tr>
<tr>
<td>asin(expr)</td>
<td>Same as acos, but use inverse sinus function.</td>
</tr>
<tr>
<td>tan(expr)</td>
<td>Trigonometric tangent function to the expr which is assumed to be in radians</td>
</tr>
<tr>
<td>atan2 (expr1, expr2)</td>
<td>Applies the inverse tangent function to the result of (expr1/expr2). If expr1 and expr2 are both zero, the result is arbitrarily set to zero. The result of the atan2 function is in radians.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>aave(expr, xdim1, xdim2, ydim1, ydim2)</td>
<td>Areal average over an X-Y region</td>
</tr>
<tr>
<td>expr</td>
<td>Expression of the variable</td>
</tr>
<tr>
<td>xdim1</td>
<td>Starting X dimension expression</td>
</tr>
<tr>
<td>xdim2</td>
<td>Ending X dimension expression</td>
</tr>
<tr>
<td>ydim1</td>
<td>Starting Y dimension expression</td>
</tr>
<tr>
<td>ydim2</td>
<td>Ending X dimension expression</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>In case the average on the global is needed:</td>
</tr>
<tr>
<td></td>
<td>aave(expr, lon=0, lon=360, lat=-90, lat=90)</td>
</tr>
<tr>
<td></td>
<td>or aave(expr, global)</td>
</tr>
<tr>
<td></td>
<td>or aave(expr, g)</td>
</tr>
<tr>
<td>amean (expr, xdim1, xdim2, ydim1, ydim2)</td>
<td>Same as aave in all respects except one: area means are not weighted by latitude. Means are weighted by grid interval to account for non-linear grid spacing.</td>
</tr>
<tr>
<td>asum(expr, xdim1, xdim2, ydim1, ydim2)</td>
<td>Areal sum over an X-Y region</td>
</tr>
<tr>
<td>asumg(expr, xdim1, xdim2, ydim1, ydim2)</td>
<td>Same as asum, except the calculations are done without weighting</td>
</tr>
<tr>
<td>ave(expr, dim1, dim2 &lt;,tinc&gt; &lt;,-b&gt;)</td>
<td>Averages the result of expr over the specified dimension range. If the summing dimension is time, an optional time increment tinc may be specified.</td>
</tr>
<tr>
<td>expr</td>
<td>Expression of the variable</td>
</tr>
<tr>
<td>dim1</td>
<td>Starting point of average</td>
</tr>
<tr>
<td>dim2</td>
<td>Ending point of average</td>
</tr>
<tr>
<td>tinc</td>
<td>Optional increment for time averaging</td>
</tr>
<tr>
<td>-b</td>
<td>Use exact boundaries</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Zonal mean of the global temperature:</td>
</tr>
<tr>
<td></td>
<td>ave(temp,lon=0,lon=360)</td>
</tr>
<tr>
<td></td>
<td>Annual rainfall standard deviation (30 year time series):</td>
</tr>
<tr>
<td></td>
<td>define cli = ave(prec,t=1,t=30)</td>
</tr>
<tr>
<td></td>
<td>sqrt(ave(pow(cli-prec,2),t=1,t=30))</td>
</tr>
<tr>
<td>mean (expr, dim1, dim2 &lt;,tinc&gt; &lt;,-b&gt;)</td>
<td>Same as ave, except the calculations in the Y dimension are not weighting by latitude. The means are weighted by grid interval to account for non-linear grid spacing</td>
</tr>
<tr>
<td>sum (expr, dim1, dim2 &lt;,tinc&gt; &lt;,-b&gt;)</td>
<td>Sums the result of expr over the specified dimension range.</td>
</tr>
<tr>
<td>sumg (expr, dim1, dim2 &lt;,tinc&gt; &lt;,-b&gt;)</td>
<td>Same as sum, except the calculations are done without weighting</td>
</tr>
<tr>
<td>tmave(maskexpr,expr,timexpr1,timexpr2)</td>
<td>This function does time averaging while applying a mask</td>
</tr>
<tr>
<td>maskexpr</td>
<td>The mask expression must be a single value when evaluations are done at a fixed time</td>
</tr>
<tr>
<td>expr</td>
<td>expression to be averaged</td>
</tr>
<tr>
<td>timexpr1,2</td>
<td>limits of the time averaging domain</td>
</tr>
</tbody>
</table>
Correlation and regression:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Parameters</th>
<th>Example</th>
</tr>
</thead>
</table>
| `scorr(expr1, expr2, xdim1, xdim2, ydim1, ydim2)` | Gives the spatial correlation between two variables over an X-Y domain. It returns a single number (between -1 and 1) | `expr1` Any valid expression varying X and Y  
`expr2` Any valid expression varying X and Y  
`xdim1` Starting X dimension expression  
`xdim2` Ending X dimension expression  
`ydim1` Starting Y dimension expression  
`ydim2` Ending X dimension expression | Correlation between annual precipitation and long wave radiation over Brazil  
```plaintext
set lat -35 5  
set lon -80 -30  
d scorr(prec, role, lon=-80, lon=-30, lat=-35, lat=5)
``` |
| `tcorr (expr1, expr2, tdim1, tdim2)` | Produces a spatial map of temporal correlation coefficients | `expr1` Any time varying valid expression  
`expr2` Any valid expression varying not only in time, but also in X and Y  
`tdim1` Starting time dimension expression  
`tdim2` Starting time dimension expression | Correlation between the 30-year series of annual rainfall in Belém and Long wave over tropical Brazil  
```plaintext
set lat -1.5  
set lon -48  
set z 1  
set t 1 30  
define belem = prec  
set lon -80 -30  
set lat -15 5  
set z 1  
set t 1  
d tcorr(belem, role, t=1, t=30)
``` |
| `sregr(expr1, expr2, xdim1, xdim2, ydim1, ydim2)` | Calculates the least-squares regression between two variables over an X-Y domain. It returns a single number. See `scorr` for the parameters definitions. |  |  |
| `tregr (expr1, expr2, tdim1, tdim2)` | Calculates the least-squares regression between two time-dependent variables. See `tcorr` for the parameters definitions. |  |  |
### Derived weather variables and vector operations

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tvrh2q(tvexpr,rhexpr)</code></td>
<td>Returns specific humidity (q, in g/g), from virtual temperature and relative humidity.</td>
</tr>
<tr>
<td><code>tvexpr</code></td>
<td>Virtual temperature (in Kelvin)</td>
</tr>
<tr>
<td><code>rhexpr</code></td>
<td>Relative humidity (in %, value from 0 to 100)</td>
</tr>
<tr>
<td><code>tvrh2t(tvexpr,rhexpr)</code></td>
<td>Returns temperature (in Kelvin), from virtual temperature and relative humidity. For parameters, see <code>tvrh2q</code>.</td>
</tr>
<tr>
<td><code>hcurl(uexpr,vexpr)</code></td>
<td>Returns vorticity at each grid, from the zonal (uexpr) and meridional (vexpr) wind.</td>
</tr>
<tr>
<td><code>hdivg(uexpr,vexpr)</code></td>
<td>Takes the zonal (uexpr) and meridional (vexpr) to compute the horizontal divergence using finite differencing.</td>
</tr>
<tr>
<td><code>mag(uexpr,vexpr)</code></td>
<td>Returns the horizontal wind speed from expressions of zonal (uexpr) and meridional (vexpr) wind.</td>
</tr>
<tr>
<td><code>skip (expr, skipx, skipy)</code></td>
<td>Sets alternating values of expr to the missing data value. Used mainly to decrease the density of vectors and barbs.</td>
</tr>
<tr>
<td><code>expr</code></td>
<td>A valid grid expression with 1 or 2 varying dimensions</td>
</tr>
<tr>
<td><code>skipx</code></td>
<td>Skip factor in the X dimension of expr</td>
</tr>
<tr>
<td><code>skipy</code></td>
<td>Skip factor in the Y dimension of expr</td>
</tr>
</tbody>
</table>

### Grid point operations:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fndlvl (expr, expr_to_find, lev1, lev2)</code></td>
<td>Given two gridded variables, <code>expr</code> and <code>expr_to_find</code>, this function finds the first vertical level at which the <code>expr_to_find</code> value occurs in <code>expr</code>. <code>lev1</code> and <code>lev2</code> specify the range of levels over which to search. The result is a grid of pressure values.</td>
</tr>
<tr>
<td><code>max(expr, dim1, dim2 &lt;,tinc&gt;)</code></td>
<td>Returns the maximum of <code>expr</code> over the specified dimension range. If the specified dimension is time, an optional time increment <code>tincr</code> may be specified.</td>
</tr>
<tr>
<td><code>maxloc(expr, dim1, dim2 &lt;,tinc&gt;)</code></td>
<td>Returns the grid coordinate for the maximum of <code>expr</code> over the specified dimension range.</td>
</tr>
<tr>
<td><code>min(expr, dim1, dim2 &lt;,tinc&gt;)</code></td>
<td>Returns the minimum of <code>expr</code> over the specified dimension range.</td>
</tr>
<tr>
<td><code>minloc(expr, dim1, dim2 &lt;,tinc&gt;)</code></td>
<td>Returns the grid coordinate for the minimum of <code>expr</code> over the specified dimension range.</td>
</tr>
<tr>
<td><code>smth9(expr)</code></td>
<td>Performs a 9 point smoothing to the gridded result of the <code>expr</code></td>
</tr>
</tbody>
</table>
Other:

| const (expr, value, <u|a>) | Change the missing values of a variable, set all the non-missing values of a variable to a constant, or set all possible values of a variable (both valid and missing) to a constant. |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| expr                     | Any valid expression                                                                                                                 |
| value                    | a constant, either an integer or floating point value                                                                               |
| -u                       | all missing data are set to value; non-missing data are unchanged                                                                   |
| -a                       | all data are set to value, both missing and non-missing                                                                              |

Example:
Plot a horizontal line on a graph line figure
set lon 0
set lat -35 10
set gxout linefill
set lev 1000
d const((tmpprs -273), -20); tmpprs -273

maskout(expr,mask)
For values in expr, put missing data value wherever the mask values are less than zero

5.5 Application examples and exercises

Example 37:
Using functions for the calculation of derived variables (write example37.gs)

```
'*open gfs_sample.ctl'

'*enable print ex37.gmf'
'set lon -20 55'
'set lat -40 40'
'set lev 1000 200'
'define medz = ave(vvelprs, lat=-5, lat=5)'

'set vpage 0 11 4.25 8.5'
'set lat 0'
'set gxout shaded'
'd medz'
'set gxout contour'
'd medz'
'draw title Zonal mean of Vertical Velocity'
'set vpage off'

'set lon -20 55'
'set lat -40 40'
'set lev 200'
'define vort = hcurl(ugrdprs,vgrdprs)'

'set lev 850'
'define dive = hdivg(ugrdprs,vgrdprs)'

'set map 15 1 10'

*(For next, see the right column)*

*(continued from the left column)*

'set vpage 0 5.5 0 5'
'set clopts 1 1 .15'
'set grads off'
'set grid off'
'set gxout shaded'
'set black -.5 .5'
'd dive/1e-5'
'set gxout contour'
'set black -.5 .5'
'd dive/1e-5'
'draw title Divergence at 850 hPa'
'set vpage off'

'set vpage 5.5 11 0 5'
'set grads off'
'set grid off'
'set gxout shaded'
'set black -.5 .5'
'd vort/1e-5'
'set gxout contour'
'set black -.5 .5'
'd vort/1e-5'
'draw title Vorticity at 200 hPa'
'set vpage off'

'enable print ex35.gmf'
'print'
```
6 SCRIPTING LANGUAGE (script.gs)

6.1 Basic Concepts

GrADS has a scripting language in which, basically, the user writes a sequence of command lines using any text editor (outside of GrADS) and then saves that program, for example, with the name of `program1.gs`. The `program1.gs` file is defined as a script (the `.gs` extension would be the acronym for grads script) to run within the GrADS prompt.

The command to run a script within the GrADS prompt is:

```
ga-> run script-file-name.gs    or    ga-> script-file-name
```

Note:

- Each line of the script must be enclosed in ' (apostrophes), as shown below:
  
  * `in this the script we want to display the temperature field`
  
  'open example.ctl'
  
  'd temp'

- Within scripts, lines beginning with the * symbol are interpreted as comments (see the example above)

- The user can also write a script without using the apostrophes, but the execution of the script is done through the command: `ga-> exec script-file-name.gs`

  ➤ Automatic script execution: `set imprun`

  The command `ga-> set imprun script-file-name.gs` automatically executes the same before a command `ga-> d variable` as shown below

---

**Example 38:**

Starting to create a library of scripts to facilitate and/or speed up our life in GrADS prompt

Open your text editor and type the below commands, save the file under the name `gshaded.gs`.

* Script made by the trainee, to plot temperature in shaded mode

  `set gxout shaded`
  
  `set clevs 200 210 220 230 240 250 260 270 280 290 300 310`
  
  `set ccols 9 14 11 5 13 3 10 7 12 8 2 6`

Open your text editor and type the below commands, save the file under the name `gcontour.gs`.

* Script made by the trainee, to plot temperature in contour mode

  `set gxout contour`
  
  `set ccolor 1`
  
  `set clab on`
  
  `set clskip 2`

Ok ... now load GrADS into portrait mode and run the commands as shown below ... see that the sequence of commands has become "cleaner" ...
Example 39:
Using a new .ctl (Precipitation and OLR monthly data observed between 1983 and 2016 i.e 34 years)

file olr_month.ctl:
DSET *olr_month.dat
UNDEF -9999.0
TITLE Monthly mean OLR data (Jan 1983 ~ Dec 2016 )
XDEF 144 linear 0.0 2.5
YDEF 73 linear -90.0 2.5
ZDEF 1 LEVELS 1
TDEF 408 LINEAR 01Jan1983 1mo
VARS 1
olr 1 99 monthly mean OLR (W/m*m)
ENDVARS

file rain_arc_month.ctl:
DSET *rain_arc_month.dat
TITLE Africa monthly Precip (Jan 1983 ~ Dec 2016 )
UNDEF -999.0
XDEF 751 LINEAR -20 0.1
YDEF 801 LINEAR -40 0.1
ZDEF 1 LEVELS 1
TDEF 408 LINEAR 01Jan1983 1mo
VARS 1
rain 0 99 ch08 merged analysis
ENDVARS

Write a script (example39.gs herafter ex39.gs), including the following actions:
- Definition of new colors,
- calculating climatological average,
- running scripts (cbarc.gs, cores.gs) inside the ex39.gs,
- putting comments etc.
reinit'
'open rain_months.ctl '

* New color script
'color'

* Coordinates of the African region
'set lat -40 40'; 'set lon -20 55'

* Define rainfall monthly climatology mean
* for the 34 years of records
'define janrainclim=ave(rain.1, t=1, t=408,12)'
define febrainclim=ave(rain.1, t=2, t=408,12)'
define marrainclim=ave(rain.1, t=3, t=408,12)'
define aprainclim=ave(rain.1, t=4, t=408,12)'
define mayrainclim=ave(rain.1, t=5, t=408,12)'
define junrainclim=ave(rain.1, t=6, t=408,12)'
define julrainclim=ave(rain.1, t=7, t=408,12)'
define augrainclim=ave(rain.1, t=8, t=408,12)'
define seprainclim=ave(rain.1, t=9, t=408,12)'
define octrainclim=ave(rain.1, t=10, t=408,12)'
define novrainclim=ave(rain.1, t=11, t=408,12)'
define decrainclim=ave(rain.1, t=12, t=408,12)'

* Plot rainfall
'set parea 1 5 1 7.5'
'set parea 1 5 1 7.5'; 'set grid off'
'set grid off'; 'set map 15 1 1'

'set parea 6 10 1 7.5'
'set gxout shaded'
'set gxout shaded'
'set ccols 49 48 47 46 45 44 43 42 41'
'set clevs 200 210 220 230 240 250 260 270'
d'mth9(janrainclim)'

'set gxout contour'; 'set cthick 6'; 'set color 2'
'set clevs 200 210 220 230 240 250 260 270'
'd'mth9(janrainclim)'

cbarc 10.5 8.1'

draw title Jan Climatological rainfall'
cbarc 5 8.1'
'set parea off'

close the .ctl file 1
'close 1'

*** The script end here ********************
6.2 Language Elements in Scripts
In general, the GrADS’s scripts contain the following elements:
- Comment
- Statement
- Assignment
- say / prompt / pull
- if / else / endif
- while / endwhile / break / continue
- function header / return

➢ **Comment**: Comments within the scripts should contain the * symbol in the first column.

➢ **Statement**: are the declarations of command lines (expressions in general)

➢ **Assignment**: a claim generally used in general to set a variable = expression

➢ **Concatenation**: 
  - 'set lat 'minlat' 'maxlat
  - 'set lat 'minlat' 'maxlat

**Example 40**: 
Type / Save the following command lines in an example40.gs and then run it in GrADS ... the result is in the figure to the side.

```
'set display color white'
'c'
'open gfs_sample.ctl'
minlat = -40
maxlat = minlat + 80
minlon = -20
maxlon = 55
'set lat 'minlat' 'maxlat
'set lon 'minlon' 'maxlon
'set mpdset hires'
'd pressfc/100'
```

➢ **say / prompt**: is used to provide information or to question the user via the terminal (GrADS prompt), according to the syntax below:
  - `say 'expression`
  - `prompt expression`
Example 41:
Type / Save the following command lines in an example41.gs and then run it in GrADS ... the result is shown in the figure on the side.

```plaintext
expression='Worth It '
say ''
say '============================================'
say ''
say 'Hujambo !!!'
say ''
say 'Hakuna Matata ...'
say ''
say 'Learning GrADS is well 'expression
say ''
say 'Alavida ... Kwaheri ... Dehina yihunu'
say ''
say '============================================'
say ''
```

- **pull** allows to load the information provided by the user through keyboard, with the syntax:
  ```plaintext
  pull variable1 variable2 ...
  ```

Example 42:
Type / Save the following command lines in example42.gs and then run it in GrADS ... the result is shown in the figure on the side.

```plaintext
'open gfs_sample.ctl'
say ''
prompt 'What are Latitudes ?'
pull minlat maxlat
say ''
prompt 'What are Longitudes ?'
pull minlon maxlon
'set lat 'minlat'' '%maxlat
'set lon 'minlon'' 'maxlon
'd pressfc/100'
```

- **if / else / endif** a way to control script execution ... the syntax is:
  ```plaintext
  if expression
  script record
  ...
  Else
  script record
  ...
  endif
  ```

Example:
```
if (i = 10); j = 20; endif
```
➤ **while / endwhile** a way to control script execution ... the syntax is:

```plaintext
while expression
  script record
... 
endwhile
```

**Example 43:**
Making a loop in time ...

```
'open rain_arc_month.ctl '
tt = 1
while (tt <= 25)
  'set t 'tt
  'd rain'
  'c'
  tt = tt + 1
endwhile
```

➤ **Global string variables** are variables that are maintained throughout the script. Any variable name starting with an underscore (_) will be assumed to be a global variable, and will keep its value throughout an entire script file. An example of an assignment statement that defines a global string variable is as follows:

```
_var1 = "global variable 1"
```

➤ **Operators**

<table>
<thead>
<tr>
<th>logical OR</th>
<th>!= not equal</th>
<th>% concatenation</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp; logical AND</td>
<td>&gt;= greater than</td>
<td>+ addition</td>
</tr>
<tr>
<td>! unary NOT</td>
<td>&gt;= greater or equal than</td>
<td>- subtraction</td>
</tr>
<tr>
<td>- unary minus</td>
<td>&lt; less than</td>
<td>* multiplication</td>
</tr>
<tr>
<td>= equal</td>
<td>&lt;= less or equal than</td>
<td>/ division</td>
</tr>
</tbody>
</table>
### Intrinsic functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>strlen(string)</code></td>
<td>This function returns the length (number of characters) of <code>string</code>.</td>
</tr>
<tr>
<td><code>substr(string, start, length)</code></td>
<td>This function gets part of a string. <code>start</code> and <code>length</code> must be an integer. The sub-string of <code>string</code> starting at location <code>start</code> for length <code>length</code> will be returned. If the string is too short, the result will be short or NULL.</td>
</tr>
<tr>
<td><code>sublin(string, n)</code></td>
<td>This function gets a single line from a string containing several lines. <code>n</code> must be an integer. The result is the <code>nth</code> line of <code>string</code>. If the string has too few lines, the result is NULL.</td>
</tr>
<tr>
<td><code>subwrd(string, n)</code></td>
<td>This function gets a single word from a string. <code>n</code> must be an integer. The result is the <code>nth</code> word of <code>string</code>. If the string is too short, the result is NULL.</td>
</tr>
</tbody>
</table>

#### `read(filename)`

This function reads individual records from file `filename`. Repeated calls must be made to read consecutive records. The record may be a maximum of 80 characters.

The result is a string containing **two lines. Use the `sublin` function to separate the result.**

<table>
<thead>
<tr>
<th>Return codes are:</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - ok</td>
<td>the first line is the return code,</td>
</tr>
<tr>
<td>1 - open error</td>
<td>the 2nd line is the record read from the file.</td>
</tr>
<tr>
<td>2 - end of file</td>
<td></td>
</tr>
<tr>
<td>8 - file open for write</td>
<td></td>
</tr>
<tr>
<td>9 - I/O error</td>
<td></td>
</tr>
</tbody>
</table>

Files are opened when the first call to `read` is made for a particular file name. Files are closed when the execution of the script file terminates (note that files remain open between function calls, etc.).

#### `write(filename, record <, append>)`

This function writes records to output file `filename`. On the first call to write for a particular file, the file is opened in write mode. **This will destroy an existing file!** If you use the optional `append` flag, the file will be opened in append mode, and all writes will be appended to the end of the file.

<table>
<thead>
<tr>
<th>Return codes are:</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - ok</td>
<td></td>
</tr>
<tr>
<td>1 - open error</td>
<td></td>
</tr>
<tr>
<td>8 - file open for read</td>
<td></td>
</tr>
</tbody>
</table>

This function closes the named file.

This must be done if you wish to read from a file you have been writing to. This can also be used to rewind a file.

<table>
<thead>
<tr>
<th>Return codes are:</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - ok</td>
<td></td>
</tr>
<tr>
<td>1 - file not open</td>
<td></td>
</tr>
</tbody>
</table>
### Complementary commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>query &lt; option&gt;</code> or <code>q &lt; option&gt;</code></td>
<td>The query command allows the user to get information about a variety of aspects of the current GrADS session. Configuration, plot characteristics, graphics specifics, and file structure are some examples.</td>
</tr>
<tr>
<td><code>q define</code></td>
<td>Lists currently defined variables</td>
</tr>
<tr>
<td><code>q defval v1 i j</code></td>
<td>Returns the value of defined variable v1 at point i,j</td>
</tr>
<tr>
<td><code>q dims</code></td>
<td>Returns current dimension environment</td>
</tr>
<tr>
<td><code>q file n</code></td>
<td>Returns info on file number n. Uses the default file if n is not given.</td>
</tr>
<tr>
<td><code>q files</code></td>
<td>Lists open files</td>
</tr>
<tr>
<td><code>q fwrite</code></td>
<td>Returns status and characteristics of fwrite output file</td>
</tr>
<tr>
<td><code>q gxinfo</code></td>
<td>Returns graphics environment info</td>
</tr>
<tr>
<td><code>q gxout</code></td>
<td>Returns current gxout settings</td>
</tr>
<tr>
<td><code>q pos</code></td>
<td>Waits for mouse click, then returns position plus additional widget information</td>
</tr>
<tr>
<td><code>q shades</code></td>
<td>Lists colors and levels of shaded contours</td>
</tr>
<tr>
<td><code>q time</code></td>
<td>Returns info about time setting</td>
</tr>
<tr>
<td><code>q xy2w v1 v2</code></td>
<td>Converts XY coords to world cords</td>
</tr>
<tr>
<td><code>q xy2gr v1 v2</code></td>
<td>Converts XY coords to grid cords</td>
</tr>
<tr>
<td><code>q w2xy v1 v2</code></td>
<td>Converts world coords to XY cords</td>
</tr>
<tr>
<td><code>q w2gr v1 v2</code></td>
<td>Converts world coords to grid cords</td>
</tr>
<tr>
<td><code>q gr2w v1 v2</code></td>
<td>Converts grid coords to world cords</td>
</tr>
<tr>
<td><code>q gr2xy v1 v2</code></td>
<td>Converts grid coords to XY cords</td>
</tr>
</tbody>
</table>
### Example 44: Calculating climatology and plotting anomalies

```plaintext
*** The script starts here  *****************************************************
'reinit'
'open rain_arc_month.ctl'

* New color script
'color'

*Coordinates of the African region
'set lat -40 40'; 'set lon -20 55'

*===== Define rainfall monthly climatology mean
*===== for the 34 years of records ==========
'define janrainclim=ave(rain.1, t=1, t=408,12)'
'define febrainclim=ave(rain.1, t=2, t=408,12)'
'define marrainclim=ave(rain.1, t=3, t=408,12)'

* plot rainfall anomalies
'set parea 5.9 10.9 0 8.5'
'set grads off'; 'set grid off'
'set mpdset hires'; 'set map 15 1 1'
'set gxout shaded'
'set ccols 29 28 27 26 25 24 23 22 21 0 51 52 53 54 55 56 57 58 59'
'set clevs -50 -40 -35 -30 -25 -20 -15 -10 -5 5 10 15 20 25 30 35 40 50'
'd smth9(rain.1(time=jan2000)-janrainclim)'
'set gxout contour'; 'set clab off'; 'set ccolor 1'
'set clevs -50 -40 -35 -30 -25 -20 -15 -10 -5 5 10 15 20 25 30 35 40 50'
'd smth9(rain.1(time=jan2000)-janrainclim)'

'draw title Jan 2000 rainfall anomalies'
'cbarc 10.9 8.1'
'set parea off'

*close the .ctl file 1
'close 1'

*** The script ends here  *****************************************************
```

```plaintext
'open olr_month.ctl'
*Coordinates of the African region
'set lat -40 40'; 'set lon -20 55'

*===== Define olr monthly climatology mean
*===== for the 34 years of records ==========
'define janolrclim=ave(olr.1, t=1, t=408,12)'
'define febolrclim=ave(olr.1, t=2, t=408,12)'
'define marolrclim=ave(olr.1, t=3, t=408,12)'

* plot OLR
'set parea 0.5 5.5 0 8.5'
'set gxout shaded'
'set ccols 29 28 27 26 25 24 23 22 21 0 51 52 53 54 55 56 57 58 59'
'set clevs -50 -40 -35 -30 -25 -20 -15 -10 -5 5 10 15 20 25 30 35 40 50'
'd smth9(olr.1(time=jan2000)-janolrclim)'
'set gxout contour'; 'set clab on'; 'set ccolor 1'
'set clevs -50 -40 -35 -30 -25 -20 -15 -10 -5 5 10 15 20 25 30 35 40 50'
'd smth9(olr.1(time=jan2000)-janolrclim)'

'cbarc 5.5 7.5'
'draw title Jan OLR anomalies'
'set parea off'

'q time'
res = subwrd(result,3)
'mthyear = substr(res,6,7)
'set strsiz 0.2 0.5'
'draw string 0.5 8.1 Anomalies in ' mthyear

* Generating GIF output file
'printim ex44.gif gif white'
```

*** The script ends here  *****************************************************
7 ADDITIONAL TOPICS

7.1 The Template Option

7.2 Generating binary files with fwrite

7.3 Creating a Mask