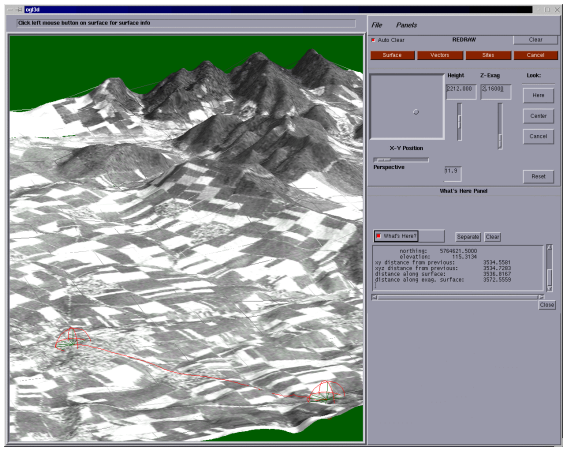
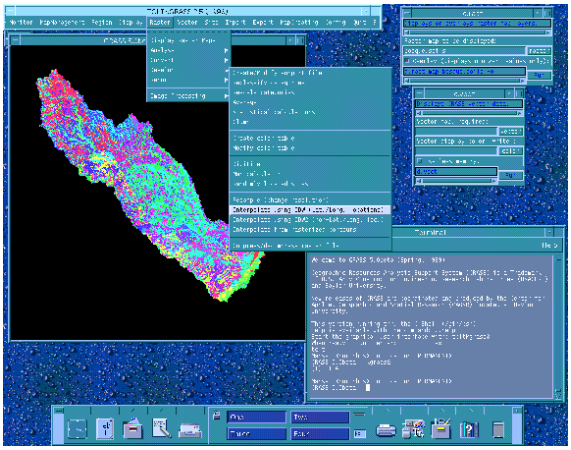
GRASS Reference Manual

General Commands





GRASS Development Team

USA Headquarters
Center for Applied Geographic & Spatial Research
Baylor University
P.O. Box 97351
Waco, Texas 76798-7351
USA

European Headquarters
Institute of Physical Geography-Landscape Ecology
University of Hannover
Schneiderberg 50
30167 Hannover
Germany

grass@baylor.edu

http://www.baylor.edu/~grass
http://www.geog.uni-hannover.de/grass/

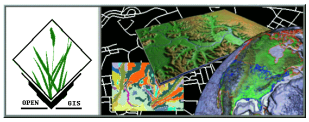


Table of Contents

GRASS Introduction

GRASS (Geographic Resources Analysis Support System) is a raster based GIS, vector GIS, image processing system, and graphics production system. GRASS contains over 200 programs and tools to render maps and images on monitor and paper; manipulate raster, vector, and sites data; process multi-spectral image data; and create, manage, and store spatial data. GRASS uses both an intuitive windows interface as well as command line syntax for ease of operations. GRASS can interface with commercial printers, plotters, digitizers, and databases to develop new data as well as manage existing data.

GRASS is ideal for use in engineering and land planning applications. Like other GIS packages, GRASS can display and manipulate vector data for roads, streams, boundaries, and other features. GRASS can also be used to keep maps updated with its integral digitizing functions. Another feature of GRASS is its ability to use raster, or cell, data. This is particularly important in spatial analysis and design. GRASS functions can convert between vector data to raster data for seamless integration.

GRASS' strengths lie in several fields. The simple user interface makes it an ideal platform for those learning about GIS for the first time. GRASS is capable of reading and writing maps and data to many popular commercial GIS packages including ARC/Info and Idrisi. Users wishing to write their own code can do so by examining existing source code, interfacing with the documented GIS libraries, and using the GRASS Programmers Manual. This allows more sophisticated functionality to be integrated in GRASS.

The ability to work with raster data gives GRASS the unique ability to function as a surface modeling system. GRASS contains more than 100 multi-function raster analysis and manipulation commands. Surface processes such as rainfall-runoff modeling, flowline construction (as shown), slope stability analysis, and spatial data analysis are just a few of the many applications of GRASS to engineering and land planning. Since many of the raster tools are multi-functional, users can create their own maps from GRASS data analysis.

In addition to standard two-dimensional analysis, GRASS allows users to view data in three-dimensions. Raster maps, vector maps, and sites data can be used for visualization. Example applications of such capabilities include airspace analysis for airport planning (as shown), terrain analysis and “flybys”, and spatial trends. Tools in GRASS allow the user to animate any spatial data available with options to switch between data layers “on-the-fly”. Data used in 3-D visualization may also be saved as still pictures, or as mpeg movie files for later replay and analysis.

Accompanying its land planning and engineering applications, GRASS contains a suite of tools to aid in hydrologic modeling and analysis. Currently, tools are also available for performing such functions as watershed analysis, curve number generation, flood analysis, and stream channel characteristics for comprehensive watershed modeling. Other GRASS programs can generate graphs, statistics, and charts of modeled and calibrated data. Additionally, GRASS can use field data for model input or simulate parameters based on numerical data.

In addition to the traditional command line version of GRASS, a new user interface, based on Tcl/Tk has been written. This puts the power of spatial analysis and modeling into an easy to use Graphical User Interface that is platform-independent. This intuitive user interface lets users quickly and easily view, manipulate, and use data. Nearly all of the programs available in GRASS are available in the new GUI, with the standard command-line still available, giving users all of the functionality of GRASS.

This manual is part of a comprehensive set of documentation written to support GRASS. This Users Guide consists of a complete set of command references for all current GRASS functions and tools, including examples. An installation guide and fact sheet guides users through the installation process. For those wishing to write their own spatial analysis and modeling applications for GRASS, a Programmers Guide is also available. GRASS runs on a variety of UNIX and Linux platforms including SUN SPARCstations and Ultras, HP, Silicon Graphics, and PC’s running Windows 95 and Windows NT.

The GRASS Development Team is currently working to further upgrade and enhance the capabilities of GRASS. Future developments include tools that give the user the ability to work completely in 3-D, a capability that does not exist in any other GIS package. Users will be able to work with raster elevation data as well as vector and sites data in the 3-D environment, adding to the

visualization capabilities of GRASS. Enhancements in the numerical processing functions of GRASS also now allow for floating-point operations to be performed on data.

For the latest information on GRASS contact the GRASS Development Team at grass@baylor.edu or visit our web sites at:

http://www.baylor.edu/~grass if you're in the U.S.

http://www.geog.uni-hannover.de/grass if you're in Europe

Look for our worldwide mirrors!

The GRASS Development Team is:

Bruce Byars and Markus Neteler are the development team leaders and coordinators.

Helena Mitasova and Bill Brown of the GMS Lab at UIUC have made significant contributions with the development of GRASS 5.

Additional authors include:
Lisa Zygo, Edward Zarecky, Jacques Bouchard, Steve Clamons, Brent Duncan, Jason Cipriano, Jim Westervelt, Michael Shapiro, Darrell McCauley, Dave Gerdes, Bill Hughes, Bernhard Reiter, Brook Milligan, Eliot Cline, Jaro Hofierka, Clay Cockrell, and Bob Lozar. See the web pages for author affiliations.

Note:

Many other people have contributed to the GRASS GIS. Without any one of them, GRASS would not exist in its current form. The authors of the individual programs are listed at the end of their manual page in the GRASS users manual, however, numerous authors of bug fixes and enhancements as well as people who have been working on coordination, integration, documentation and testing are not mentioned.

Please allow us to extend our most cordial thanks to all of you. If you contributed to GRASS at any point during its existence, let us know your name and e-mail address so we can add your name to the comprehensive on-line list.

To reference GRASS:

GRASS Development Team, 1999, Geographic Resources Analysis and Support System - GRASS: Baylor University, Waco, Texas.

GRASS Development Team
Center for Applied Geographic and Spatial Research
Baylor University
P.O. Box 97351
Waco, Texas, U.S.A. 76798-7351

exit

NAME
exit - Exits the user from the current GRASS session.

GRASS VERSION
4.x, 5.x

SYNOPSIS
exit

DESCRIPTION
The *exit* command ends the user's current GRASS session and returns the user to the directory in which he was working prior to entering GRASS. When the user exits GRASS, he is asked whether he wishes to save the files and data stored under his current mapset, and (if maps are present) whether the user wishes to selectively remove map layers, before exiting the system. By default, if the user presses RETURN without responding to these questions, all maps in the user's current mapset are saved. However, because such maps can consume much storage space on the computer, the user should remove any unneeded files before exiting. (The user can also remove data before attempting to exit GRASS using the *g.remove* command.)

Before typing *exit*, the polite user should remember to release the graphics display monitor (using the command *d.mon -r*) for use by other GRASS users. Otherwise, the display device may be locked for use by the user, even though the user has exited GRASS, until another user runs *d.mon* and unlocks this monitor for others' use.

Each time the user re-enters GRASS, the variables described in *g.gisenv* are re-set. If the user wishes to change the mapset, location, or data base on which he is working (i.e., those affected by any GRASS programs running during the user's current GRASS session), the user must exit GRASS and then re-enter GRASS and specify a different mapset, location, and/or data base location on the GRASS start-up page. When the user re-enters GRASS, these variable settings (the current mapset, location, and data base) are set by default to those used in the user's previous GRASS session, unless changed by the user.

NOTES
This program requires no command line arguments; the user simply types *exit* on the command line to exit GRASS.

SEE ALSO
d.mon, g.gisenv, g.remove

g.access

NAME
g.access - Controls user access to the current GRASS mapset.
(GRASS File Management Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
g.access

DESCRIPTION
This program allows the user to control access to the current mapset. Normally, any user can read data from any GRASS mapset. But sometimes it is desirable to prohibit access to certain sensitive data. The *g.access* command allows a user to restrict read and execute access to the current mapset (see UNIX chmod command). *g.access* will not modify write access to the current mapset.

The user may, for example, allow only users in the same UNIX group to read data files in the mapset, or restrict the mapset to personal use only.

After typing *g.access* the user will be presented with a screen page, which reflects the current mapset permissions. The user can then change them. The screen page looks like:

```
| LOCATION: spearfish    MAPSET: demo                                        |
|                                                                            |
| This program allows you to control access to your mapset by other users.  |
| Access may be granted/removed for everyone, or for everyone in your group.|
|                                                                            |
| Mark an 'x' to allow access;  erase the field to restrict access.          |
|                                                                            |
|GROUP: _x_                                                                  |
|OTHER: _x_                                                                  |
|                                                                            |
|AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE                         |
|    (OR <Ctrl-C> TO CANCEL)                                                 |
```

If you remove the x (using the space bar), access will be denied to that class of user (group or other). If you type an x, access will be granted to that class of user.

NOTES
There is no non-interactive version of *g.access*.

Under GRASS version 4.0, access to the mapset PERMANENT must be open to all users. This is because GRASS looks for the user's default geographic region definition settings and the location title in files that are stored under the PERMANENT mapset directory. The *g.access* command, therefore, will not allow you to restrict access to the PERMANENT mapset.

The *g.mapsets* command isn't smart enough to tell if access to a specified mapset is restricted, and the user is therefore allowed to include the names of restricted mapsets in his search path. However, the data in a restricted mapset is still protected; any attempts to look for or use data in a restricted mapset will fail. The user will simply not see any data listed for a restricted mapset.

SEE ALSO
UNIX manual entries for chmod and group, *g.mapsets*

AUTHOR
Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

g.ask

NAME
g.ask - Prompts the user for the names of GRASS data base files.
(GRASS File Management Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
g.ask help
g.ask type=name [prompt="string"] element=name [desc="string"] unixfile=name

DESCRIPTION
g.ask is designed for shell scripts that need to prompt the user for the name of a data base file in the user's current GRASS location. After *g.ask* is invoked with needed parameters, it will query the user for a file name of the specified type and element. After the user responds to this query, the program will write four lines to the UNIX output file specified by unixfile.

Parameters:
type=name The type of query. Options for name are old, new, any, and mapset; their functions are given below. "New", "any", and "mapset" only look in the user's current mapset.

old For existing data files anywhere in the user's mapset search path.

new Used to create a new file in the current mapset, which must not already exist there (if a file with this name exists there, it will not be overwritten).

any Creates a file in the current mapset, which may or may not already exist there. If a file with this name exists in the current mapset, it will be overwritten; if not, a new file with this name will be created.

mapset For files that must exist in the current mapset, the shell write wants the name of a file which exists in the user's current mapset. This type would be used instead of "old" if the file is to be modified.

prompt="string" The prompt to be displayed to the user. If more than one word, the prompt should be enclosed within double quotes ("").

element=name The name of the GRASS data base element (i.e., directory under a GRASS mapset) whose files are to be queried.

desc="string" A short description of the data base element which is meaningful to the user. If description contains more than one word, it should be enclosed within double quotes ("").

unixfile=name The name of a UNIX file to store the user's response. See next section for what is written to this file and how it can be used by shell scripts.

OUTPUT
Upon receiving the user's response to its request for a file name, *g.ask* writes four lines to the specified unixfile; this output file is placed in the user's current working directory, unless otherwise specified, and contains the following lines:

name='some_name'
mapset='some_mapset'

fullname='some_fullname'
file='some_fullpath'

The output is /bin/sh commands to set the variable name to the file name specified by the user (of the element and type requested by *g.ask*), mapset to the GRASS mapset in which this file resides (or will be created), fullname is the name with the mapset embedded in it, and file to the full UNIX path name identifying this file. These variables may be set in the /bin/sh as follows:

.unixfile

The '.' is a shell command which means read the unixfile and execute the commands found there. It is NOT part of the unixfile name and MUST be followed by a space.

NOTES
The user may choose to simply hit the return key and not enter a file name. If this happens the variables will be set as follows:

name=
mapset=
fullname=
file=

The following is a way to test for this case:

```
if [ ! "$file" ]
then
exit
fi
```

SEE ALSO
d.ask, g.filename, g.findfile, g.gisenv

AUTHOR
Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

g.copy

NAME
g.copy - Copies available data files in the user's current mapset search path and location to the appropriate element directories under the user's current mapset.
(GRASS File Management Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
g.copy
g.copy help
g.copy [rast=from,to] [vect=from,to] [icon=from,to] [labels=from,to] [sites=from,to] [region=from,to] [group=from,to]

DESCRIPTION
A user may access data stored under the other mapsets listed in his mapset search path. However, the user may only modify data stored under his own current mapset. *g.copy* allows the user to copy existing data files from other mapsets to the user's current mapset ($MAPSET). The files to be copied must exist in the user's current mapset search path and location; output is sent to the relevant data element directory(ies) under the user's current mapset.

The user specifies the type(s) of data files he wishes to copy (raster, vector, etc.), the name of the existing file to be copied (i.e., the from file name), and the name of the new file copy to be placed in the user's current mapset (the to file name). This information can be given either (non- interactively) on the command line, or entered in response to program prompts given via the standard *parser* interface described in the manual entry for *parser*.

Information can be entered on the command line in the following format:

g.copy [rast=from,to] [vect=from,to] [icon=from,to] [labels=from,to] [sites=from,to] [region=from,to] [group=from,to]

For example, if the user wished to copy the existing raster file "soils" to a file called "soils.ph" and to copy an existing vector file "roads" to a file called "rds.old", the user could type:

g.copy rast=soils, soils.ph vect=roads, rds.old

Data files can also be specified by their mapsets. For example, the below command copies the raster file named soils from the mapset wilson to a new file called soils to be placed under the user's current mapset:

g.copy rast='soils@wilson',soils

If no mapset name is specified, *g.copy* searches for the named from map in each of the mapset directories listed in the user's current mapset search path in the order in which mapsets are listed there (see *g.mapsets*).

If the user does not enter parameter values but instead types only *g.copy* on the command line the program will prompt the user for a data type, the name of a file of this data type to copy, and the name of a new file to hold the copy. After both file names have been entered, the copy is created and the user is again prompted for a data element to be copied, until the user hits RETURN. When prompted for file names, the user may enter 'list' to see a list of existing files, or hit RETURN to end the file listing.

Parameters:
rast=from,to where from is an existing raster map layer to be copied, and to is the name given to the copy.

vect=from,to where from is an existing binary vector map layer to be copied, and to is the name given to the copy.

icon=from,to where from is an existing paint icon file to be copied, and to is the name given to the copy.

labels=from,to where from is an existing /paint/labels file to be copied, and to is the name given to the copy.

sites=from,to where from is an existing site_lists file to be copied, and to is the name given to the copy.

region=from,to where from is an existing region definition (windows) file to be copied, and to is the name given to the copy.

group=from,to where from is an existing imagery group file to be copied, and to is the name given to the copy.

NOTE
If a file has support files (e.g., as do raster data files), these support files will also be copied.

SEE ALSO
g.access, g.list, g.mapsets, g.remove, g.rename, parser

AUTHOR
Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

g.filename

NAME
g.filename - Prints GRASS data base file names.
(GRASS File Management Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
g.filename
g.filename help
g.filename element=name mapset=name file=name

DESCRIPTION
g.filename is designed for Bourne shell scripts that need to know the full UNIX file name for raster map layers, vector files, site list files, geographic region definition (windows) files, imagery group files, etc., in the GRASS data base. If the user runs *g.filename* without command line arguments (i.e., simply types *g.filename*), this program will prompt the user for input using the standard *parser* interface described in the manual entry for *parser*.

Parameters:
element=name The name of a GRASS data base element (i.e., directory within the GRASS mapset location).

mapset=name The name of a GRASS data base mapset. As a convenience, a single dot (.) can be used to designate the current mapset.

file=name The name of a GRASS data base file.

OUTPUT
g.filename writes one line to standard output:

file='full_file_pathname'

The output is a /bin/sh command to set the variable specified by the file name to the full UNIX path name for the data base file. This variable may be set in the /bin/sh as follows:

eval `g.filename element=name mapset=name file=name`

NOTES
This routine generates the filename, but does not care if the file (or mapset or element) exists or not. This feature allows shell scripts to create new data base files as well as use existing ones. If the mapset is the current mapset, *g.filename* automatically creates the element specified if it doesn't already exist. This makes it easy to add new files to the database without having to worry about the existence of the required data base directories. (This program will not create a new mapset, however, if that specified does not currently exist.) The program exits with a 0 if everything is ok; it exits with a non-zero value if there is an error, in which case file='full_file_pathname' is not output.

SEE ALSO
g.ask, g.findfile, g.gisenv, parser

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

g.findfile

NAME
g.findfile - Searches for GRASS database files and sets variables for the shell.
(GRASS File Management Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
g.findfile
g.findfile help
g.findfile element=name [mapset=name] file=name

DESCRIPTION
g.findfile is designed for Bourne shell scripts that need to search for raster map layer files, vector files, site list files, geographic region definition (windows) files, and imagery group files in the GRASS data base. If the user runs *g.findfile* without command line arguments, he will be prompted for the names of a GRASS element, file, and mapset, through the standard *parser* interface (see manual entry for *parser*).

Parameters:
element=name The data base element (i.e., directory within a GRASS mapset) to be searched.

mapset=name The mapset in which to search for the specified file name. If not specified, all mapsets in the user's GRASS search path are searched. Otherwise, the specified mapset is searched. As a convenience, if specified as a single dot (.) only the current mapset is searched.

file=name The name of a GRASS data file (of the stated element type) for which to search.

OUTPUT
g.findfile writes four lines to standard output:

name='file_name'
mapset='mapset_name'
file='unix_filename'
fullname='grass_fullname'

The output is /bin/sh commands to set the variable name to the GRASS data base file name, mapset to the mapset in which the file resides, and file to the full UNIX path name for the named file. These variables may be set in the /bin/sh as follows:

eval `g.findfile element=name mapset=name name=name`

NOTES
If the specified file does not exist, the variables will be set as follows:

name=
mapset=
fullname=
file=

The following is a way to test for this case:

```
if [ ! "$file" ]
then
exit
fi
```

SEE ALSO
g.ask, g.filename, g.gisenv, g.mapsets, parser

AUTHOR
Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

g.gisenv

NAME
g.gisenv - Outputs the user's current GRASS variable settings.
(GRASS File Management Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
g.gisenv
g.gisenv [variable_name]

DESCRIPTION
When a user runs GRASS, certain variables are set specifying the GRASS data base, location, mapset, peripheral device drivers, etc., being used in the current GRASS session. These variable name settings are recognized as long as the user is running a GRASS session.

No prompts are given to the user when running *g.gisenv*. If run without arguments, *g.gisenv* lists all of the user's current GRASS variable settings. Results are sent to standard output, and may look like this:

```
GISDBASE=/usr/grass4/data
LOCATION_NAME=spearfish
MAPSET=PERMANENT
```

In this example, the full path name of the user's current location (i.e., $LOCATION_NAME) is /usr/grass4/data/spearfish, and the full path name of the user's current mapset (i.e., $MAPSET) is /usr/grass4/data/spearfish/PERMANENT.

If the user specifies a variable_name on the command line (e.g., *g.gisenv* MAPSET), only the value for that particular GRASS variable is output to standard output. Possible variable names depend on the user's system.

While other variables may be associated with each GRASS session (e.g., DIGITIZER, PAINTER, DISPLAY, and other variables), those stated below are essential.

GISDBASE - The $GISDBASE is a directory in which all users' GRASS data are stored. Within the $GISDBASE, data are segregated into subdirectories (called "locations") based on the map coordinate system used and the geographic extent of the data. Each "location" directory itself contains subdirectories called "mapsets"; each "mapset" stores "data base elements" -- the directories (e.g., the cell, cellhd, dig, etc., directories) in which GRASS data files are actually stored.

LOCATION_NAME - The user must choose to work with the under a single GRASS location within any given GRASS session; this location is then called the current GRASS location, and is specified by the variable $LOCATION_NAME. The $LOCATION_NAME is the GRASS data base location whose data will be affected by any GRASS commands issued during the user's current GRASS session, and is a subdirectory of the current $GISDBASE. Each "location" directory can contain multiple "mapset" directories (including the special mapset "PERMANENT"). Maps stored under the same GRASS LOCATION_NAME (and/or within the same MAPSET) must use the same coordinate system and typically fall within the boundaries of the same geographic region (a.k.a., "location").

MAPSET - Each "mapset" contains a set of maps relevant to the LOCATION_NAME directory in which it appears. Each LOCATION_NAME can contain multiple mapsets. (Mapsets which fall under the same LOCATION_NAME all contain data geographically relevant to the LOCATION_NAME, and all store

data in the same map coordinate system. Frequently, maps are placed into different mapsets to distinguish file ownership -- e.g., each user might have his own mapset, storing any maps that he has created and/or are relevant to his work.) During each GRASS session, the user must choose one mapset to be the current mapset; the current mapset setting is given by $MAPSET, and is a subdirectory of $LOCATION_NAME. During a single GRASS session, the user can use available data in any of the mapsets stored under the current LOCATION_NAME directory that are in the user's mapset search path and accessible by the user. However, within a single GRASS session, the user only has write access to data stored under the current mapset (specified by the variable $MAPSET).

Each "mapset" stores GRASS data base elements (i.e., the directories in which GRASS data files are stored). Any maps created or modified by the user in the current GRASS session will be stored here. The MAPSET directory "PERMANENT" is generally reserved for the set of maps that form the base set for all users working under each LOCATION_NAME.

Once within a GRASS session, GRASS users have access only to the data under a single GRASS data base directory (the current GRASS data base, specified by the variable $GISDBASE), and to a single GRASS location directory (the current location, specified by the variable $LOCATION_NAME). Within a single session, the user may only modify the data in the current mapset (specified by the variable $MAPSET), but may use data available under other mapsets under the same LOCATION_NAME.

All of these names must be legal names on the user's current system. For UNIX users, names less than 14 characters and containing no non-printing or space codes are permissible. Examples of permissible names include: one, mymap, VeGe_map, and 1_for_me. The underscore character can safely be used in place of a blank for multiple-word names.

NOTES
The output from *g.gisenv* when invoked without arguments is directly usable by /bin/sh. The following command will cast each variable into the UNIX environment:

eval `g.gisenv`

This works only for /bin/sh. The format of the output is not compatible with other UNIX shells.

SEE ALSO
g.access, g.ask, g.filename, g.findfile, g.mapsets

AUTHOR
Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

g.help

NAME
g.help - GRASS help facility.
(GRASS Help Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
g.help

DESCRIPTION
g.help provides the user with functional information on GRASS programs, a glossary, and access to on-line User's Reference Manual entries. The help facility is accessed by simply typing *g.help* on the command line. The user can then wend his way through a series of menus (organized by functional area) and key-word searches.

On-line reference manual entries can also be accessed directly, through the GRASS *g.manual* command.

SEE ALSO
GRASS User's Reference Manual
g.manual

AUTHOR
James Westervelt, U.S. Army Construction Engineering Research Laboratory
Deb Brinegar, U.S. Army Construction Engineering Research Laboratory
Mary Martin, U.S. Army Construction Engineering Research Laboratory

Note: The help facility uses the hyper program written by Jim Westervelt.

g.list

NAME
g.list - Lists available GRASS database files of the user-specified data type to standard output.
(GRASS File Management Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
g.list
g.list help
g.list [-f] type=datatype [mapset=name]

DESCRIPTION
g.list allows the user to list user-specified, available and accessible files from mapsets under the user's current location. When invoked simply as *g.list*, the program prompts the user for the type of data to be listed from all mapsets in the user's current mapset search path. The user can list files from a mapset not listed in the current mapset search path by running the program non-interactively, specifying the (optional) flag setting and parameter values on the command line. Program flag and parameters are described below.

Flag:
-f Returns a verbose file listing that includes map titles.

Parameter:
type=datatype The type of data to be listed.

Options:
rast Raster files
vect Binary vector files
icon Paint icon files
labels Paint labels files
sites Site list files
region Region definition files
group Imagery group files
mapset=name The name of a mapset to be searched for files of the specified type. Any mapset name under the current location, whether or not it is listed in the user's current mapset search path, can be specified.
Default: If unspecified, files of the specified type from all mapsets in the user's current search path will be listed to standard output.

NOTES
If the user requests that files from a mapset to which access has been restricted (see *g.access* be listed, no files from this mapset will be listed.

SEE ALSO
g.access, g.mapsets

AUTHOR
Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

g.man2html

NAME
g.man2html - convert a GRASS manual page to HTML
(GRASS Shell Script)

GRASS VERSION
4.x

SYNOPSIS
g.man2html
g.man2html help
g.man2html name

DESCRIPTION
g.man2html is a Bourne shell script that converts the roff source of a GRASS manual page to HyperText Markup Language (HTML) and prints the results to standard output.

OPTIONS
Parameter:
name Name of a GRASS man page (full path to roff source)

NOTES
g.man2html is likely to be used by programmers when preparing documentation for their code.

FILES
$GISBASE/scripts/g.man2html

SEE ALSO
g.nroff, g.manual, start.man.sh

AUTHOR
James Darrell McCauley, Agricultural Engineering Purdue University

g.manual

NAME
g.manual - Accesses GRASS User's Reference Manual entries.
(GRASS File Management Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
g.manual
g.manual help
g.manual [-1aefs] [entries=name[,name,...]]

DESCRIPTION
The *g.manual* command provides user access to the on-line GRASS User's Reference Manual entries. The user may request a list of available manual entries, and may get a manual entry printed to the terminal screen and/or to the line printer.

g.manual can be run either interactively or non-interactively. If the user types

g.manual

on the command line without program arguments, the program will prompt the user for a manual entry to display. The user may enter "list" to get a section-by-section listing of the manual entries available. Once the user has viewed the desired information it may be printed by responding to the questions appropriately.

The user can run the program non-interactively, by specifying the appropriate options and/or the name(s) of the manual entries to be displayed.

OPTIONS
Flags:
-1 This option will list all manual entries, one per line.

-a This option will list all manual entries in a more appealing format. The manual page list will be separated by manual section.

-e This option tells *g.manual* to ignore empty manual sections when printing the listings from the *-1* or *-a* options.

-f This option will add formfeeds to output listing when using the *-a* option.

-s This option will cause *g.manual* to run silently. Instead of displaying the manual page it will simply set the exit status to:

0 if entry exists, or
1 if it does not exist.

These entries may also be accessed through the *g.help* command.

SEE ALSO
GRASS User's Reference Manual
g.help

AUTHOR
Kurt Buehler, U.S. Army Construction Engineering Research Laboratory

g.mapsets

NAME
g.mapsets - Modifies the user's current mapset search path, affecting the user's access to data existing under the other GRASS mapsets in the current location.
(GRASS File Management Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
g.mapsets
g.mapsets help
g.mapsets [-lp] [mapset=name[,name,...]]

DESCRIPTION
A mapset holds a distinct set of data layers, each relevant to the same (or a subset of the same) geographic region, and each drawn in the same map coordinate system. At the outset of every GRASS session, the user identifies a GRASS data base, location, and mapset that are to be the user's current data base, current location, and current mapset for the duration of the session; any maps created by the user during the session will be stored under the current mapset ($MAPSET) set at the session's outset.

The user can add, modify, and delete data layers that exist under his current mapset. Although the user can also access (i.e., use) data that are stored under other mapsets in the same GRASS location, the user can only make permanent changes (create or modify data) located in the current mapset. The user's mapset search path lists the order in which other mapsets in the same GRASS location can be searched and their data accessed by the user. The user can modify the listing and order in which these mapsets are accessed by modifying the mapset search path; this can be done using the *g.mapsets* command. This program allows the user to use other's relevant map data without altering the original data layer, and without taking up disk space with a copy of the original map.

g.mapsets shows the user available mapsets under the current GRASS location, lists mapsets to which the user currently has access, and lists the order in which accessible mapsets will be accessed by GRASS programs searching for data files. The user is then given the opportunity to add or delete mapset names from his search path, or modify the order in which mapsets will be accessed.

When the user specifies the name of a data base element file (e.g., a particular vector file, raster file, imagery group file, etc.) to a GRASS program, the program searches for the named file under each of the mapsets listed in the user's mapset search path in the order listed there until the program finds a file of the given name. (Users can also specify a file by its mapset, to make explicit the mapset from which the file is to be drawn; e.g., the command:

g.copy rast='soils.file@PERMANENT',my.soils

ensures that a new file named my.soils is to be a copy of the file soils.file from the mapset PERMANENT.)

It is common for a user to have the special mapset PERMANENT included in his mapset search path, as this mapset typically contains finished base maps relevant to many applications. Often, other mapsets which contain sets of interpreted map layers will be likewise included in the user's mapset search path. Suppose, for example, that the mapset Soil_Maps contains interpreted soils map layers to which the user wants access. The mapset Soil_Maps should then be included in the user's search path variable.

The mapset search path is saved as part of the current mapset. When the user works with that mapset in subsequent GRASS sessions, the previously saved mapset search path will be used (and will continue to be used until it is modified by the user with *g.mapsets*).

OPTIONS

Flags:

-l List all available mapsets under the user's current location.

-p Print the user's current mapset search path to standard output.

Parameter:

mapset=name [,name,...] Name(s) of existing GRASS mapset(s) under the current location.

g.mapsets sets the current mapset search path to the mapsets named on the command line. If *g.mapsets* is typed but no mapset names are specified by the user on the command line, the program will print the user's current mapset search path, list available mapsets, and prompt the user for a new mapset search path listing.

NOTES

Users can restrict others' access to their mapset files through use of the GRASS program *g.access*. Mapsets to which access is restricted can still be listed in another's mapset search path; however, access to these mapsets will remain restricted.

SEE ALSO

g.access, g.copy, g.gisenv, g.list

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory
Greg Koerper, ManTech Environmental Technology, Inc.

g.nroff

NAME
g.nroff - runs nroff on a GRASS manual page
(GRASS Shell Script)

GRASS VERSION
4.x

SYNOPSIS
g.nroff help
g.nroff name [value]
g.nroff < name

DESCRIPTION
g.nroff is a Bourne shell script that runs nroff on a GRASS manual page and prints the results to standard output.

OPTIONS
Parameters:
name Name of a GRASS man page (full path to roff source)

section Integer section of the manual.
1 for main programs
2 for alpha programs
3 for shell scripts
4 for contributed, untested code
5 file format descriptions
Default: 4

NOTES
g.nroff is likely to be used by programmers when preparing documentation for their code.

FILES
$GISBASE/scripts/g.nroff

SEE ALSO
g.manual, start.man.sh

AUTHOR
James Darrell McCauley, Agricultural Engineering, Purdue University

g.region

NAME
g.region - Program to manage the boundary definitions for the geographic region.
(GRASS Region Management Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
g.region
g.region help
g.region [-dgpu] [region=name] [raster=name] [vector=name] [sites=name] [3dview=name] [n=value] [s=value] [e=value] [w=value] [res=value] [nsres=value] [ewres=value] [zoom=name] [align=name] [save=name]

DESCRIPTION
The *g.region* program allows the user to manage the settings of the current geographic region. These regional boundaries can be set by the user directly and/or set from a region definition file (stored under the windows directory in the user's current mapset). The user can create, modify, and store as many geographic region definitions as desired for any given mapset. However, only one of these geographic region definitions will be current at any given moment, for a specified mapset; i.e., GRASS programs that respect the geographic region settings will use the current geographic region settings.

INTERACTIVE PROGRAM USE: MAIN MENU
The main menu consists of an information section listing the current GRASS data base LOCATION, MAPSET, and CURRENT REGION, followed by user options:

```
------------------------------------------------------------
|REGION FACILITY                                           |
| LOCATION: sampleMAPSET: grass                            |
|                                                          |
| CURRENT REGION: N=5167600  S=5156755  RES=50  ROWS=216   |
|  E=729314   W=705924   RES=50  COLS=467                  |
| PROJECTION: 1 (UTM)                                      |
| ZONE: 13                                                 |
|                                                          |
|                                                          |
| Please select one of the following options               |
|                                                          |
|Current Region    Region Database                         |
|                                                          |
| 1 Modify current region6 Save current region in          |
|   directlythe database                                   |
|                                                          |
| 2 Set from default region   7 Create a new region        |
|                                                          |
| 3 Set from a database  8 Modify an existing region       |
|   region                                                 |
|                                                          |
| 4 Set from a raster map                                  |
|                                                          |
| 5 Set from a vector map                                  |
|                                                          |
| RETURN to quit                                           |
------------------------------------------------------------
```

DEFINITIONS

Region:

Here, a region refers to a geographic area with some defined boundaries, based on a specific map coordinate system and map projection. Each region also has associated with it the specific east-west and north- south resolutions of its smallest units (rectangular units called "cells").

The region's boundaries are given as the northernmost, southernmost, easternmost, and westernmost points that define its extent. The north and south boundaries are commonly called northings, while the east and west boundaries are called eastings.

The region's cell resolution defines the size of the smallest piece of data recognized (imported, analyzed, displayed, stored, etc.) by GRASS programs affected by the current region settings. The north-south and east-west cell resolutions need not be the same, thus allowing non-square data cells to exist.

Default Region:

Each GRASS LOCATION_NAME has a fixed geographic region, called the default geographic region (stored in the region file DEFAULT_WIND under the special mapset PERMANENT) that defines the extent of the database. While this provides a starting point for defining new geographic regions, user-defined geographic regions need not fall within this geographic region.

Current Region:

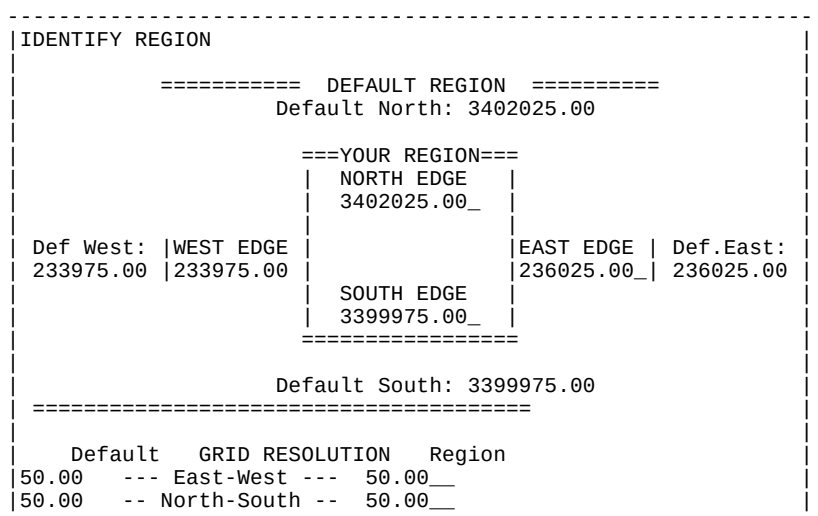
Each mapset has a current geographic region. This region defines the geographic area in which all GRASS displays and analyses will be done. Data will be resampled, if necessary, to meet the cell resolutions of the current geographic region setting.

Region Data Base:

Each GRASS MAPSET may contain any number of pre- defined, and named, geographic regions. These region definitions are stored in the user's current mapset location under the windows directory (also referred to as the user's data base of region definitions). Any of these pre-defined geographic regions may be selected, by name, to become the current geographic region. Users may also access saved region definitions stored under other mapsets in the current location, if these mapsets are included in the user's mapset search path.

REGION EDIT PROMPT

Most of the options will require the user to edit a geographic region, be it the current geographic region or one stored in the user's data base of region definitions (the windows directory). A standard prompt is used to perform this edit. An example is shown below:



```
|                                                              |
|                                                              |
|AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE           |
----------------------------------------------------------------
```

The fields NORTH EDGE, SOUTH EDGE, WEST EDGE and EAST EDGE, are the boundaries of the geographic region that the user can change. The fields Default North, Default South, Def West and Def East are the boundaries of the default geographic region that are displayed for reference and cannot be changed. The two GRID RESOLUTION Region fields (east-west, and north-south) are the geographic region's cell resolutions that the user can change. The two GRID RESOLUTION Default fields list the resolutions of the default geographic region; these are displayed for reference and cannot be changed here by the user.

REGION MANAGEMENT MENU OPTIONS

1 Modify the current geographic region directly
Allows the user to edit the current region.

2 Set current geographic region from default region
Copies the default region to the current geographic region, and then lets the user edit the current geographic region.

3 Set current geographic region from a data base geographic region
Allows the user to select a geographic region by name from the database of geographic regions to become the current geographic region, and then lets the user edit the current geographic region.

Note: geographic region definition files may be selected from other mapsets as well, if accessible and in the user's mapset search path.

4 Set current geographic region from a raster (cell) map layer
Allows the user to select a raster map layer, copies the cell header for this map layer to the current geographic region, and then lets the user edit the current geographic region. This option is useful when subsequent GRASS operations will be used to produce a raster map layer from one input raster map layer and it is necessary that the result coincide with the input raster map layer.

5 Save the current geographic region (window) in the database
Allows the user to save the current geographic region settings in the user's database of such settings. These files are stored in the windows directory under the user's current mapset. This option is useful when the current geographic region is set directly using option 2, or even by another GRASS program (e.g., *d.display*). This option installs an otherwise temporary geographic region setting into the geographic region definition database for recall when needed.

6 Create a new data base geographic region setting
Creates a new geographic region definition in the user's database of such settings in the windows directory under the current mapset, using the geographic region edit prompt described above. After the geographic region definition is created, the user is asked if this geographic region setting should also be used as the current geographic region.

7 Modify a data base geographic region setting
Modifies a geographic region setting (in the data base of such settings in the windows directory of the current mapset), using the geographic region edit prompt. After the changes have been made, the user is asked if this geographic region setting should also be used as the current geographic region.

NON-INTERACTIVE PROGRAM USE

Alternately, the user can modify the settings of the current geographic region by specifying all needed parameters on the command line. The user enters the command *g.region* [parms], where [parms] are the following parameters and/or flags:

Flags:

-d Set current region settings equal to default region settings.

-g Print the current region settings (shell script style) in a format that can be given back to *g.region* on its command line.

-p Print the current region settings.

-u Do not update the current region file settings. Allows the user to temporarily use a different region setting, without saving this setting.

Parameters:

region=name Make current region settings same as the named region file settings

raster=name Make current region settings same as those in the named raster map's cell header. But see zoom=name option, below.

vector=name Make the current region settings the same as those of the named vector map.

sites=name Set the current region to the smallest region encompassing all coordinates in the named site_lists file, aligned with the current region.

3dview=name Make current region settings same as those in the named 3dview file, which holds the region that was current when the 3dview was saved.

n=value Set map coordinate value for the region's northern edge to *value*

s=value Set map coordinate value for the region's southern edge to *value*

e=value Set map coordinate value for the region's eastern edge to *value*

w=value Set map coordinate value for the region's western edge to *value*

res=value Set grid resolution (both north-south and east-west) to *value*

nsres=value Set north-south grid resolution to *value*

ewres=value Set east-west grid resolution to *value*

zoom=name Set current region settings to the smallest region encompassing all non-zero data in the named raster map layer that fall inside the user's current region. If the user also includes the *raster=name* option on the command line, *zoom=name* will set the current region settings to the smallest region encompassing all non-zero data in the named zoom map that fall inside the region stated in the cell header for the named raster map.

align=name Set the current resolution equal to that of the named raster map, and align the current region to a row and column edge in the named map. Alignment only moves the existing region edges outward to the edges of the next nearest cell in the named raster map -- not to the named map's edges. To perform the latter function, use the *raster=name* option.

save=name Save current region settings in the named region file

EXAMPLES

g.region n=7360100 e=699000

will reset the northing and easting for the current region, but leave the south edge, west edge, and the region cell resolutions unchanged.

g.region -dp s=698000

will set the current region from the default region for the GRASS data base location, reset the south edge to 698000, and then print the result.

g.region n=n+1000 w=w500

The n=value may also be specified as a function of its current value: n=n+value increases the current northing, while n=nvalue decreases it. This is also true for s=value, e=value, and w=value. In this example the current region's northern boundary is extended by 1000 units and the current region's western boundary is decreased by 500 units.

g.region n=s+1000 e=w+1000

This form allows the user to set the region boundary values relative to one another. Here, the northern boundary coordinate is set equal to 1000 units larger than the southern boundary's coordinate value, and the eastern boundary's coordinate value is set equal to 1000 units larger than the western boundary's coordinate value. The corresponding forms s=n-value and w=e-value may be used to set the values of the region's southern and western boundaries, relative to the northern and eastern boundary values.

g.region raster=soils

This form will make the current region settings exactly the same as those given in the cell header file for the raster map layer soils.

g.region raster=soils zoom=soils

This form will first look up the cell header file for the raster map layer soils, use this as the current region setting, and then shrink the region down to the smallest region which still encompasses all non-zero data in the map layer soils. Note that if the parameter raster=soils were not specified, the zoom would move to encompass all non-zero data values in the soils map that were located within the current region setting.

g.region -up raster=soils

The -u option suppresses the re-setting of the current region definition. This can be useful when it is desired to only extract region information. In this case, the cell header file for the soils map layer is printed without changing the current region settings.

g.region -u raster=soils zoom=soils save=soils

This will zoom into the smallest region which encompasses all non-zero soils data values, and save the new region settings in a file to be called soils and stored under the windows directory in the user's current mapset. The current region settings are not changed.

g.region -p

This will print the current region in the format:

```
projection:   1 (UTM)
zone:    15
north:   4294050.00
south:   4249950.00
east:    526050.00
west:    500950.00
nsres:   100.00
ewres:   100.00
```

```
rows:    441
cols:    251
```

g.region -g
The *-g* option prints the region in the following format:

```
n=4294050.00
s=4249950.00
e=526050.00
w=500950.00
nsres=100.00
ewres=100.00
```

This format does not have the rows and columns, but can be fed back into *g.region* on its command line.

The *-p* (or *-g*) option is recognized last. This means that all changes are applied to the region settings before printing occurs.

NOTE
After all updates have been applied, the current region's southern and western boundaries are (silently) adjusted so that the north/south distance is a multiple of the north/south resolution and that the east/west distance is a multiple of the east/west resolution.

SEE ALSO
d.display, d.zoom, g.access, g.mapsets

AUTHOR
Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

g.remove

NAME
g.remove - Removes data base element files from the user's current mapset. (GRASS File Management Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
g.remove
g.remove help
g.remove [rast=name[,name,...]] [vect=name[,name,...]]
[icon=name[,name,...]][labels=name[,name,...]] [sites=name[,name,...]] [region=name[,name,...]]
[group=name[,name,...]]

DESCRIPTION
g.remove allows the user to remove specified data base element files from the current mapset. If *g.remove* is invoked without arguments on the command line, a menu will appear listing possible data element types, as below:

```
1  raster maps
2  vector maps
3  paint icon files
4  paint label files
5  site list files
6  region files
7  imagery group files
RETURN to exit
```

Once the element type is selected, the user is prompted to name a specific file of this element type for removal. (This list will vary, depending on what files currently exist in the user's mapset.) The specified file is removed, and the user is again prompted for the name of a file of this element type to be removed. When prompted for a file name, the user may enter list to see a list of existing files of this element type, or hit RETURN to get back to the above menu.

Alternately, the user can specify the data base element type and file(s) to be removed on the command line. Data base element types are specified by the names to the left, below.

Parameters:

rast=name[,name,...] Name(s) of raster file(s) to be removed.

vect=name[,name,...] Name(s) of vector file(s) to be removed.

icon=name[,name,...] Name(s) of paint icon file(s) to be removed.

labels=name[,name,...] Name(s) of paint labels file(s) to be removed.

sites=name[,name,...] Name(s) of site list file(s) to be removed.

region=name[,name,...] Name(s) of region file(s) to be removed.

group=name[,name,...] Name(s) of imagery group file(s) to be removed.

The data base element file(s) named by the user on the command line are subsequently removed from the user's current mapset.

EXAMPLE
For example, the below command will cause the raster files named soils, slope, and temp, the vector files named roads and rail, and the imagery group files named nhap.1 and nhap.2, and these files' associated support files (e.g., cell header files, category files, etc.), to be removed from the user's current mapset.

g.remove rast=soils,slope,temp vect=roads,rail group=nhap.1,nhap.2

NOTE
If a particular data base element file has support files associated with it (e.g., as is commonly the case with raster files), *g.remove* will remove these support files along with the data base element file specified.

The user can only use *g.remove* to remove data files existing under the user's current mapset.

FILES
$GISBASE/etc/element_list lists the element types whose files can be removed by the user.

SEE ALSO
g.copy, g.list, g.rename

AUTHOR
Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

g.rename

NAME
g.rename - To rename data base element files in the user's current mapset.
(GRASS File Management Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
g.rename
g.rename help
g.rename [rast=old,new] [vect=old,new] [icon=old,new] [labels=old,new] [sites=old,new] [region=old,new] [group=old,new]

DESCRIPTION
g.rename allows the user to rename data base element files in the user's current mapset. The user can specify all necessary information to *g.rename* on the command line, by specifying: the type of data base element to be renamed (one or more of: rast, vect, icon, labels, sites, region, and group); the specific file element in the current mapset to be renamed (old); and the new name to be assigned to this file element (new) in the current mapset. The file element old is then renamed to new.

Users can also simply type *g.rename* without arguments on the command line, to receive a menu of existing data base element types and files from which to choose for renaming:

```
1    raster maps
2    binary vector maps
3    paint icon files
4    paint label files
5    site list files
6    region definition files
7    imagery group files
RETURN to exit
```

NOTE
If a data base element has support files (e.g., as is commonly the case with raster files), these support files also are renamed.

If the user attempts to rename a file to itself by setting the new file name equal to the old file name (e.g., *g.rename rast=soils,soils*), *g.rename* will not execute the rename, but instead state that no rename is needed. However, *g.rename* will allow the user to overwrite other existing files in the current mapset by making the new file name that of an already existing file.

SEE ALSO
g.copy, g.list, g.remove

AUTHOR
Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

g.setproj

NAME
g.setproj - Allows the user to create the PROJ_INFO and the PROJ_UNITS files to record the projection information associated with a current location.

GRASS VERSION
4.x, 5.x

SYNOPSIS
g.setproj

DESCRIPTION
Allows a user to create a PROJ_INFO file in the PERMANENT mapset of the current location. PROJ_INFO file is used to record the projection information associated with the specified mapset.

NOTES
User running *g.setproj* must own PERMANENT mapset. It is highly recommended to run *g.setproj* after creating a new location so that conversion programs (such as *v.proj*) can be run.

The current location must not contain a PROJ_INFO or PROJ_UNITS file.

The user will be prompted for the projection name. The specification of any projection other than ll and stp will generate a request to the user for a name of a standard ellipse.

The projections of aea, lcc, merc, leae (version 5.x), leac (version 5.x), and tmerc will generate a request to the user for the prime meridian and standard parallel for the output map. The projection of stp will generate a request to the user for the state abbreviation and choice of zone for the output map.

The user will be prompted for the spheroid and zone of the UTM projection.

SEE ALSO
v.proj, *m.proj*

AUTHOR
Irina Kosinovsky, US Army CERL, Champaign, IL
Morten Hulden, morten@tor.ngb.se - rewrote and added 121 projections

g.tempfile

NAME
g.tempfile - Creates a temporary file and prints the file name
(GRASS File Management Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
g.tempfile help
g.tempfile pid=value

DESCRIPTION
g.tempfile is designed for shell scripts that need to use large temporary files. GRASS provides a mechanism for temporary files that does not depend on /tmp. GRASS temporary files are created in the database with the assumption that there will be enough space under the database for large files. GRASS periodically removes temporary files that have been left behind by programs that failed to remove them before terminating.

g.tempfile creates a unique file and prints the name. The user is required to provide a process-id, which will be used as part of the name of the file. Most Unix shells provide a way to get the process id of the current shell. For /bin/sh and /bin/csh this is $$. It is recommended that $$ be specified as the process-id for *g.tempfile*.

EXAMPLE
For /bin/sh scripts the following syntax should be used:

temp1=`g.tempfile pid=$$`
temp2=`g.tempfile pid=$$`

For /bin/csh scripts, the following can be used:

set temp1=`g.tempfile pid=$$`
set temp2=`g.tempfile pid=$$`

NOTES
Each call to *g.tempfile* creates a different (i.e. unique) name.

Although GRASS does eventually get around to removing tempfiles that have been left behind, the programmer should make every effort to remove these files. They often get large and take up disk space. If you write /bin/sh scripts, learn to use the /bin/sh trap command. If you write /bin/csh scripts, learn to use the /bin/csh onintr command.

AUTHOR
Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

g.version

NAME
g.version - Outputs the GRASS version number and date.
(GRASS File Management Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
g.version
g.version help

DESCRIPTION
g.version prints to standard output the GRASS version number and date, in the form:

GRASS 4.0 (Summer 1991)

NOTES
This program requires no command line arguments; the user simply types *g.version* on the command line to see the version number and date of the GRASS software currently being run by the user.

AUTHOR
Michael Shapiro, U.S. Army Construction Engineering Research Laboratory