A short tutorial on GrADS – Grid Analysis and Display System

Table of Contents

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 comunities 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:

```
DSET   ^model.dat
OPTIONS little_endian
UNDEF  -2.56E33
TITLE 5 Days of Sample Model Output
XDEF 72 LINEAR  0.0 5.0
YDEF 46 LINEAR  -90.0 4.0
ZDEF 7 LEVELS 1000 850 700 500 300 200 100
TDEF 5 LINEAR 02JAN1987 1DY
VARS 8
ps     0   99   Surface Pressure
u      7   99   U Winds
v      7   99   V Winds
hgt    7   99   Geopotential Heights
tair   7   99   Air Temperature
q      5   99   Specific Humidity
tsfc   0   99   Surface Temperature
p      0   99   Precipitation
ENDVARS
```

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)

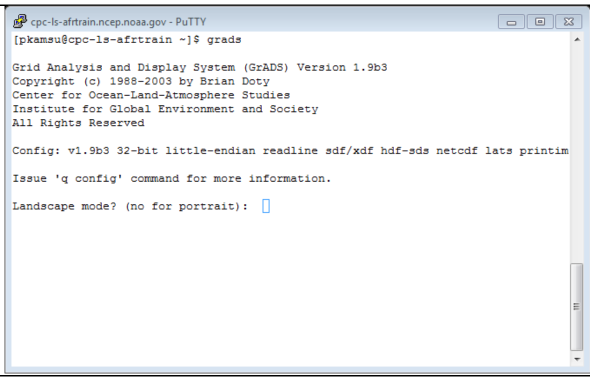
TITLE *5 Days of Sample Model Output*	Title of the data set	
XDEF *72 LINEAR 0.0 5.0*	Zonal (longitude) grid specifications:	number of grid boxes, increment type, minimum, resolution
YDEF *46 LINEAR -90.0 4.0*	Meridional (latitude) grid specifications:	
ZDEF *7 LEVELS 1000 850 700 500 300 200 100*	Vertical 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	
ps 0 99 Surface Pressure	List of variables: name used by GrADS, number of vertical levels, units (used only for grib; use 99 otherwise), description	
u 7 99 U Winds		
v 7 99 V Winds		
hgt 7 99 Geopotential Heights		
tair 7 99 Air Temperature		
q 5 99 Specific Humidity		
tsfc 0 99 Surface Temperature		
p 0 99 Precipitation		
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

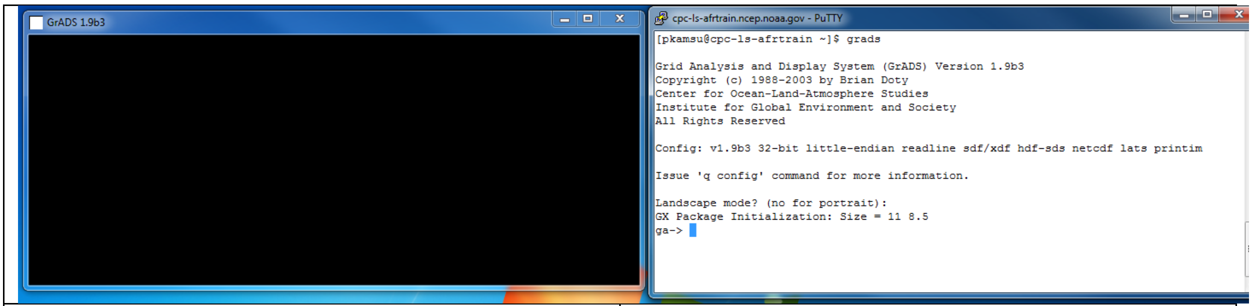
2.3 Running GrADS (initiation session)

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.



Graphics window	Commands window (console)

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

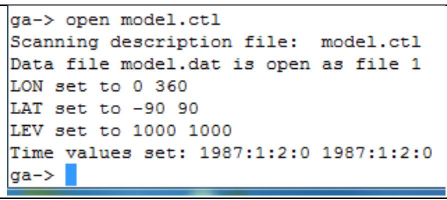
Tip: The GrADS preview screen always opens with the black background, which sometimes makes it difficult to interpret certain graphics. To change the background of the preview screen to white, in the console where you have grads prompt (ga->) type following command:

ga-> **set display color white**

ga-> **clear**

What happened?

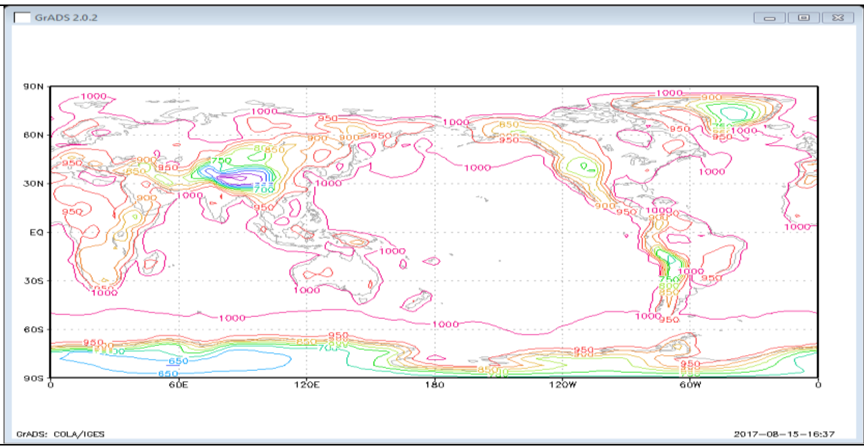
- ✓ Open a data file

Within the GrADS prompt, the command to open the descriptor file (which in turn controls the data file) is done as follows:	Informations that appears at the opening of the .ctl file.
ga-> **open model.ctl**	

- ✓ You may want to see what is in this file, so enter:

query **file**
or
q **file** (q is short for query)
	`ga-> q file`
	`File 1 : 5 Days of Sample Model Output`
	`  Descriptor: model.ctl`
	`  Binary: model.dat`
	`  Type = Gridded`
	`  Xsize = 72  Ysize = 46  Zsize = 7  Tsize = 5`
	`  Number of Variables = 8`
	`    ps 0 99 Surface Pressure`
	`    u 7 99 U Winds`
	`    v 7 99 V Winds`
	`    hgt 7 99 Geopotential Heights`
	`    tair 7 99 Air Temperature`
	`    q 5 99 Specific Humidity`
	`    tsfc 0 99 Surface Temperature`
	`    p 0 99 Precipitation`
	`ga->`

- ✓ This data contains surface pressure, represented by a variable name, ps, display this variable by entering:

display ps or **d ps** (d is short for display)	

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*** extention, and you should be able to see the file you have created while you were in GrADS environemnt.
  - Which linux command did use to open this file?

Note for this initiation section:

Other opening commands are listed in the following table:

grads -l	Opens GrADS in landscape mode
grads -p	Opens GrADS in portrait mode
grads -b	Runs GrADS in batch mode (No window opens)
grads -c "GrADS command line "	Open GrADS and run the quoted command

These options can be used in combinations, such as:

grads -lc "open model.ctl"	Opens GrADS in landscape mode and run the quoted command (open the file model.ctl)
grads -bpc "run scripts.gs"	Opens GrADS in portrait mode, in batch mode (No graphical window opens) run the command in the grads script file script.gs

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:

When	*where*	*how*
ga-> set t 1	**ga-> set lat -20 -10**	**ga-> set gxout line**

2.5 Manipulating Dimensions

The dimensions are manipulated using the ***set*** command, according to the examples below:

ga-> set lat valofLAT1 valofaLAT2	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:

  ga-> set lat -30 -5

  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:

- **ga-> q file** Specifies general information for the descriptor file

File 1 : 5 Days of Sample Model Output Descriptor: model.ctl Binary: model.dat Type = Gridded Xsize = 72 Ysize = 46 Zsize = 7 Tsize = 5 Esize = 1 Number of Variables = 8 ps 0 99 Surface Pressure u 7 99 U Winds v 7 99 V Winds hgt 7 99 Geopotential Heights tair 7 99 Air Temperature q 5 99 Specific Humidity tsfc 0 99 Surface Temperature p 0 99 Precipitation	Results of the command ***q file***

Note: If multiple descriptor files are open, use the following:

ga-> q files	**ga-> q file n**
Specifies general informations for all the descriptors files opened	to have information about the opened descriptor file number n

- **ga-> q dims** Specifies the current dimensions

Default file number is: 1 X is varying Lon = 0 to 360 X = 1 to 73 Y is varying Lat = -90 to 90 Y = 1 to 46 Z is fixed Lev = 1000 Z = 1 T is fixed Time = 00Z02JAN1987 T = 1 E is fixed Ens = 1 E = 1	Results of the command ***q dims***

- **ga-> clear** or **ga-> c** Clear the preview screen Same as above
- **ga-> reinit** Restart GrADS; Close all the opened .ctl
- **ga-> reset** Restart GrADS; But without closing .ctl
- **ga-> !command-line** Run operating system command line
- **ga-> help** 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.**

Example 1: Open GrADS in Portrait mode and plot the pressure variable at the mean sea level. At the GrADS prompt, type: **ga-> set display color white** **ga-> c** **ga-> open gfs_sample.ctl** **ga-> q file** **ga-> d prmslmsl**	***Proposed exercise 1:*** *Open GrADS in Landscape and plot the precipitation field*

Example 2: Plotting two overlapping variables (pressure and horizontal wind). At the GrADS prompt, type: **ga-> c** **ga-> d prmslmsl** **ga-> d ugrdprs;vgrdprs** or **ga-> d skip(ugrdprs,20); vgrdprs**	***Proposed exercise 2:*** *Plot the precipitation field superimposed on the horizontal wind field*

Example 3: Plot of surface temperature for African Countries. At the GrADS prompt, type: **ga-> c** **ga-> set mpdset hires brmap** **ga-> q dims** **ga-> set lat -40 40** **ga-> set lon -20 55** **ga-> d tmpsfc**	***Proposed exercise 3:*** *Plot the map of specific humidity over your country*

Example 4: Map of geopotential at 500 hPa At the GrADS prompt, type: **ga-> c** **ga-> set lev 500**	***Proposed exercise 4:*** *Plot the horizontal wind at 200 hPa*

ga-> d hgtprs	

Example 5: Vertical temperature profile on the center point in Ndjamena At the GrADS prompt, type: **ga-> c** **ga-> set lat 12.15** **ga-> 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: **ga-> reset** **ga-> set lat 0** **ga-> set z 1 7** **ga-> set zlog on** **ga-> d tmpprs**	***Proposed exercise 6:*** *Plot the vertical meridonal 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: **ga-> reinit** **ga-> open rain_arc_month.ctl** **ga-> set lat -40 40** **ga-> set lon -20 55** **ga-> set time jan1992 jun1992** **ga-> 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**	- *Equatorial band*

3 PLOTING GRAPHICS

3.1 Graphics types

There are several graphics options. By default, if the user does not specify graphics output type, of the output will be line type (for 1-dimensional data) and contour type (for2 -dimensional graphs).

The command line to select the graphics output type is:
ga-> set gxout *graphic_type*

The following examples summarize different graphice output options:

Example 9: contours (Isolines) **ga-> open gfs_sample.ctl** **ga-> set display color white** **ga-> c** **ga-> set t 3** **ga-> set mpdset hires** **ga-> set lat -40 40** **ga-> set lon -20 55** **ga-> set gxout contour** **ga-> d tmpprs-273**	

Example 10: shaded contours (colors bands) **ga-> c** **ga-> set gxout shaded** **ga-> d tmpprs-273**	

Example 11: same as Example 10, but here shading a made on grid points. **ga-> c** **ga-> set gxout grfill** **ga-> d tmpprs-273**	

Example 12: Values in the grid points **ga-> c** **ga-> set gxout grid** **ga-> d tmpprs-273**	

Example 13: Vectors (arrows) **ga-> c** **ga-> set gxout vector** **ga-> d ugrdprs;vgrdprs**	

Example 14: streamlines **ga-> c** **ga-> set gxout stream** **ga-> d ugrdprs;vgrdprs**	

Example 15: Wind with barb (synoptic chart) **ga-> c** **ga-> set gxout barb** **ga-> d ugrdprs; vgrdprs**	

Example 16: Shaded in the grid points of the values specified by the ***set fgvals value col value col …*** **ga-> c** **ga-> set gxout fgrid** **ga-> set fgvals 20 4 23 8 26 2** **ga-> d tmpprs-273**	

Example 17: Bar graph and error bar graph **ga-> c** **ga-> set t 3** **ga-> set lat 0** **ga-> set gxout bar** or **ga-> set gxout errbar** **ga-> d pratesfc**	

Example 18: Line Graph **ga-> c** **ga-> set gxout line** **ga-> d pratesfc**	

Example 19: Scatter plot **ga-> c** **ga-> set gxout scatter** **ga-> d tmpsfc-273; tmpprs-273**	

Example 20: Statistics (information) on the data (without graph) **ga-> c** **ga-> set gxout stat** **ga-> d tmpprs**	

ga-> set gxout fwrite	Write (generates) grads fwrite file with binary data (no graph)
ga-> set gxout linefill	Lines with color filling between two lines
ga-> set gxout value	Station value (station points)
ga-> set gxout wxsym	Symbols of the Synoptic map (weather conditions)
ga-> set gxout findstn	Find the nearest station

3.2 Projections maps

The following examples summerise different projection options:

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

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

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

Example 24: ***robinson*** **ga-> reset** **ga-> set mproj robinson** **ga-> set lon –180 180** **ga-> set lat –90 90**	

ga-> d pratesfc (t=2)	

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)**	

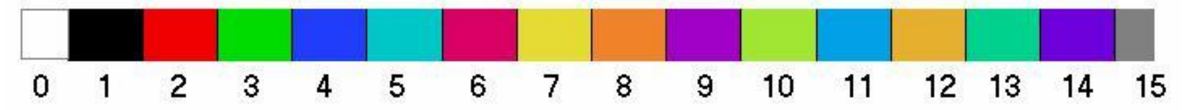
3.3 Inserting Titles, Texts, Forms and Symbols

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

ga-> draw title *graphic-title*	Write title at the top of the picture
ga-> draw xlab *X-Title*	Write title on x-axis
ga-> draw ylab *Y-Title*	Write title on y-axis
ga-> draw string x y *text*	Write text at the point (x, y)
ga-> draw line x1 y1 x2 y2	Draw a line between (x1,y1) and (x2,y2)
ga-> draw rec xlo ylo xhi yhi	Draw a rectangle
ga-> draw recf xlo ylo xhi yhi	Draw a solid (fill) rectangle
ga-> draw polyf x1 y1 x2 y2 ... xn yn	Draws a polygon between (x1,y1), (x2,y2) ... (xn,yn)
ga-> draw mark marktype x y size	Draw a mark on point (x,y)
ga-> draw wxsym symbol x y size color thickness	Draw a weather symbol on point (x,y)

3.4 Controlling Graphical Options

- Color coding:



0 = White	8 = Orange
1 = Black	9 = Purple
2 = Red	10 = Yellow/Green
3 = Green	11 = Medium Blue
4 = Blue	12 = Dark Yellow
5 = Cyan	13 = Aqua
6 = Magenta	14 = Dark Purple
7 = Yellow	15 = Grey

Note: For the 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

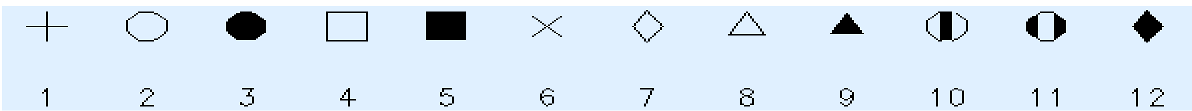
Usage: **ga-> set line** *color style thickness*

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*



0 - none
1 - plus sign
2 - open circle (default)
3 - closed circle
4 - open square
5 - closed square
6 - multiplication sign
7 - open diamond
8 - open triangle
9 - closed triangle
10 - open circle with vertical bar
11 - closed circle with vertical bar
12 - closed diamond (GrADS version 2.1+)

- Weather Symbol code (from 1 to 41, as shown below):

Usage: **ga-> draw wxsym** *symbol x y size color thickness*

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

ga-> q pos (Click the screen on the desired point)
or
ga-> q ll2xy lon lat (No need to click the screen)

- Command to control text (string):

ga-> set string *color justification thickness rotation*
This command sets attributes for strings drawn with the draw string command.

Justification coding:		
tl = top left	tc = top center	tr = top right
l = left	c = center	r = right
bl = bottom left	bc = bottom center	br = bottom right

ga-> set strsiz *hsiz vsiz*
This command sets the string character size, where *hsiz* is the width of the characters; *vsiz* is the height of the characters, in virtual page inches. If *vsiz* is not specified, it will be set the the same value as *hsiz*.

ga-> set font *number*
This command allows the user to select the font for subsequent text operations. With font type (*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*	Set 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

- Graphic type (gxout = **shaded** or **grfill**):

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 csmooth *on|off* — Apply smoothing. If on, the grid is interpolated to a finer grid using cubic interpolation before contouring

- Graphic type (gxout = **grid**):

ga-> set dignum *number* — Number of digits after the decimal place

ga-> set digsiz *size* — Size (in inches, or plotter units) of numbers. 0.1 to 0.15 is usually a good range to use

- Graphic type (gxout = **vector** ou **barb**):

ga-> set ccolor *color#* — Set the color of the vectors

ga-> set cthick *thickness* — Set the thickness of vectors (thickness from 1 to 10)

ga-> set arrlab *on|off* — Shows or not the reference vector below the plot

ga-> set arrscl *size magnitude* — Specifies arrow length scaling. Length of the vector according to magnitude

ga-> set arrowhead *value* — Set the size of the arrowhead

ga-> set cint *value* — Sets the vectors interval to the specified value

ga-> set cmax *value* — Controls the maximum magnitude of the vectors

ga-> set cmin *value* — Controls the maximum magnitude of the vectors

ga-> set black *val1 val2* — Omits vectors of magnitudes 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

- Graphic type (gxout = **scatter**):

ga-> set cmark *markertype* — Set the style of the marker

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 marker's colors

ga-> set vrange *v1 v2* — Set the range of values for the scale on the X-axis

ga-> set vrange2 *v1 v2* — Set the range of values for the scale on the Y-axis

- 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

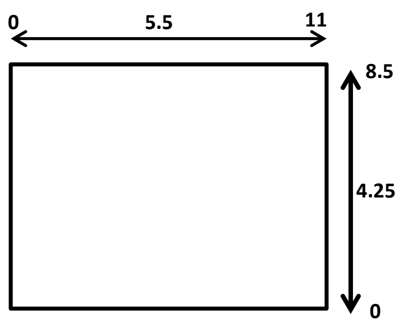
➢ Commands to control axes, maps, etc:

ga-> set grid *status style color# thickness*	Specifies the characteristics of the displayed grid lines. Valid options for *status* are : *on* - both latitude and longitude lines a drawn *off* - grid lines are drawn *horizontal* - only latitude grid lines are drawn *vertical* - only longitude grid lines are drawn
ga-> set zlog *on|off*	Sets log scaling of the Z dimension on or off
ga-> set xaxis *start end <increment>*	Range x-axis from *start* to *end* with *increment*
ga-> set yaxis *start end <increment>*	Range y-axis from *start* to *end* with *increment*
ga-> set xlevs *lev1 lev2 …*	Specify individual labeled tick mark for the X-axis
ga-> set ylevs *lev1 lev2 …*	Specify individual labeled tick mark for the Y-axis
ga-> set xlabs *lab1|lab2| …*	Label the X-axis with lab1, lab2, …
ga-> set ylabs *lab1|lab2| …*	Label the Y-axis with lab1, lab2, …
ga-> set xlint *interval*	Specifies the interval between labeled tick marks on the X-axis
ga-> set ylint *interval*	Specifies the interval between labeled tick marks on the Y-axis
ga-> set xyrev *on|off*	Reverses the axes on a plot
ga-> set xflip *on|off*	Flip the order of the horizontal axis
ga->set yflip *on|off*	Flip the order of the vertical axis
ga-> set xlopts *color# thickness size*	Controls the appearance of the tick labels on the X-axis
ga-> set ylopts *color# thickness size*	Controls the appearance of the tick labels on the Y-axis
ga-> set annot *color# thickness*	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)
ga-> set mpdset *lowres|mres|hires*	Controls the map lines resolution
ga-> set map *color# style thickness*	Controls the appearance of the map lines
ga-> set mpdraw *on|off*	If off, does not draw the map background
ga-> set grads *on|off*	Controls the display of the GrADS logo and the time label for screen or printed output

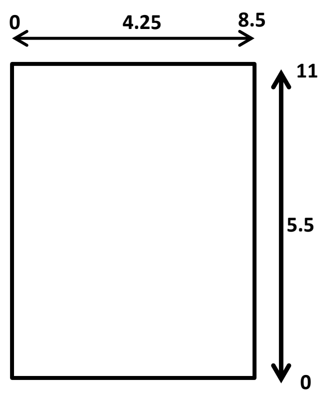
3.5 Page Control

Screen Display standard sizes are:

grads -l (landscape: 11 x 8.5)



grads -p (portrait: 8.5 x 11)



➢ Page can be controlled using the following commands :

Virtual page	**ga-> set vpage** *off*	Default setting, virtual page is equal to real page
	ga-> set vpage *xmin xmax ymin ymax*	Defines a "virtual page" that fits within the specified limits of the real page. All the graphics output will be drawn until another *set vpage* is entered
Print Area	**ga-> set parea** *off*	Default setting, plotting area is chosen depending on the type of the graphics output
	ga-> set parea *xmin xmax ymin ymax*	Specifies the area for plotting contour plots, maps, or lines graphs. This area does not include axis labels, titles, etc.

3.6 Application examples and exercises

<table>
<tr>
<td>
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
</td>
<td>
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.
</td>
</tr>
</table>

Example 30: Two figures on the same portrait page, Rain and Outgoing Long-wave Radiation in Africa **ga-> set mpdset hires** **ga-> set map 1 1 10** **ga-> set grid off** **ga-> set grads 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 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

Example 31: Procedure to generate an *.gmf* file	
ga-> enable print archive1.gmf	Open the file
ga-> d tmpprs	
ga-> print	Save the file
ga-> disable print	Close the file

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

Example 32: Generating a .gmf and viewing with gxtran
ga-> c **ga-> enable print archive2.gmf** **ga-> d tmpprs (z=1)** **ga-> print** **ga-> c** **ga-> d tmpprs (z=3)** **ga-> print** **ga-> c** **ga-> d tmpprs (z=5)**

```
ga-> print
ga-> c
ga-> d tmpprs (z=7)
ga-> print
ga-> disable print
ga-> ! gxtran -a -g 800x600 -i archive2.gmf
```

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:

ga-> ! 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 :

ga-> ! 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:

ga-> 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:

ga-> 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

Example 33: Vertical section (Longitude x Height) of UR and Wind (Uvel; Omega) with generation of .gmf to be placed in Word document as figure
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 arrscl** *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

ga-> c
ga-> enable print *ex34.gmf*
ga-> set parea *2 8.5 1 7.7*
ga-> set lon *-100 0*
ga-> set lat *0*
ga-> set grid *off*
ga-> set grads *off*
ga-> set xaxis *1 11 1*
ga-> set xlopts *1 1 0.2*
ga-> set gxout *line*
ga-> set ccolor *2*
ga-> set ylopts *2 1 0.12*
ga-> set t *3*
ga-> d *pratesfc*
ga-> set ccolor *4*
ga-> set ylopts *4 1 0.12*
ga-> d *tcdcclm*
ga-> set ccolor *3*
ga-> set ylopts *3 1 0.12*
ga-> d *ulwrftoa*
ga-> set strsiz *0.4 0.3*
ga-> set string *2*
ga-> draw string *2.5 8 Precipitation*
ga-> set string *4*
ga-> draw string *4.5 8 Cloud Cover*
ga-> set string *3*
ga-> draw string *6.5 8 OLR*
ga-> draw xlab *time*
ga-> print
ga-> disable print

5 VARIABLES, EXPRESSIONS AND FUNCTIONS

5.1 Names of Variables

The complete specification for a variable name is:

abbrev.file# (dimexpr,dimexpr,...)	**abbrev**	Abbreviation for the variable as specified in the .ctl file
	file#	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.
	dimexpr	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.

Absolute dimensions are:
X | Y | Z | T | Lon | Lat | Lev | 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* **ga-> set lat** *-35 10* **ga-> set lev** *1000 200* **ga-> define** *tempc = tmpprs - 273* **ga-> set lev** *1000* **ga-> d** *tempc* **ga-> 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:

hgtprs - hgtprs (t-1)
tmpprs (lev=500) -tmpprs (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:

abs(expr)	Provides the absolute value of ***expr***. Missing data values do not participate.
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 **define** dtx = cdiff(temp,x) **define** dty = cdiff(temp,y) **define** dx = cdiff(lon,x)*3.1416/180 **define** dy = cdiff(lat,y)*3.1416/180 **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 **psexpr**: surface pressure, in mb, which bounds the integral on the bottom **expr**: expression representing the quantity to be integrated **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 **vint**(*psnm,umes,275*)

➢ Trigonometric Functions:

cos(expr)	Takes the cosine of the ***expr***. Values are assumed to be in radians
acos(expr)	Applies the inverse cosinus 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.
sin(expr)	Takes the sin of the provided expression. It is assumed the expression is in radiians. Result values are in the range -1 to 1
asin(expr)	Same as ***acos,*** but use inverse sinus function.
tan(expr)	Trigonometric tangent function to the ***expr*** which is assumed to be in radians
atan2 (expr1, expr2)	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.

- Averages and sums:

<table>
<tr><td rowspan="7">aave(expr, xdim1, xdim2, ydim1, ydim2)</td><td colspan="2">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 colspan="2">Example:
In case the average on the global is needed :
aave(expr, lon=0, lon=360, lat=-90, lat=90)
or aave(expr, global)
or aave(expr, g)</td></tr>
<tr><td>amean (expr, xdim1, xdim2, ydim1, ydim2)</td><td colspan="2">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 colspan="2">Areal sum over an X-Y region</td></tr>
<tr><td>asumg(expr, xdim1, xdim2, ydim1, ydim2)</td><td colspan="2">Same as asum, except the calculations are done without weighting</td></tr>
<tr><td rowspan="7">ave(expr, dim1, dim2 <,tinc> <,-b>)</td><td colspan="2">Averages the result of expr over the specified dimension range. If the summing dimension is time, an optional time increment tincr 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 colspan="2">Example:
Zonal mean of the global temperature: ave(temp,lon=0,lon=360)
Annual rainfall standard deviation (30 year time series):
define cli = ave(prec,t=1,t=30)
sqrt(ave(pow(cli-prec,2),t=1,t=30))</td></tr>
<tr><td>mean (expr, dim1, dim2, <,tinc> <,-b>)</td><td colspan="2">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, <,tinc> <,-b>)</td><td colspan="2">Sums the result of expr over the specified dimension range.</td></tr>
<tr><td>sumg (expr, dim1, dim2, <,tinc> <,-b>)</td><td colspan="2">Same as sum, except the calculations are done without weighting</td></tr>
<tr><td rowspan="4">tmave(maskexpr,expr,timexpr1,timexpr2)</td><td colspan="2">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>
</table>

- Correlation and regression:

<table>
<tr><td rowspan="8">scorr(expr1, expr2, xdim1, xdim2, ydim1, ydim2)</td><td colspan="2">Gives the spatial correlation between two variables over an X-Y domain. It returns a single number (between -1 and 1)</td></tr>
<tr><td>expr1</td><td>Any valid expression varying X and Y</td></tr>
<tr><td>expr2</td><td>Any valid expression varying X and Y</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 colspan="2">Example:
Correlation between annual precipitation and long wave radiation over Brazil
set lat -35 5
set lon -80 -30
d scorr(prec, role, lon=-80, lon=-30, lat=-35, lat=5)</td></tr>
<tr><td rowspan="6">tcorr (expr1, expr2, tdim1, tdim2)</td><td colspan="2">Produces a spatial map of temporal correlation coefficients</td></tr>
<tr><td>expr1</td><td>Any time varying valid expression</td></tr>
<tr><td>expr2</td><td>Any valid expression varying not only in time, but also in X and Y</td></tr>
<tr><td>tdim1</td><td>Starting time dimension expression</td></tr>
<tr><td>tdim2</td><td>Starting time dimension expression</td></tr>
<tr><td colspan="2">Example: Correlation between the 30-year series of annual rainfall in Belém and Long wave over tropical Brazil
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)</td></tr>
<tr><td>sregr(expr1, expr2, xdim1, xdim2, ydim1, ydim2)</td><td colspan="2">Calculates the least-squares regression between two variables over an X-Y domain. It returns a single number. See scorr for the parameters definitions.</td></tr>
<tr><td>tregr (expr1, expr2, tdim1, tdim2)</td><td colspan="2">Calculates the least-squares regression between two time-dependent variables. See tcorr for the parameters definitions.</td></tr>
</table>

- Derived weather variables and vector operations

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

- Grid point operations:

fndlvl (expr, expr_to_find, lev1, lev2)	Given two gridded variables, ***expr*** and ***expr_to_find***, this function finds the first vertical level at which the ***expr_to_find*** value occurs in ***expr***. ***lev1*** and ***lev2*** specify the range of levels over which to search. The result is a grid of pressure values.
	Example: Find the pressure levels of the 30 degree isotherm between 1000 and 200 hPa **d fndlvl (tmpprs, const(tmpprs,30), lev=1000, lev=200)**
max(expr, dim1, dim2 <,tinc>)	Returns the maximum of ***expr*** over the specified dimension range. If the specified dimension is time, an optional time increment ***tincr*** may be specified.
maxloc(expr, dim1, dim2 <,tinc>)	Returns the grid coordinate for the maximum of ***expr*** over the specified dimension range.
min(expr, dim1, dim2 <,tinc>)	Returns the minimum of ***expr*** over the specified dimension range.
minloc(expr, dim1, dim2 <,tinc>)	Returns the grid coordinate for the minimum of ***expr*** over the specified dimension range.
smth9(expr)	Performs a 9 point smoothing to the gridded result of the ***expr***

- Other:

<table>
<tr><td rowspan="6">const (expr, value, <-u|-a>)</td><td colspan="2">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.</td></tr>
<tr><td>expr</td><td>Any valid expression</td></tr>
<tr><td>value</td><td>a constant, either an integer or floating point value</td></tr>
<tr><td>-u</td><td>all missing data are set to value; non-missing data are unchanged</td></tr>
<tr><td>-a</td><td>all data are set to value, both missing and non-missing</td></tr>
<tr><td colspan="2">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</td></tr>
<tr><td>maskout(expr,mask)</td><td colspan="2">For values in expr, put missing data value wherever the mask values are less than zero</td></tr>
</table>

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

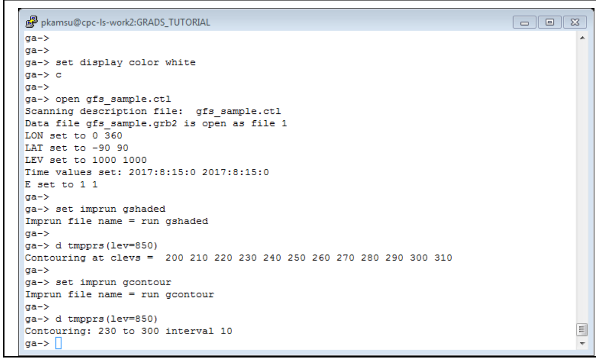
Open your text editor and type the below commands, save the file under the name gshaded.gs.

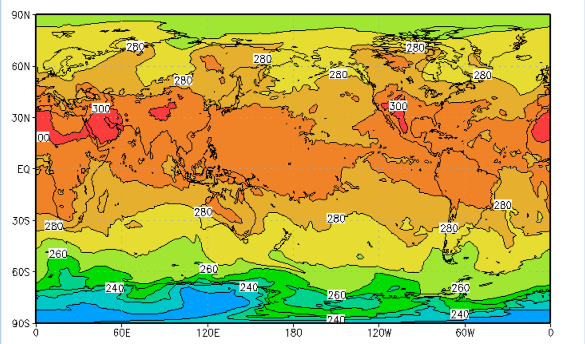
```
* Script made by the trainee, to plot temperature in shaded mode
'set gxout shaded'
 'set clevs 200 210 220 230 240 250 260 270 280 290 300 310'
 'set ccols 9 14 4 11 5 13 3 10 7 12 8 2 6'
```

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:	file rain_arc_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	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.

```
*** 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)'
*'define aprrainclim=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 grads off'; 'set grid off'
'set mpdset hires'; 'set map 15 1 1'

'set gxout shaded'
'set ccols 29 27 25 23 21 32 34 36 38 39'
'set clevs 40 50 60 70 80 90 100 120 140'
'd smth9(janrainclim)'

'set gxout contour'; 'set clab off'; 'set ccolor 1'
'set clevs 40 50 60 70 80 90 100 120 140'
'd smth9(janrainclim)'

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

*close the .ctl file 1
'close 1'

'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)'
*'define aprolrclim=ave(olr.1, t=4, t=408,12)'
*'define mayolrclim=ave(olr.1, t=5, t=408,12)'
*'define junolrclim=ave(olr.1, t=6, t=408,12)'
*'define julolrclim=ave(olr.1, t=7, t=408,12)'
*'define augolrclim=ave(olr.1, t=8, t=408,12)'
*'define sepolrclim=ave(olr.1, t=9, t=408,12)'
*'define octolrclim=ave(olr.1, t=10, t=408,12)'
*'define novolrclim=ave(olr.1, t=11, t=408,12)'
*'define decolrclim=ave(olr.1, t=12, t=408,12)'

* plot OLR
'set parea 6 10 1 7.5'
'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 smth9(janolrclim)'

'set gxout contour'; 'set clab off'; 'set cthick 6'; 'set ccolor 2'
'set clevs 200 210 220 230 240 250 260 270'
'd smth9(janolrclim)'

'cbarc 10.5 8.1'

'draw title Jan Climatological OLR'
'set parea off'

* Generating GIF output file
'printim ex39.gif gif white'

*** The script ends 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)

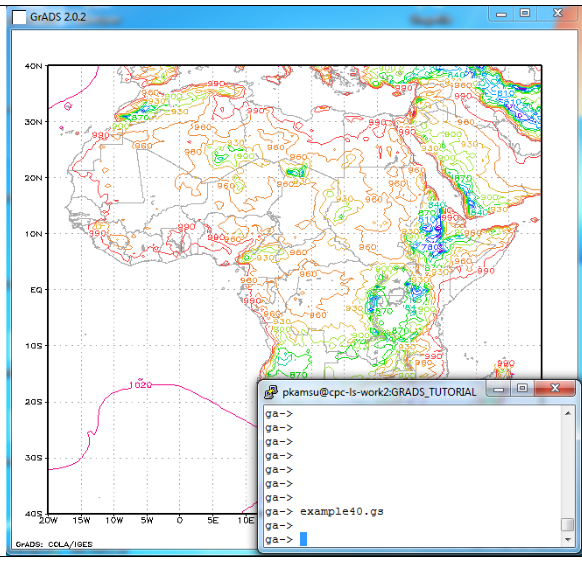
➢ **Assignment**: a claim generally used in general to set a variable = expression

➢ **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.

```
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 ' '
```

```
ga->
ga->
ga-> example41.gs

=======================================

 Hujambo !!!

 Hakuna Matata ...

 Learning GrADS is well Worth It

 Alavida ...  Kwaheri … Dehina yihunu

=======================================

ga->
```

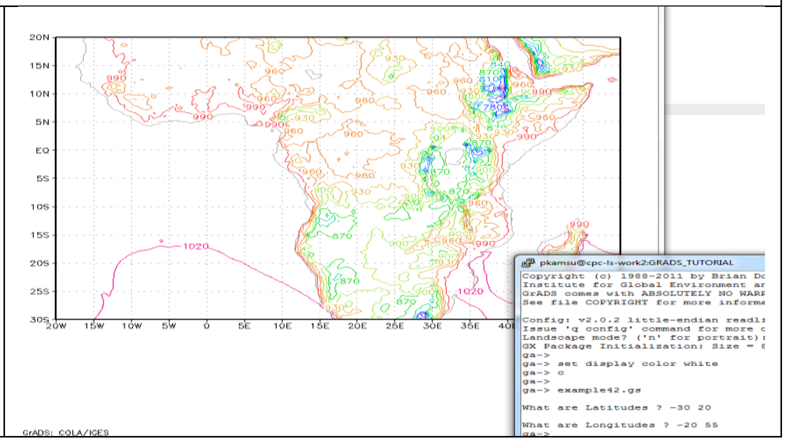
- **pull** allows to load the information provided by the user through keyboard, with the syntax:

  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.

```
'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:

  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:

  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**

| *logical OR*	**!=** *not equal*	**%** *concatenation*
& *logical AND*	**>** *greater than*	**+** *addition*
! *unary NOT*	**>=** *greater or equal than*	**-** *subtraction*
- *unary minus*	**<** *less than*	***** *multiplication*
= *equal*	**<=** *less or equal than*	**/** *division*

➢ Intrinsic functions

strlen(*string***)**	This function returns the length (number of characters) of *string*.
sublin (*string, n***)**	This function gets a single line from a string containing several lines. ***n** must be an integer.* The result is the ***nth*** line of ***string***. If the string has too few lines, the result is NULL.
subwrd (*string, n***)**	This function gets a single word from a string. ***n** must be an integer.* The result is the nth word of string. If the string is too short, the result is NULL.
substr (*string, start, length***)**	This function gets part of a string. ***start*** and ***length*** *must be an integer.* The sub-string of ***string*** starting at location ***start*** for length ***length*** will be returned. If the string is too short, the result will be short or NULL.
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.*: the first line is the return code, the 2nd line is the record read from the file. Return codes are: 0 - ok 1 - open error 2 - end of file 8 - file open for write 9 - I/O error *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 Return codes are: 0 - ok 1 - open error 8 - file open for read
close (*name*)	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. Return codes are: 0 - ok 1 - file not open

Complementary commands

query < ***option*****>** *or* **q <** ***option*****>**	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.
q define	Lists currently defined variables
q defval v1 i j	Returns the value of defined variable v1 at point i,j
q dims	Returns current dimension environment
q file n	Returns info on file number n. Uses the default file if n is not given.
q files	Lists open files
q fwrite	Returns status and characteristics of fwrite ouput file
q gxinfo	Returns graphics environment info
q gxout	Returns current gxout settings
q pos	Waits for mouse click, then returns position plus additional widget information
q shades	Lists colors and levels of shaded contours
q time	Returns info about time setting
q xy2w v1 v2	Converts XY coords to world cords
q xy2gr v1 v2	Converts XY coords to grid cords
q w2xy v1 v2	Converts world coords to XY cords
q w2gr v1 v2	Converts world coords to grid cords
q gr2w v1 v2	Converts grid coords to world cords
q gr2xy v1 v2	Converts grid coords to XY cords

6.3 Application examples and exercises

Example 44: Calculating climatology and plotting anomalies ...

```
*** 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'

*-----------------------------------------------------------------------
'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