1. Starting and Quitting GrADS

GrADS is started by entering the command: **grads**
Before initializing the graphics output environment, GrADS will prompt for landscape or portrait mode. Landscape is 11 x 8.5 inches (usually what you want). Portrait is 8.5 x 11 inches, primarily used for producing vertically oriented hardcopy output. The actual size of the window will not, of course, be 11 x 8.5 inches (or 8.5 x 11 inches), but instead will be whatever size you chose by using your workstation's window manager. But GrADS will treat the window as if it were one of the above sizes, so it is best to size the window with approximately the proper aspect ratio. This can be done using the window manager or from GrADS using the command:

set xsize x y

which resizes the window to x,y pixels.
After answering this prompt, a separate graphics output window will be opened (but not on PCs). You may move or resize this window at any time.
You will enter GrADS commands in the text window from where you started GrADS. Graphics output will appear in the graphics window in response to the commands you enter. You will thus need to make the text window the "active" window; the window that receives keyboard input.

Startup options

You may specify the following options as arguments to the **grads** command when GrADS is started:

- **-b** Run grads in batch mode. No graphics output window is opened.
- **-l** Run grads in landscape mode. The Portrait vs. Landscape question is not asked.
- **-p** Run grads in portrait mode.
- **-c** Execute the supplied command as the 1 st GrADS command after GrADS is started.

An example:

```
grads -c "run profile.gs"
```

These options may be used in combinations. For example:

```
grads -blc "run batch.gs"
```

Would run grads in batch mode, using landscape orientation (thus no questions are asked at startup); and execute the command:

batch.gs upon startup.

Leaving GrADS

To leave GrADS, enter the command:

```
quit
```

2. Basic Concept of Operation

When you have successfully installed and started GrADS, you'll be confronted with two windows -- a terminal window with a prompt, much like the infamous C:> in MS-DOS, and a resizable window (black background by default) where graphics are displayed. GrADS commands are entered in the terminal window and the response from GrADS is either graphics in the graphics window or text in the terminal window. The three fundamental GrADS commands:

open	open or make available to GrADS a data file with either gridded or station data
d	display a GrADS "expression" (e.g., a slice of data)
set	manipulate the "what" "where" and "how" of data display

The GrADS "expression," or what you want to look at, can be as simple as a variable in the data file that was opened, e.g., **d slp** or an arithmetic or GrADS function operation on the data, e.g., **d slp/100** or **d mag(u,v)** where mag is a GrADS intrinsic function.

The "where" of data display is called the "dimension environment" and defines which part, chunk or "hyperslab" of the 4-D geophysical space (lon,lat,level,time) is displayed. The dimension environment is manipulated through the set command and is controlled in either grid coordinates (x,y,z,t or indices) or world coordinates (lon, lat,lev, time).

The "what" and "how" of display is controlled by **set** commands and includes both graphics methods (e.g., contours, streamlines) and data (e.g., d to a file).

GrADS graphics can be written to a file (i.e., **enable print** *filename* and **print**) and then converted to postscript for printing and/or conversion to other image formats.

In addition, GrADS includes graphic primitives (e.g., lines and circles) and basic labelling through the **draw** command.

The **q** or **query** command is used to get information from GrADS such as which files are opened and even statistics.

3. Using GrADS Data Files

In GrADS, the raw binary data and the meta data (information about the binary data) are stored in separate files. The meta data file contains a complete description of the binary data as well as instructions for GrADS on where to find the data and how to read it. The binary data file is purely data with no space or time identifiers. The meta data file is the one you open in GrADS -- it is called the **data descriptor file**. The data descriptor file has a **.ctl** extension and is therefore also referred to as a **control file**.

The Data Descriptor File

The data descriptor file contains a complete description of the binary data as well as instructions for GrADS on where to find the data and how to read it. The descriptor file is an ASCII file that can be created easily with a text editor. The general contents of a gridded data descriptor file are as follows:

- Filename for the binary data
- Missing or undefined data value

- Mapping between grid coordinates and world coordinates
- Description of variables in the binary data set

The data descriptor file is free format, which means the components of each record (line of text) are blank delimited. Leading blanks at the beginning of each record are removed before parsing. Comment records must start with an asterisk (*). Individual records may not be more than 255 characters long. Here is an example of a basic data descriptor file:

```
DSET   ^gridded_data_sample.dat
TITLE Gridded Data Sample
UNDEF -9.99E33
XDEF 180 LINEAR 0.0   2.0
YDEF   90 LINEAR -90   2.0
ZDEF   10 LEVELS 1000 850 700 500 400 300 250 200 150 100
TDEF    4 LINEAR 0Z10apr1991 12hr
VARS       4
slp       0   99   sea level pressure
hgt      10   99   heights
temp     10   99   temperature
shum      6   99   specific humidity
ENDVARS
```

In this example, the binary data set is named gridded_data_sample.dat and is located in the same directory as the descriptor file. This is specified by the caret (^) in front of the data filename. The undefined or missing data value is -9.99e33, there are 180 grid points in the X direction, 90 grid points in the Y direction, 10 vertical levels, 4 time steps, and 4 variables. The variable "slp" is a surface variable -- it has no vertical levels, but is assigned a default vertical coordinate of Z=1. The variables "hgt" and "temp" have 10 vertical levels, and the variable "shum" has 6 vertical levels (the first six listed, 1000 to 300).

Structure of a Gridded Binary Data File

The binary data file is purely data with no space or time identifiers. The data descriptor specifies the data's grid dimensions, but it is up to the user to make sure that the binary

data have been written to file in the proper order so GrADS will interpret them correctly.

GrADS views gridded data sets as 5-dimensional arrays varying in longitude, latitude, vertical level, variable, and time. It is helpful to think of a gridded binary data file as a sequence of "building blocks", where each building block is a horizonal grid of data varying in the X and Y dimensions. The first dimension (X) always varies from west to east; the second dimension (Y) varies from south to north (by default). One horizontal grid represents a particular variable at a particular height and time.

Each horizontal grid in a GrADS binary data file must be the same size. If you have two variables with different horizontal grids, you must create two separate data sets.

The structure of a 3-D, 4-D, or 5-D data set is determined by the order in which the horizonal grids are written to file. The building blocks are stacked in a sequence according to dimension. The sequence goes in the following order starting from the fastest varying dimension to the slowest varying dimension: longitude (X), latitude (Y), vertical level (Z), variable (VAR), time (T).

For example, suppose you want to create a 4-D binary data set containing four variables. The horizonal grids would be written to the data set in the following order:

Time 1, Variable 1 , Each vertical level from bottom to top
Time 1, Variable 2 , Each vertical level from bottom to top
Time 1, Variable 3 , Each vertical level from bottom to top
Time 1, Variable 4 , Each vertical level from bottom to top

Time 2, Variable 1 , Each vertical level from bottom to top
Time 2, Variable 2 , Each vertical level from bottom to top
Time 2, Variable 3 , Each vertical level from bottom to top
Time 2, Variable 4 , Each vertical level from bottom to top

etc.

Binary Formats

GrADS can read binary data that are formatted with or without FORTRAN record length headers. Files containing record length headers are called "sequential" and those without embedded record length information are called "direct access" or "stream" files. Unless otherwise specified, GrADS will assume the data file does not contain the record length headers.

GrADS can also directly read GRIB formatted data -- one of GrADS most powerful and unique features! See the section on *Creating Data Descriptor Files for GRIB Data* for more information.

A third category of data formats that GrADS can read are "self-describing files" such as NetCDF or HDF-SDS. For more information, see the references pages for **sdfopen** and **xdfopen**.

Creating Data Files

The default format for GrADS gridded binary data files is "stream" or "direct access". If you want to read FORTRAN "sequential" unformatted binary data files, you must include the following additional record in the data descriptor file:

```
OPTIONS sequential
```

Following are three examples of how to create gridded binary data files with simple FORTRAN programs.

Suppose you have U and V wind components in 4-dimensions (X, Y, Z, and T) and you want to write them out in so they can be viewed in GrADS. The FORTRAN code might look something like this:

```
parameter (ni=144,nj=91,nk=8,nt=4)
dimension u(ni,nj,nk),v(ni,nj,nk),dum(ni,nj)
do n=1,nk
   call load(u,ni,nj,nk,n,dum)
   write(10) dum
end do
do n=1,nk
```

```
    call load(v,ni,nj,nk,n,dum)
    write(10) dum
end do

subroutine load(a,ni,nj,nk,n,dum)
dimension a(ni,nj,nk),dum(ni,nj)
do i=1,ni
    do j=1,nj
    dum(i,j)=a(i,j,n)
end do
end do
return
```

The data descriptor file would look something like:

```
DSET        ^model.dat
TITLE       Sample Model Data
UNDEF       0.10000E+ 16
XDEF        144 linear     0 2.5
YDEF         91 linear -90 2.0
ZDEF          8 levels 1000 900 800 700 500 300 100 50
TDEF          4 linear 00z01apr85 6hr
VARS         2
    u 8 99 U component
    v 8 99 V component
ENDVARS
```

This simple example write out one variable:

```
  REAL   Z(72,46,16)
   ....
   OPEN(8,FILE='grads.dat',FORM='UNFORMATTED',
 & ACCESS='DIRECT',RECL=72*46)
   ....
   IREC=1
   DO 10 I=1,16
      WRITE (8,REC=IREC) ((Z(J,K,I),J=1,72),K=1,46)
      IREC=IREC+ 1
10 CONTINUE
```

Another simple sample might be:

```
  REAL X(100)
   DO 10 I=1,100
      X(I)=I
10 CONTINUE
   OPEN (8,FILE='samp.dat',FORM='UNFORMATTED',ACCESS='DIRECT',
  &RECL=100)
   WRITE (8,REC=1) X
   STOP
   END
```

The associated descriptor file:

```
DSET         samp.dat
TITLE        Sample Data Set
UNDEF        -9.99E33
XDEF        100 LINEAR 1 1
YDEF        1 LINEAR 1 1
ZDEF         1 LINEAR 1 1
TDEF         1 LINEAR 1JAN2000 1DY
VARS         1
x   0   99   100 Data Points
```

ENDVARS

Once created, you can use this data set to experiment with GrADS data functions, such as:

```
display sin(x/50)
```

Components of a GrADS Data Descriptor File

DSET *data_filename*

This entry specifies the filename of the data file being described. If the data and the descriptor file are not in the same directory, then data_filename must include a full path. If a ^ character is placed in front of data_filename, then data_filename is assumed to be relative to the path of the descriptor file. If you are using the ^ character in the DSET entry, then the descriptor file and the data file may be moved to a new directory without changing any entries in the data descriptor file, provided their relative paths remain the same. For example:

If the data descriptor file is:

```
/data/wx/grads/sa.ctl
```

and the binary data file is:

```
/data/wx/grads/sa.dat
```

then the data file name in the data descriptor file can be:

```
DSET ^sa.dat
```

instead of:

```
DSET /data/wx/grads/sa.dat
```

If data_filename does not include a full path or a ^, then GrADS will only look for data files in the directory where you are running GrADS.

GrADS allows you use a single DSET entry to aggregate multiple data files and handle them as if they were one individual file. The individual data files must be identical in all dimensions except time, and the time range of each individual file must be indicated it

its filename. To accomplish this, the DSET entry has a substitution template instead of a filename.
See the section on Using Templates (http://grads.iges.org/grads/gadoc/templates.html) for a description of all the possible components of the template. Second, the OPTIONS entry must contain the template keyword.

CHSUB *t1 t2 string*

(GrADS version 1.9b4) This entry is used with a new option for templating data files that allows for any user-specified string substitution, instead of only date string substitution. This is useful when none of the standard template options match the time ranges in the files you wish to aggregate, or if the files are located on different disks. When you put the %ch template in your DSET entry, then you also need to put additional CHSUB entries in the descriptor file. The string will be substituted for %ch in the data file name for the time steps beginning with t1 and ending with t2.See the section on Using Templates (http://grads.iges.org/grads/gadoc/templates.html) for examples.

DTYPE *keyword*

The DTYPE entry specifies the type of data being described. There are four options: grib, hdfsds, netcdf, or station. If the data type is none of these, then the DTYPE entry is omitted completely from the descriptor file and GrADS will assume the data type is gridded binary.

bufr	(GrADS version 1.9) Data file is a BUFR station data file. This data type must be accompanied by the following special entries: XVAR, YVAR, TVAR, STID. Optional special entries are: ZVAR, TOFFVAR.
grib	Data file is an indexed GRIB file. This data type requires a secondary entry in the descriptor file: INDEX. The INDEX entry provides the filename (including the full path or a ^) for the grib index file. The index file is created by the gribmap utility. You must run gribmap and create the index file before you can display the grib data in GrADS.
hdfsds	(GrADS version 1.9) Data file is an HDF Scientific Data Set (SDS). Although HDF-SDS files are self-describing and may be read automatically using the sdfopen/xdfopen commands, this DTYPE gives you the option of overriding

	the file's own metadata and creating a descriptor file for some or all of the variables in the file. This **DTYPE** may also be used if the metadata in the HDF-SDS file is insufficient or is not coards-compliant. This data type requires a special entry in the units field of the variable declaration. The **undef** and **unpack** entries contain special options for this dtype.
netcdf	(GrADS version 1.9) Data file is NetCDF. Although NetCDF files are self-describing and may be read automatically using the **sdfopen/xdfopen** commands, this DTYPE gives you the option of overriding the file's own metadata and creating a descriptor file for some or all of the variables in the file. This DTYPE may also be used if the metadata in the NetCDF file is insufficient or is not coards-compliant. This data type requires a special entry in the units field of the variable declaration. The **undef** and **unpack** entries contain special options for this dtype.
station	Data file is in GrADS station data format. This data type requires a secondary entry in the descriptor file: **STNMAP**. The **STNMAP** entry provides the filename (including the full path or a ^) for the station data map file. The map file is created by the **stnmap** utility. You must run **stnmap** and create the map file before you can display the station data in GrADS.

INDEX *filename*

This entry specifies the name of the grib map file. It is required when using the **DTYPE** grib entry to read grib formatted data. The file is generated by running the external utility **gribmap**. Filenaming conventions are the same as those described for the **DSET** entry.

STNMAP *filename*

This entry specifies the name of the station map file. It is required when using the **DTYPE** station entry to read GrADS-formatted station data. The file is generated by running the external utility **stnmap**. Filenaming conventions are the same as those described for the **DSET** entry.

TITLE *string*

This entry gives brief description of the contents of the data set. String will be included in the output from a query command and it will appear in the directory listing if you are serving this data file with the GrADS-DODS Server (GDS), so it is helpful to put meaningful information in the title field. For GDS use, do not use double quotation marks (") in the title.

UNDEF *value <undef_attribute_name>*

This entry specifies the undefined or missing data value. **UNDEF** is a required entry even if there are no undefined data. GrADS operations and graphics routines will ignore data with this value from this data set.
(GrADS version 1.9b4) An optional second argument has been added for data sets of **DTYPE** netcdf or hdfsds -- it is the name of the attribute that contains the undefined value. This should be used when individual variables in the data file have different undefined values. After data I/O, the missing values in the grid are converted from the variable undef to the file-wide undef (the numerical value in the first argument of the UNDEF record). Then it appears to GrADS that all variables have the same undef value, even if they don't in the original data file. If the data require a transformation using the attributes named in the **UNPACK** entry, GrADS assumes the variable undef value corresponds to the data values as they appear in the file, i.e., before they are transformed using a scale factor and offset. Missing packed data values are thus assigned the file-wide undef value and are never unpacked. Attribute names are case sensitive, and it is assumed that the name is identical for all variables in the netcdf or hdfsds data file. If the name given does not match any attributes, or if no name is given, the file-wide undef value will be used.
Example: UNDEF 1e+33 _FillValue

UNPACK *scale_factor_attribute_name <add_offset_attribute_name>*

(GrADS version 1.9) This entry is used with **DTYPE** netcdf or hdfsds for data variables that are 'packed' -- i.e. non-float data that need to be converted to float by applying the following formula:

y = x * scale_factor + add_offset

Only the attribute name for the scale factor is required. If your netcdf or hdfsds file does not have an offset attribute, the 2nd argument may be omitted, and the offset will be assigned the default value of 0.0. Attribute names are case sensitive, and it is

assumed that the names are identical for all variables in the netcdf or hdfsds data file. If the names given do not match any attributes, the scale factor will be assigned a value of 1.0 and the offset will be assigned a value of 0.0. The transformation of packed data is done after the undef test has been applied.
Examples:
UNPACK scale_factor add_offset
UNPACK Slope Intercept

FILEHEADER *length*

This optional entry tells GrADS that your data file has a header record of length bytes that precedes the data. GrADS will skip past this header, then treat the remaineder of the file as though it were a normal GrADS binary file after that point. This optional descriptor file entry is only valid for GrADS gridded data sets.

THEADER *length*

This optional entry tells GrADS that the data file has a header record of length bytes preceding each time block of binary data. This optional descriptor file entry is only valid for GrADS gridded data sets. See the section on structure of a gridded binary data file (http://grads.iges.org/grads/gadoc/aboutgriddeddata.html#structure) for more information.

XYHEADER *length*

This optional entry tells GrADS that the data file has a header record of length bytes preceding each horizontal grid (XY block) of binary data. This optional descriptor file entry is only valid for GrADS gridded data sets. See the section on structure of a gridded binary data file for more information.

XVAR *x,y*

(GrADS version 1.9) This entry provides the x,y pair for the station's longitude. This entry is required for DTYPE bufr.

YVAR *x,y*

(GrADS version 1.9) This entry provides the x,y pair for the station's latitude. This entry is required for DTYPE bufr.

ZVAR *x,y*

(GrADS version 1.9) This entry provides the x,y pair for the station data's vertical coordinate (e.g., pressure). This is an optional entry for DTYPE bufr.

STID *x,y*

(GrADS version 1.9) This entry provides the x,y pair for the station ID. This entry is required for DTYPE bufr.

TVAR *yr x,y mo x,y dy x,y hr x,y mn x,y sc x,y*

(GrADS version 1.9) This entry provides the x,y pairs for all the base time coordinate variables. Each time unit (year=yr, month=mo, day=dy, hour=hr, minute=mn, second=sc) is presented as a 2-letter abbreviation followed by the x,y pair that goes with that time unit. The time for any individual station report is the base time plus the offset time (see TOFFVAR). All six base time units are not required to appear in the TVAR record, only those that are in the data file. This entry is required for DTYPE bufr.

TOFFVAR *yr x,y mo x,y dy x,y hr x,y mn x,y sc x,y*

(GrADS version 1.9) This entry provides the x,y pairs for all the offset time coordinate variables. The syntax is the same as TVAR. The time for any individual station report is the base time plus the offset time. All six offset time units are not required to appear in the TOFFVAR record, only those that are in the data file. This is an optional entry for DTYPE bufr.

OPTIONS *keyword*

This entry controls various aspects of the way GrADS interprets the raw data file. It replaces the old FORMAT record. The keyword argument may be one or more of the following:

yrev	Indicates that the Y dimension (latitude) in the data file has been written in the reverse order from what GrADS assumes. An important thing to remember is that GrADS still presents the view that the data goes from south to north. The YDEF statement does not change; it still describes the transformation from a grid space going from south to north. The reversal of the Y axis is done as the data is read from the data file.
zrev	Indicates that the Z dimension (pressure) in the data file has been written from top to bottom, rather than from bottom to top as GrADS assumes. The same considerations as noted above for yrev also apply.
template	Indicates that a template for multiple data files is in use. For more information, see the section on Using Templates. (http://grads.iges.org/grads/gadoc/templates.html)
sequential	Indicates that the file was written in sequential unformatted I/O. This keyword may be used with either station or gridded data. If your gridded data is written in sequential format, then each record must be an X-Y varying grid. If you have only one X and one Y dimension in your file, then each record in the file will be one element long (it may not be a good idea to write the file this way).
365_day_calendar	Indicates the data file was created with perpetual 365-day years, with no leap years. This is used for some types of model ouput.
byteswapped	Indicates the binary data file is in reverse byte order from the normal byte order of your machine. Putting this keyword in the OPTIONS record of the descriptor file tells GrADS to swap the byte order as the data is being read. May be used with gridded or station data.

The best way to ensure hardware independence for gridded data is to specify the data's source platform. This facilitates moving data files and their descriptor files between machines; the data may be used on any type of hardware without having to worry about byte ordering. The following three OPTIONS keywords are used to describe the byte ordering of a gridded or station data file:

big_endian	Indicates the data file contains 32-bit IEEE floats created on a big endian platform (e.g., sun, sgi)

little_endian	Indicates the data file contains 32-bit IEEE floats created on a little endian platform (e.g., iX86, and dec)
cray_32bit_ieee	Indicates the data file contains 32-bit IEEE floats created on a cray.

PDEF

PDEF is so powerful it has its own documentation page.
(http://grads.iges.org/grads/gadoc/pdef.html)

XDEF *xnum mapping <additional arguments>*

This entry defines the grid point values for the X dimension, or longitude. The first argument, xnum, specifies the number of grid points in the X direction. xnum must be an integer >= 1. mapping defines the method by which longitudes are assigned to X grid points. There are two options for mapping:

LINEAR Linear mapping
LEVELS Longitudes specified individually

The LINEAR mapping method requires two additional arguments: start and increment. start is a floating point value that indicates the longitude at grid point X=1. Negative values indicate western longitudes. increment is the spacing between grid point values, given as a positive floating point value.

The LEVELS mapping method requires one additional argument, value-list, which explicitly specifies the longitude value for each grid point. value-list should contain xnum floating point values. It may continue into the next record in the descriptor file, but note that records may not have more than 255 characters. There must be at least 2 levels in value-list; otherwise use the LINEAR method.

Here are some examples:

XDEF 144 LINEAR 0.0 2.5
XDEF 72 LINEAR 0.0 5.0
XDEF 12 LEVELS 0 30 60 90 120 150 180 210 240 270 300 330
XDEF 12 LEVELS 15 45 75 105 135 165 195 225 255 285 315 345

YDEF *ynum mapping <additional arguments>*

This entry defines the grid point values for the Y dimension, or latitude. The first argument, ynum, specifies the number of grid points in the Y direction. ynum must be an integer >= 1. mapping defines the method by which latitudes are assigned to Y grid points. There are several options for mapping:

LINEAR	Linear mapping
LEVELS	Latitudes specified individually
GAUST62	Gaussian T62 latitudes
GAUSR15	Gaussian R15 latitudes
GAUSR20	Gaussian R20 latitudes
GAUSR30	Gaussian R30 latitudes
GAUSR40	Gaussian R40 latitudes

The LINEAR mapping method requires two additional arguments: start and increment. start is a floating point value that indicates the latitude at grid point Y=1. Negative values indicate southern latitides. increment is the spacing between grid point values in the Y direction. It is assumed that the Y dimension values go from south to north, so increment is always positive.

The LEVELS mapping method requires one additional argument, value-list, which explicitly specifies the latitude for each grid point, from south to north. value-list should contain ynum floating point values. It may continue into the next record in the descriptor file, but note that records may not have more than 255 characters. There must be at least 2 levels in value-list; otherwise use the LINEAR method.

The Gaussian mapping methods require one additional argument: start. This argument indicates the first gaussian grid number. If the data span all latitudes, start would be 1, indicating the southernmost gaussian grid latitude.

Here are some examples:

```
YDEF 73 LINEAR -90 2.5
YDEF 180 LINEAR -90 1.0
```

YDEF 18 LEVELS -85 -75 -65 -55 -45 -35 -25 -15 -5 5 15 25 35 45 55 65
75 85
YDEF 94 GAUST62 1
YDEF 20 GAUSR40 15

The NCEP/NCAR Reanalysis surface variables are on the GAUST62 grid.
The final example shows that there are 20 Y dimension values which start at Gaussian Latitude 15 (64.10 south) on the Gaussian R40 grid

ZDEF *znum mapping <additional arguments>*

This entry defines the grid point values for the Z dimension. The first argument, znum, specifies the number of pressure levels. znum must be an integer >= 1. mapping defines the method by which longitudes are assigned to Z grid points. There are two options for mapping:

LINEAR Linear mapping
LEVELS Pressure levels specified individually

The LINEAR mapping method requires two additional arguments: start and increment. start is a floating point value that indicates the longitude at grid point Z=1. increment is the spacing between grid point values in the Z direction, or from lower to higher. increment must be non-zero, but may be a negative value.

The LEVELS mapping method requires one additional argument, value-list, which explicitly specifies the pressure level for each grid point in ascending order. value-list should contain znum floating point values. It may continue into the next record in the descriptor file, but note that records may not have more than 255 characters.

Here are some examples:

ZDEF 10 LINEAR 1000 -100
ZDEF 7 LEVELS 1000 850 700 500 300 200 100
ZDEF 17 LEVELS 1000 925 850 700 600 500 400 300 250 200 150 100 70 50

TDEF *tnum LINEAR start increment*

This entry defines the grid point values for the T dimension. The first argument, tnum, specifies the number of time steps. tnum must be an integer >= 1. The method by which times are assigned to T grid points is always LINEAR.

start indicates the initial time value at grid point T=1. start must be specified in the GrADS absolute date/time format:

hh:mmZddmmmyyyy

where:

hh = hour (two digit integer)
mm = minute (two digit integer)
dd = day (one or two digit integer)
mmm = 3-character month
yyyy = year (may be a two or four digit integer; 2 digits implies a year between 1950 and 2049)

If not specified, hh defaults to 00, mm defaults to 00, and dd defaults to 1. The month and year must be specified. No intervening blanks are allowed in the GrADS absolute date/time format.

increment is the spacing between grid point values in the T direction. *increment* must be specified in the GrADS absolute time increment format:

vvkk

where:

vv = an integer number, 1 or 2 digits
kk = mn (minute)
hr (hour)
dy (day)
mo (month)
yr (year)

Here are some examples:

TDEF 60 LINEAR 00Z31dec1999 1mn
TDEF 73 LINEAR 3jan1989 5dy
TDEF 730 LINEAR 00z1jan1990 12hr
TDEF 12 LINEAR 1jan2000 1mo
TDEF 365 LINEAR 12Z1jan1959 1dy
TDEF 40 LINEAR 1jan1950 1yr

VECTORPAIRS *U-component,V-component*

(GrADS version 1.9b4) This entry is for explicity identifying vector component pairs. This is only necessary if the data are on a native projection other than lat/lon (i.e. you are using PDEF) and if the winds have to be rotated from a grid-relative sense to an Earth-relative sense. (GrADS has to retrieve both the u and v component in order to do the rotation calculation.)

Using this entry replaces the old technique of putting 33 (for U) or 34 (for V) in the first element of the units field in the variable declaration. The U-component and V-component arguments should be variable names that appear in the VARS list. They are separated by a comma, with no spaces. More than one pair of components may be listed; in this case, the pairs should be separated by a space. For example:

VECTORPAIRS u,v u10,v10 uflx,vflx

VARS varnum
variable_record_1
variable_record_2
...
variable_record_varnum
ENDVARS

This ensemble of entries describes all the variables contained in the data set. varnum indicates the number of variables in the data set and is therefore also equal to the number of variable records that are listed between the VARS and ENDVARS entries.

ENDVARS must be the final line of the Grads data descriptor file. Any blank lines after the ENDVARS statement may cause open to fail!

The format of the variable records is as follows:

varname levs units description

The syntax of varname and units is different depending on what kind of data format (DTYPE) you are describing. Details provided below:

varname	This is a 1-15 character "name" or abbreviation for the data variable. varname may contain alphabetic and numeric characters but it must start with an alphabetic character (a-z).
Varname (DTYPE netcdf or hdfsds)	(GrADS version 1.9) For DTYPE netcdf or hdfsds, varname may have a different syntax: SDF_varname=>grads_varname SDF_varname is the name the data variable was given when the SDF file was originally created. For NetCDF files, this name appears in the output from ncdump. It is important that SDF_varname exactly matches the variable name in the data file. SDF_varname may contain uppercase letters and non-alpha-numeric characters. The classic varname syntax (i.e., when "SDF_varname =>" is omitted) may be used if SDF_varname meets the criteria for GrADS variable names: it must be less than 16 characters, start with an alphabetic character, and cannot contain any upper case letters or non-alpha-numeric characters.
levs	This is an integer that specifies the number of vertical levels the variable contains. levs may not exceed znum as specified in the ZDEF statement. If levs is 0, the variable does not correspond to any vertical level. Surface variables (e.g. sea level pressure) have a levs value of 0. For DTYPE station or bufr, surface variables have a levs value of 0 and upper air variables have a levs value of 1. (Exception to this rule for bufr data: replicated surface variables are given a levs value of 2).
description	This is text description or long name for the variable, max 40 characters.

The units component of the variable record is used for data with **DTYPE** bufr, grib, netcdf, or hdfsds. It is also used for non-standard binary data files that require special "unpacking" instructions, and special cases of pre-projected wind components. If the data you are describing does not fall into any of these categories, put a value of 99 in the *units* field.

units	For flat binary files containing 4-byte floating-point data that are not pre-projected, this field is ignored but must be included. Put in a value of 99.
units (DTYPE bufr)	(GrADS version 1.9) For **DTYPE** bufr files, this field contains the x,y pair for the named variable.
units (DTYPE grib)	For **DTYPE** grib, the units field specifies the GRIB parameters of the variable. This information is used by the **gribmap** utility for mapping the variables listed in the descriptor file to the data records in the GRIB files. This parameter may contain up to four comma-delimited numbers: VV,LTYPE,LEVEL,RI where, VV = The GRIB parameter number (Required) LTYPE = The level type indicator (Required) LEVEL = The value of the LTYPE (Optional) RI = The "range indicator" (for certain level types) (Optional) The external utilities **gribscan** and **wgrib** are quite useful in determining what the values for the units field should be for a GRIB data file. Examples: u 39 33,100 U Winds [m/s] t 39 11,100 Temperature [K] ts 0 11,1 Surface Temperature [K] tb 0 11,116,60,30 Temperature, 30-60mb above surface [K] dpt 0 17,100,1000 Dew Point Temperature at 1000 mb [K]
units	(GrADS version 1.9) For **DTYPE** netcdf or hdfsds, the units field is a

(DTYPE netcdf or hdfsds)	comma-delimited list of the varying dimensions of the variable. Dimensions expressed as x, y, z, or t correspond to the four axes defined by XDEF, YDEF, ZDEF and TDEF. For example, a surface variable such as sea level pressure might look like this: presSFC=>psfc 0 y,x Surface Pressure A time-varying atmospheric variable such as geopotential height might look like this: Height=>hght 17 t,z,y,x Geopotential Height (m) The order of the dimensions listed in the units field does matter. They must describe the shape of the variable as it was written to the SDF data file. For NetCDf files, this information appears in the output from ncdump next to the variable name. If your data file contains a variable that also varies in a non-world-coordinate dimension (e.g. histogram interval, spectral band, ensemble number) then you can put a non-negative integer in the list of varying dimensions that will become the array index of the extra dimension. For example: VAR=>hist0 0 0,y,x First historgram interval for VAR VAR=>hist1 0 1,y,x Second historgram interval for VAR VAR=>hist2 0 2,y,x Third histogram interval for VAR Another option in this example would be to fill the unused Z axis with the histogram intervals: zdef 3 linear 1 1 ... VAR=>hist 3 z,y,x VAR Histogram In this case, it would appear to GrADS that variable 'hist' varies in Z, but the user would have to remember that the Z levels correspond to histogram intervals. The latter technique makes it easier to slice through the data, but is not the most accurate representation. And if you don't

<table>
<tr><td></td><td>have an unsued world-coordinate axis available, then you still have a way to access your data.</td></tr>
<tr><td>units
(non-
standard
binary)</td><td>For non-standard binary files, the units field is used to instruct GrADS how to read binary files that do not conform to the default structure or do not contain 4-byte float data. GrADS assumes the data were written in the following order (starting from the fastest varying dimension to the slowest): longitude (X), latitude (Y), vertical level (Z), variable (VAR), time (T). If your binary data set was created or "packed" according to a different dimension sequence, then you can use the units field to tell GrADS exactly how to unpack the data.

For these non-standard binary files, the units field is a series of one or more comma-delimited numbers, the first of which is always -1. The syntax is as follows:

-1, structure <,arg>

There are four options for structure, outlined below. Some of these options have additional attributes which are specified with arg.
<table>
<tr><td>-1,10,arg</td><td>This option indicates that "VAR" and "Z" have been transposed in the dimension sequence. The order is: longitude (X), latitude (Y), variable (VAR), vertical level (Z), time(T). Thus, all variables are written out one level at a time.

This feature was designed to be used with NASA GCM data in the "phoenix" format. The upper air prognostic variables were transposed, but the diagnostic variables were not. Thus an arg of 1 means the variable has been var-z transposed, and an arg of 2 means the variable has not.</td></tr>
<tr><td>-1,20,arg</td><td>This option indicates that "VAR" and "T" have been transposed in the dimension sequence. The order is: longitude (X), latitude (Y), vertical level (Z), time(T), variable (VAR). Thus, all times for one variable are written out in order followed by all times for the next variable, etc.</td></tr>
</table>
</td></tr>
</table>

		If your data set is actually a collection of separate files that are aggregated by using a template, then you must use arg to tell GrADS how many time steps are contained in each individual file. For example, here are the relevant records from a descriptor file for 10 years of monthly wind and temperature data packaged in 10 separate files (one for each year) with "VAR" and "T" dimensions transposed: DSET ^monthlydata_%y4.dat OPTIONS template TDEF 120 linear jan79 1mo VARS 3 u 17 -1,20,12 U-Wind Component v 17 -1,20,12 V-Wind Component t 17 -1,20,12 Temperature ENDVARS
	-1,30	This option handles the cruel and unusual case where X and Y dimensions are transposed and the horizontal grids are (lat,lon) as opposed to (lon,lat) data. This option causes GrADS to work very inefficiently. However, it is useful for initial inspection and debugging.
	-1,40,arg	This option handles non-float data. Data are converted to floats internally after they are read from the binary file. The dimension sequence is assumed to be the default. The secondary arg tells GrADS what type of data values are in the binary file: units = -1,40,1 = 1-byte unsigned chars (0-255) units = -1,40,2 = 2-byte unsigned integers units = -1,40,2,-1 = 2-byte signed integers units = -1,40,4 = 4-byte integers
units (pre-projected	For pre-projected vector component data that require the use of **PDEF** and **rotation**, GrADS has to retrieve both the u and v component in order to do the rotation calculation. The new (and recommended) method for	

wind components)	matching vector components is to use the **VECTORPAIRS** descriptor file entry. The old technique (for versions older than 1.9b4) is to use the units field of the variable record. The u-component variable must have a units value of 33, and the v-component variable must have a units value of 34. (This is the GRIB convention). If there are more than one u/v pairs, secondary units values are used.

@ *varname attribute_type attribute_name attribute_value*

(GrADS version 1.9b4) To supplement the metadata in your descriptor file, use attribute comments. The first two characters of the attribute comment must be "@" followed by a space -- this distinguishes it from an ordinary comment (see below). Attribute comments may appear anywhere in the descriptor file, and they will be ignored if used with older versions of GrADS.

All file attributes may be retrieved with the **'query attr'** command.

varname may be set to "global" to describe general attributes that are valid for the entire data set. Set *varname* to "lon", "lat", "lev", or "time" to describe attributes of the four coordinate axes; otherwise, use one of the variable names listed in the variable declarations. If a variable name is aliased, use the grads_varname instead of the native SDF_varname.

attribute_type should be one of the following case-sensitive types: String, Byte, Int16, UInt16, Int32, UInt32, Float32, Float64.

attribute_name may be any single word or string with no spaces (e.g.: "units", "minimum_value")

attribute_value can be any string as long as the length of the entire entry does not exceed 512 characters.

For example:

@ precip String units mm/day
@ global String documentation http://put.your.documentation.url.here

* *comment*

You may put comments in your descriptor file by beginning the entry with * . Use @ for formatted attribute comments (see above).

GrADS reference card version 1.7

(GrADS Version 1.7 beta 7) compiled by *Karin Meier-Fleischer*, DKRZ (beratung@dkrz.de

GrADS program executables

grads	link to one of the following executables
gradsnc	grads with netCDF enabled
gradshdf	grads with HDF enabled
gradsc	grads "classic", without netCDF/HDF/Athena GUI, etc.

Command line options

Program: grads [-lbpC] [-c 'command']		
	-c 'command'	execute 'command' when starting GrADS
	-b	run grads in batch mode. No graphics output window is opened.
	-l	run grads in landscape mode. The orientation question is not asked.
	-p	run grads in portrait mode. The orientation question is not asked.
	-C	enable automatic setting of century for years < 100

General settings

help	gives a summary list of operations
set grads on | off	enable/disable display of the GrADS logo

set display <option <color>>	
	sets the mode of the display. options are:
grey|greyscale	sets the mode to greyscale.
color <black|white>	shading and contouring is done with rainbow colors. Default.

set frame <option>	
	control the frame on a plot. options are:
on	plots a rectangular frame around clipped region
off	plots no frame
circle	plots a rectangular frame for lat-lon projections, plots a circular frame for a polar plot at the outermost latitude.Whole hemisphere plots only.

set background ic	set background color to color or color index ic
display expression d expression	display data via the graphics output window; the simplest expression is a variable abbreviation
open filename	open descriptor file
sdfopen file.nc <template #timesteps>	
	opens a netCDF or HDF-SDS format file that conforms to the COARDS conventions. The optional arguments are for string a time-series of files together as one GrADS data object.
xdfopen file	opens a non-COARDS-conformant netCDF or HDF-SDS file via a data descriptor file similar to those used with the 'open' command.
close file#	close the last descriptor file opened.
set dfile number	change to descriptor file number for current file
define var=expr var=expr	create new variable, which then can be used in subsequent expressions
undefine var	free the resources used by the defined variable

modify varname <time type>	
	define variable, which is climatological. varname is the defined grid. Time types are:
seasonal	monthly or multi-monthly means
diurnal	over some time period less than a day

query <option> q <option>	query options are:
config	list GrADS configuration information
files	lists open files
file n	gives info on particular file
define	lists currently defined variables
dims	gives current dimension environment
gxinfo	gives graphics environment info
shades	gives colors and levels of shaded contours
pos	waits for mouse click, returns the position
time	gives info about time settings
fwrite	print name of fwrite ouput file
string s	gives the width of string s
defval v1 i j	gives the value of a defined variable v1 at point i,j
udft	list the user defined function table
lats	state of the GrADS-LATS interface
xy2w v1 v2	XY coords to world coords
xy2gr v1 v2	XY coords to grid coords
w2xy v1 v2	world coords to XY coords
w2gr v1 v2	world coords to grid coords
gr2w v1 v2	grid coords to world coords
gr2xy v1 v2	grid coords to XY coords
ll2xy lon lat	LON/LAT coords to XY coords
pp2xy ppx ppy	page coords to XY coords

set imprun script	automatically executes script before every display command
run file-name <params>	
	load and run a GrADS script (with parameters)
exec fname <arg0,..,arg9>	
	executes a sequence of GrADS commands from file fname. If a clear command is encountered, GrADS waits until enter is pressed before clearing and continuing with command processing

clear <option> c <option>	c <option> clear the display; option are:
events	flush event buffer
graphics	clear graphic, not widgets
hbuff	clear display buffer, when in double buffer mode

reset <option>	initializes GrADS to its initial state with following exceptions: 1) No files are closed. 2) No defined objects are released. 3) The 'set display' settings are not modified. Options are: events; graphics; hbuff; norset
reinit	same as reset, and in addition closes all open files and releases all defined objects
quit	quit - to leave GrADS
!shell-command	runs a shell command on GrADS command line. The output will not be returned to the script, only displayed.

Dimension environments

set lon	**val1 <val2>**	sets longitude to vary from val1 to val2
set lat	**val1 <val2>**	sets latitude to vary from val1 to val2
set lev	**val**	sets the level to val - fixed dimension
set t	**val1 <val2>**	sets time to the "val" time in the data set
set x	**val1 <val2>**	set x values or fix it to one value
set y	**val1 <val2>**	set y values or fix it to one value
set z	**val1 <val2>**	set z values or fix it to one value

Page control

set vpage	**off**	real page is equal to "virtual page"; default state
set vpage	**xmn xmx ymn ymx**	defining one "virtual" page
set parea	**xmn xmx ymn ymx**	control the area within the virutal page

Graphic types

set gxout graphic-type	where graphic-type could be:
bar	Bar chart
barb	Plot wind barb at station
contour	Contour plot
errbar	Error bar
fgrid	specific value grid fill plot
findstn	Find closest station to x,y point
fwrite	Write data to file instead of displaying
grfill	Filled grid boxes
grid	Grid boxes with values
line	Line graph
linefill	Color fill between two lines
model	Plot station model
scatter	Scatter graph plot
shaded	Shaded contour plot
stat	Display information about data
stream	Streamline plot
tserwx	Plot time series of weather symbols at a point (1-D station)
tserbarb	Plot time series of wind barbs at a point (1-D)
value	Plot station values
vector	Vector wind arrows
wxsym	Plot weather symbols at station

Default colors, line styles and marker types

colors used by many settings (i.e. ccolor, line, string button, clopts, lfcols, ...):			
0	black	1	white
2	red	3	green
4	blue	5	cyan
6	magenta	7	yellow
8	orange	9	purple
10	yellow/green	11	med.blue
12	dark yellow	13	aqua
14	dark purple	15	grey

line **styles** used by many settings (i.e. cstyle, line, mpt, map, grid, ...):			
0	none	1	solid
2	long dash	3	short dash
4	long short dash	5	dots
6	dot dash	7	dot dot dash

marker types used by many settings (i.e. cmark, mark, ...):			
0	none	1	cross
2	open circle	3	closed circle
4	open square	5	closed square
6	X	7	diamond
8	triangle	10	open circle with vertical line
9	none	11	open oval

Graphics options

set clip xlo xhi ylo yhi	clipping area for drawing graphics primitives
set ccolor index	sets the contour color to index, see Default colors and line styles. You can also issue: rainbow - rainbow color sequence revrain - reversed rainbow color sequence
set cstyle style	sets the contour or line style, see Default colors and line styles. (gxout = contour, only style 1,2,3 and 5 available).
set cmark marker	sets line marker, see Default colors and line styles .
set cterp on | off	turns spline smoothing on or off
set clab on | off | forced | string | auto	
	controls contour labeling
set clopts col <thick <size>>	contour line options
set clskip val	skip val contour lines when labelling
set cthick thckns	sets the line thickness for the contours [1-10]
set csmooth on | off | linear	interpolate to a finer grid using cubic or linear interpolation
set cint value	sets the contour interval to the specified "value"
set cmax value	contours not drawn above this value
set cmin value	contours not drawn below this value
set clevs lev1 lev2 ...	sets specified contour levels
set ccols col1 col2 ...	sets specified color for clev levels
set line col <style> <thick>	sets current line attributes. thickness range 1 - 6 (see Default colors and line styles).
set lfcols col1 col2	set color below and above lines (gxout linefill)
set black off | val1 val2	contours not drawn within this interval
set rbcols c1 c2 <c3 ... cn>	specifies a new 'rainbow' color sequence
set rbcols <auto>	built in rainbow sequence is used
set rbrange low high	range of values used to determine which values acquire which rainbow color

set grid on | off | <style> <color> | horizontal | vertical	
	draw grid lines using the specified options or not
set bargap val	sets the gap between bars in percent
set barbase value | bottom | top	bar rises from or falls from value
set baropts filled | outline	bar outlined or filled; default: filled
set dignum number	number of digits after the decimal place
set digsize size	size (in inches, or plotter units) of the numbers
set arrlab on|off	set arrow labeling on or off
set arrscl size <magnitude>	specifies arrow length scaling
set arrowhead size	specifies arrow head size
set fgvals v1 c1 <v2 c2>...	fgrid output type treats the grid values as rounded integers, and will shade a specified integer valued grid with the specified color.
set zlog on | off	sets log scaling of the Z dimension on or off
set strmden value	specifies the streamline density, where value is from 1 to 10. Default: 5
set stnopts <dig3> <nodig3>	plot the number in the slp location as a three digit number with only the last three digits of the whole number plotted
set mdlopts noblank | blank | dig3 | nodig3	
	plot the number of the model data as a three digit
set stid on | off	controls wether the station id is displayed next to the values or not
set wxcols c1 c2 c3 c4 c5 c6	set colors for weather symbols c1 - c6

Axis labeling/Annotation/labeling

set xaxis start end <incr>	specifies the axis is to be labeled
set yaxis start end <incr>	specifies the axis is to be labeled
set xlevs lab1 lab2 ...	specifies the label levels to plot for the X axis
set ylevs lab1 lab2 ...	specifies the label levels to plot for the Y axis
set xlint interval	specifies the label interval of the X axis
set ylint interval	specifies the label interval of the Y axis
set xyrev on	reverses the axes on a plot
set xflip on	flips the order of the horizontal axis
set yflip on	flips the order of the vertical axis
set xlab on | off | auto | string	controls and/or draws X axis label
set ylab on | off | auto | string	controls and/or draws Y axis label
set xlabs lab1 | lab2 |	abel the x axis with lab1, lab2, lab3, ...
set ylabs lab1 | lab2 |	label the y axis with lab1, lab2, lab3, ...
draw xlab string	draw x axis label
draw ylab string	draw y axis label
set xlopts col <thick <size>>	controls X axis
set ylopts col <thick <size>>	controls Y axis
set xlpos offset side	controls position of x axis labels. Where offset - in inches; side - b or t (bottom or top)
set ylpos offset side	controls position of y axis labels. Where offset - in inches; side - r or l (right or left)
set zlog on | off | swap | undefine	
	sets log scaling of the Z axis
set annot col <thick>	sets color and line thickness for the above 3 draw commands
set vrange vlo vhi	Set range for plotting 1-D or scatter plots; range of the variable values for y-axis scaling
set vrange2 vlo vhi	Set range for plotting 1-D or scatter plots; range of the variable values for x-axis scaling
set missconn on | off	lines will be connected across missing data
draw title string	draw title at top of graph

Map projections/drawing

set mproj proj	sets current map projection. Keywords are:
latlon	Lat/lon projection with aspect ratio maintained. Default.
scaled	latlon projection where aspect ratio is not maintained.The plot fills the plotting area.
nps	north polar stereographic
sps	south polar stereographic
robinson	Robinson projection
orthogr	Orthographic projection
mollweide	Mollweide projection
lambert	Lambert conformal conic projection
off	same as scaled, but no map is drawn and lables are not interpreted as lat/lon labels

set mpt type off | <<col> <style> <thick>>	
	command to control map background behavior. type is the map type; it can be a number from 0 to 255, or it can be an asterick (*) to indicate this command applies to all the type values. The color can be set to -1, which indicates to GrADS to use the set map settings for this map type, rather than the settings specified by the set mpt command.
set mpvals off | lnmn lnmx ltmn ltmx	
	sets reference longitudes and latitudes for polar stereogr. plots
set mpdset lowres | mres | hires | nmap	
	mres and hires have state and country outlines. nmap covers only North America. Default:lowres.
set map auto | color <style <thick>>	
	draws the map background using the requested line attributes or auto mode
set mpdraw on | off	
	if off, does not draw the map background
set grid on | off <style <col>> | horizontal | vertical	
	draw or do not draw lat/lon lines on polar plots using the specified color and linestyle
set poli on | off	selects whether you want political boundries drawn for the mres or hires map data sets. Default is on

Graphic primitives

draw line x1 y1 x2 y2	draws a line from x1, y1 to x2, y2 using current line drawing attributes
draw rec xlo ylo xhi yhi	draws an unfilled rectangle
draw recf xlo ylo xhi yhi	draws a filled rectangle
draw mark marktype x y size	
	draws a marker. Marker types (see Default colors and line styles).
draw polyf x1 y1 x2 y2 ... xn yn	
	draw a filled polyline, where xn=x1 and yn=y1
draw wxsym symbol x y size <color <thickness>>	
	Draws the specified wx symbol at the specified location

String primitives

set string col <justification> <thick> <rotation>	
	sets string drawing attributes. Justification: l - left; c - center; r - right; tl - top left; tc - center top; tr - right top; bl - bottom left; tc - center bottom; tr - right bott. Roation: 90 - counterclockwise, -90 - clockwise..
set strsiz width <height>	sets the string character size
draw string x y string	draws the character string at the x,y position
draw title string	draw a title 'string' on top of the graph

Color settings

set rgb num red green blue	defines new colors within GrADS, and assigns them to a new color number.color-number num must be a value between 16 and 99 (0 to 15 are predefined)

Font settings

set font number	change to font number [0-5]

Widgets

set button 1 bcol1 bcol2 bcol3 0 fcol1 fcol2 fcol3 thickness	
	set button colors. 1 - "on" state; 0 - "off" state
draw button number x y width height string	
	draws a button on position x,y with the attributes
redraw button number 0 | 1	
	redraws button number; 1 - "on"; 0 - "off"
set rband wn mode x1 y1 x2 y2	
	rubber banding. wn = widget #; mode = box or line x1, y1 = lowest point in x/y page units x2, y2 = highest point in x/y page units
draw dropmenu number x y width height text	
	display a dropmenu similar to 'draw button' command widget number (0 to 64); x and y are the center location for the 'base' of the dropmenu; width and height are the size of the 'base' of the dropmenu.

Double buffering

set dbuff on|off	sets double buffer mode on or off
swap	swaps buffers, when double buffer mode is on

Animation

set looping on|off	control animation; set animation on or off
set loopdim x|y|z|t	animate through x,y,z or t; default: t
set loopincr incr	set looping increment

Hardcopy output

enable print fname	enables the print command to the given file fname
print	copy the contents of current display into a file in a metacode format
disable print	close print output file
outxwd file	output the graphicw window to a file in the X windows dump format
wi file.format	output to a file with format (using ImageMagick), e.g. wi test.gif

Create/Write a grid file

set fwrite fname	output grid fname; if not set, fname=grads.fwrite
set gxout fwrite	enables grid file output
disable fwrite	close output grid file

Mathematical Functions

abs(expr)	absolute value of result of expr. Operates on gridded and station data
acos(expr)	applies the $\cos^{-1}$ function to the result of expr
asin(expr)	applies the $\sin^{-1}$ function to the result of expr
atan2(expr1,expr2)	applies the $\tan^{-1}$ function to the result of the two expr, using $\tan\theta = y/x$
cos(expr)	takes the cosine of the expr
exp(expr)	performs the ex operation, where expr is x. gridded and station data
gint(expr)	general integral, same as ave except do not devide by the total area
log(expr)	takes the natural logarithm of expr
log10(expr)	takes the logarithm base 10 of the expr
pow(expr1,expr2)	raises the values of expr1 to the power of expr2
sin(expr)	takes the sine of the provided expr (in radians)
sqrt(expr)	takes the square root of the result of the expr
tan(expr)	takes the trigonometric tangent of the expr

Averaging Functions

ave(expr,dexpr1,dexpr2<,tinc<,flags>>)	
	generalized averaging function. expr is averaged through the dimension range specified by dim1 and dim2
aave(expr,xdim1,xdim2,ydim1,ydim2)	
	does area average. xdim1 and xdim2 must be for lon or x, ydim1 and ydim2 must be for lat or y (e.g. aave(t,lon=0,lon=180,lat=0,lat=90))
mean(expr,dexpr1,dexpr2<,tinc<,flags>>)	
	same as ave, except that area weighting is disabled
amean(expr,xdim1,xdim2,ydim1,ydim2)	
	same as aave, except that area weighting is disabled
vint(psexpr,expr,top)	performs a mass-weighted vertical integral in mb pressure coordinates, where: exprexpression for quantity to be integrated psexpr expression yielding the surface pressure, in mb,which will be used to bound the integration on the bottom topconstant, giving the bounding top pressure, in mb. This cannot be provided as an expression

Grid Functions

const(expr,const<,flag>)	function allows you to set various parts of a grid to a constant
maskout(expr,mask)	whenever the mask values are less than zero, the values in expr are set to the missing data value
skip(expr,skipx,skipy)	sets alternating values of the expr to the missing data value.This function is used while displaying wind arrows or barbs to thin the number of arrows or barbs

Filtering Functions

smth9(expr)	performs a 9 point smoothing to the gridded result of expr

Finite Difference Functions

cdiff(expr,dim)	performs a centered difference operation on expr in the direction specified by dim

Meteorological Functions

tvrh2q(tvexpr,rhexpr)	given virtual temperature and relative humidity, tvrh2q returns specific humidity, q, in g/g
tvrh2t(tvexpr,rhexpr)	given virtual temperature and relative humidity, tvrh2t returns the temperature in degrees Kelvin

Special Purpose Functions

tloop(expr)	when time is varying dimension in the dimension environment, tloop function evaluates the expr at fixed times, then constructs the time series to obtain a final result that is the time varying

Vector Functions

hcurl(uexpr,vexpr)	calculates the vertical component of the curl (i.e.vorticity) at each grid pointusing finite differencing on the grids provided
hdivg(expr1,expr2)	calculates the horizontal divergence using the finite differencing
mag(uexpr,vexpr)	performs the calculation: sqrt(uexpr*uexpr+vexpr*vexpr)

Station Data Functions

gr2stn(grid_expr,stn_expr)	performs an interpolation from grid space back to station locations
oacres(grid_expr,stn_expr<,radii<first guess>>)	
	a Cressman objective analysis is performed on the station data to yield a gridded result representing the station data
stnave(expr,dexpr1,dexpr2<,-m cnt>)	
	takes an average of station data over time
stnmin(expr,dexpr1,dexpr2<,-m cnt>)	
	examines a time series of station data and returns the minimum value encountered for each station
stnmax(expr,dexpr1,dexpr2<,-m cnt>)	
	examines a time series of station data and returns the maximum value encountered for each station

Create PostScript files

Program: gxps [-c] [-r] [-d] [-i mfile] [-o ofile]

converts the GrADS meta file into a PostScript file. Command line options:

-c	color on a white background (=old gxpscw)
-r	color on a black background (=old gxpsc)
-d	add ctrl-d to the end of the file, useful if printing on HP 1200C/PS printer
-i mfile	where mfile is the name of the input GrADS meta file
-o ofile	where ofile is the name of the output PostScript file

Program: gxeps [-1][-2][-a | -l][-c][-r][-d][-L][-n][-s][-v] [-i mfile] [-o ofile]

converts the GrADS meta file into a PostScript file. Command line options:

-1	PostScript Level 1 output
-2	PostScript Level 2 output
-a	DIN A4 paper size
-c	color on a white background
-d	add ctrl-d to the end of the file, useful if printing on HP 1200C/PS printer
-l	US letter paper size
-L	ask for a label to be printed on the plot
-n	ask for a note to include in postscript file header
-r	color on a black background
-s	add a file & time stamp
-v	verbose
-i mfile	where mfile is the name of the input GrADS meta file
-o ofile	where ofile is the name of the output PostScript file

Create GIF files

Program: gxgif [-i mfile] [-o ofile]

converts the GrADS meta file into a GIF file. Command line options:

-i mfile	where mfile is the name of the input GrADS meta file
-o ofile	where ofile is the name of the output GIF file

Variables

complete specification for a variable name

abbrev.file#(dimexpr,dimexpr,...)

abbrev is the abbreviation for the variable as specified in the data descriptor file file# is the file number that contains this variable. The default initially is 1. dimexpr is a dimension expression that locally modifies the current dimension environment.

GrADS scripting language reference card version 1.7

(GrADS Version 1.7 beta 7) compiled by *Karin Meier-Fleischer*, DKRZ (beratung@dkrz.de

General Information

The GrADS scripting language, used via the GrADS run command, provides a similar capability to the exec command, except that a script may have variables, flow control and access GrADS command output. Scripts may be written to perform a variety of functions, such as allowing a user to point and click on the screen to select something, to animate and desired quantities, to annotate plots with information obtained from GrADS query commands.

Important: GrADS needs a carriage return after the last command line in the script file, otherwise GrADS won't execute this command line.

Variables

Script language variable names are 1 to 8 characters, beginning with an alphabetic character and containing letters or numbers only. The name is case sensitive. The contents of a script variable is always a character string! For some operations, the character string will be interpreted as a number.

Predefined variables **lat**
lon
lev
result
rec

String variables or **string constants** are enclosed either with single or double quotes.

name = 'Peter Pan'
or **name = "Peter Pan"**

Compound variables can be used to construct arrays in scripts. A compound variable has a variable name with segments seperated by periods.

varname.i.j

Example:
```
i = 10
j = 3
varname.i.j = 343
```
or `varname.10.3 = 343`

Note: The compound variable name MAY NOT be longer than 16 characters either BEFORE or AFTER substitution. GrADS scripting language is not particular efficient in handling large numbers of variables. Thus compound variables should not be used to create large arrays!

Global variables start with an underscore (_) and will keep its value throughout an entire script file using (also in functions).

_varname

Example: `_var1 = 1024`

Note: The global variables cannot be used in function headers
'function myfunc (_var1)' would be invalid!
It wouldn't make sense, cause it's a global variable!!!

Assignment

The format to assign a record is: **variable = expression**

The expression is evaluated, and the result is assigned to be the value of the indicated variable.

Logical values

Logical values are

TRUE 1
FALSE 0

Operators

The following operators are implemented:

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

Arithmetic operations are done in floating point. If the result is integral, the result string will be integer. A logical operator will give a character 0 (zero), if the result is FALSE, and a character 1 (one), if the result is TRUE.

Expressions

Script expression consists of operands, operatores and parentheses.
The precedence of the operators is

```
- !(unary)
/ *
+ -
%
= != > >= < <=
&
|
```

Within the same precedence level, operations are performed left to right. Parentheses modify the order of operation in the expected ways.

To concatenate two or more strings using the concatenate operator (%) or just two single quotes (' ') instead of the operator.

Example:
```
col1 = '16 17 18 19 20 '
col2 = '21 22 23 24 25 '
col3 = '26 27 28 29 30'
colors = col1%col2%col3
```
or `colors = col1"col2"col3`

```
'set ccols 'colors
```
is equal to `'set ccols 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30'`

Standard input/output

To write information to the terminal (standard output):

say expression

To write an input request string:

prompt expression

The 'prompt' command works the same way as 'say' except it does **not append** a carriage return!

To read an input string/value from the standard input:

pull variable

The script pauses for the user keyboard input (up to the carriage return), and the string entered by the user is assigned to the indicated variable name.

Examples:
```
line = 'Peter Pan, the flying one'
say line

prompt 'Enter latitude: '
pull lat
prompt 'Enter longitude: '
pull lon
'set lat 'lat
'set lon 'lon
```

To combine variables and comments writing to standard out:

```
say 'She said, it is 'line
```
produces She said, it is Peter Pan, the flying one

Control flow

IF Block:

```
if expression
  command
  .....
else
  command
  .....
endif
```
each must be seperated lines

optional

required!

Note: There is NO 'else if ' element implemented in GrADS!

Example: `if(i=10); j=20; endif`

is equal to
```
if(i=10)
   j=20
endif
```

WHILE Loop:

```
while expression
  command
  .....
endwhile
```
To continue the while loop use the **continue** command; to exit the while loop use the **break** command

Example:
```
t=1
while(t<10)
   'set t 't
   'd z'
   t = t + 1
endwhile
```

Functions

Functions are invoked as a script expression is beeing evaluated. Functions always have a single string result, but may have one or more string arguments! Functions are invoked by:

name(arg1, arg2, arg3, ..., argn)

If the function has no arguments, you must still provide the parentheses:

name()

To define a user own function by using the function definition record:

function name(var1, var2, var3, ..., varn)

To return from a function, use the return command:

return(expression)

The expression is optional, if not provided, a NULL string will be returned.

Example:
```
x = 10
y = 30
z = addit(x,y)
say 'Result of addition:  z = 'z
....
function addit(var1,var2)
   sum=var1+var2
return (sum)
```

terminal output: Result of addition: z = 40

Sending commands

The statement record consists only of an expression

expression

The expression is evaluated, and the resulting string is submitted to GrADS as a command. After this record is executed, the script variable '**result**' is given the value.

Examples:
```
hallo = 'draw string 4.0 8.0 HALLO'
hallo
```
or
```
'query'
say result
```
produces
```
GrADS Version 1.7Beta6 ---
LATS=GRIB_NCSA_HDF__SDF_READ=NCSA_netCDF_HDF
query or q Options:
 q config   List configuration of this build
 q files:   Lists open files
 q file n:  Gives info on particular file
 q define:  Lists currently defined variables
 q fwrite:  Fwrite Status
 q lats:    State of the GrADS-LATS Interface
 q dims:    Gives current dimension environment
 q time:    Gives current time
 q gxinfo:  Gives graphics environment info
 q shades:  Gives colors and levels of shaded contours
 q pos:     Waits for mouse click, returns position
 q w2gr:    Convert world to grid coordinates
 q gr2w:    Convert grid to world coordinates
 q w2xy:    Convert world to x,y screen coordinates
 q xy2w:    Convert x,y screen coordinates to world
            coordinates
 q gr2xy:   Convert grid to x,y screen coordinates
 q xy2gr:   Convert x,y screen to grid coordinates
 q pp2xy:   Convert pre-projected grid coordinates to
            screen x,y coordinates
 q defval:  Gives defined grid value given grid i,j
```

Intrinsic functions

To get a single word from a string:

res = subwrd(string,word)

The result is the nth 'word' from the string. If string is too short, the result is the NULL string. 'word' must be an integer value.

To get a single line from a string containing several lines:

res = sublin(string,line)

The result is the nth 'line' from the string. If the string has too few lines, the NULL string is returned. 'line' must be an integer value.

To get a part of a string:

res = substr(string,start,length)

The substring of 'string' starting at location 'start' for length 'length' will be returned. If the string is too short, the result will be short or the NULL string. 'start' and 'length' must be an integer value.

Examples:
```
'query time'
res = subwrd(result,3)
year = substr(res,9,4)
say year
```
produce e.g. 1880

The function sublin is very usefull, if you want to control opening, reading, writing and closing an extern ASCII file.
For example, the first record in the ASCII file 'the_title.txt' to be read is

Szenario A 1880 - 2099

The following part of a script will open, read and close the file, controling the status of each statement:

```
ret = read('the_title.txt')
code = sublin(ret,1)
if(code != 0)
   say 'read error  #'code
   'quit'
endif
title = sublin(ret,2)
'draw title 'title
ret = close('the_title.txt')
code = sublin(ret,1)
if(code != 0)
   say 'close error #'code
   'quit'
endif
```

I/O functions

To read records from an ASCII file:

res = read(filename)

The result is a string containing two lines. The first line is the return status and the second line is the record. The record may have a maximum of 80 characters. Use the sublin intrinsic function to seperate the lines.

The return status of read:

0	ok
1	open error
2	end of file
8	file open for write
9	I/O error

To write records to an ASCII output file:

res = write(filename, record <,append>)

The record is written to the 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 'filename'! 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.

The return status of write:

0	ok
1	open error
8	file open for read

To close an opened file:

res = close(filename)

The close command closes the named ASCII file and can also be used to rewind the file.

The return status of close:

0	ok
1	file not open

Examples:
```
'q file 1'
ret = result
res = write('file_1.txt', ret)
status = sublin(res,1)
if(status != 0)
   say 'write error  #'status
   'quit'
endif
res = close('file_1.txt')
status = sublin(res,1)
if(status != 0)
   say 'close error #'status
   'quit'
endif
```

Example script

The following example script draws 1200 shaded contour frames (1200 time records). The year, which will be used in the title string, is read from the 'query time' result. The private colors are defined in the function palette(). The 'set clip ..' command is used with the 'set dbuff on' and 'swap' commands to restrict the redraw of the plot to areas with changes from frame to frame.

At the DKRZ - Hamburg, videos were recorded using this kind of animation within GrADS. To achieve smooth animations, the single frame technique had been applied.

```
'reinit'
'open descriptor.ctl'
count = 0
rec = 1200
incr = 1;   t = 1
palette()
'set vpage 0.0 11.0 0.0 8.5'
'set parea 1.0 10.0 1.4 7.9'
'set dbuff on'
'set mpdset lowres'
'set map 0 1 10'
'set lat -90 90'
'set lon -180 180'
'set mpvals -180 180 -90 90'
'set mproj robinson'
'set grid on 5 0'
while (count < rec)
        'set t  't
        'q time'
        res = subwrd(result,3)
        year = substr(res,9,4)
        'set grads off'
        'set string 1 c 8'
        'set strsiz 0.23 0.26'
        'draw string 5.5 7.6 Aerosol - Control 'year
        'set gxout shaded'
        'set cint 1.0'
        'set cmin -4.0'
        'set cmax 4.0'
        'set clevs -4.0 -3.0 -2.0 -1.0 0.0 1.0 2.0 3.0 4.0'
        'set ccols  17   18   19   21   22  23  24  25  26  27'
                'display data'
        'set gxout contour'
        'set cterp off'
        'set csmooth off'
        'set cint 1.0'
        'set clab off'
                'display data'
        'run cbar.gs'
        'set clip 1.0 10.0 1.4 7.9'
        'swap'
    count = count + incr
    t = t + incr
endwhile

function palette()
        'set rgb  16  0  0  20'
        'set rgb  17  0  29  85'
        'set rgb  18  0  44  128'
        'set rgb  19  0  83  230'
        'set rgb  21  0  151  250'
        'set rgb  22  104  173  255'
        'set rgb  23  177  213  255'
        'set rgb  24  255  250  110'
        'set rgb  25  255  209  116'
        'set rgb  26  255  160  80'
        'set rgb  27  255  100  65'
return
```