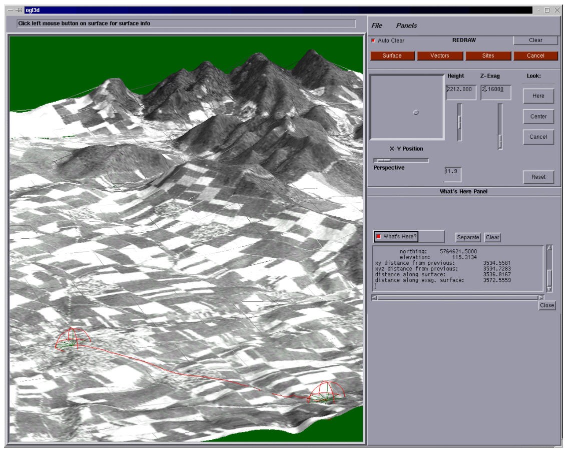
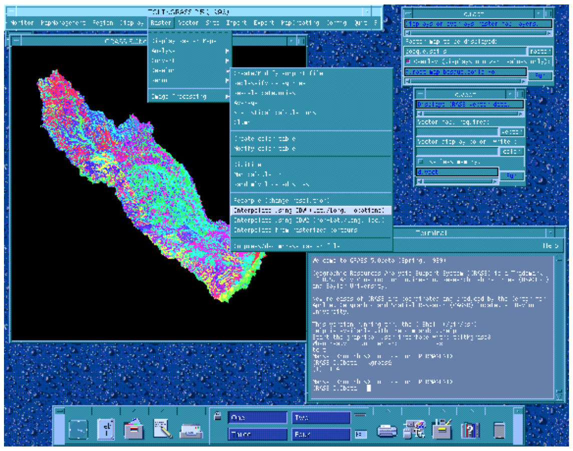
GRASS Reference Manual

Paint, Photo, and Postscript 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

p.chart

NAME
p.chart **-** Prints the color chart of the currently selected printer.
(GRASS Hardcopy Output Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
p.chart

DESCRIPTION
This function prints the color chart of the user's hardcopy color printer. The colors in the chart are numbered with the (device-dependent) number associated with each color. This function is useful both for the *p.colors* command as well as for color selection with *p.map*.

The user must select a printer using *p.select* before running this command.

SEE ALSO
d.colors, p.colors, p.map, p.select, r.colors

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

p.colors

NAME
p.colors **-** Allows the user to develop a color table that associates map categories with user-specified printer colors.
(GRASS Hardcopy Output Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
p.colors

DESCRIPTION
This function allows the user to modify a color table for a raster map layer, by assigning colors to the categories in the raster map layer based on printer color numbers (instead of red, green, blue percentages).

The user will need to have the printer color chart available to properly select the colors (see manual entry for *p.chart*).

The *p.colors* function allows the user to exercise perfect control over the colors, which appear in the hardcopy image.

NOTES
If the user assigns a printer color number outside of the printer's range, *p.colors* will not complain, but will not save the out-of-range printer color number to the raster map layer's color table.

This command modifies the color table associated with a map layer, and will therefore also affect the colors in which a map layer is displayed on the user's graphics monitor. (See map color table files stored under $LOCATION/colr and $LOCATION/colr2.)

SEE ALSO
d.colors, p.chart, p.map, r.colors

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

p.icons

NAME
p.icons **-** Creates and modifies icons for map display and output.
(GRASS Hardcopy Output Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
p.icons

DESCRIPTION
This program allows the user to create and maintain icons, which are used by the *p.map* and *d.icons* commands to depict sites.

Icon files created by the user are stored under $LOCATION/icons.

SEE ALSO
d.icons, d.points, d.sites, p.map, s.menu

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

p.labels

NAME
p.labels **-** Create labels for hardcopy maps.
(GRASS Hardcopy Output Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
p.labels

DESCRIPTION
This module allows the user to create or modify labels files. These labels files, which are stored in the database, define text information for printing with *p.map* and for graphics display with *d.paint.labels*. Each label has components, which determine the text, the location of the text on the image, its size, and the background for the text.

The interface is a screen-oriented input/edit layout. Each label is entered with a single screen. After filling in the required information (described below), the user hits <ESC> to accept the label and start a new one. After the last label has been accepted, the user then hits the <ESC> one more time (on an empty label screen) to exit the module and save the labels.

SCREEN LAYOUT
The screen layout for the labels looks like this:

```
----------------------------------------------------------------------
PAINT LABELS: labelfile    new labels[1]

TEXT: _________________________________________________SKIP: no__
 _________________________________________________
 _________________________________________________
 _________________________________________________

LOCATION:  EAST: ___________________  OFFSET: _________
NORTH: ___________________  OFFSET: _________
 PLACEMENT: center_____________

FONT:    standard___________
TEXT SIZE:    500______
TEXT COLOR:   black______________  WIDTH: 1________
HIGHLIGHT COLOR:   none_______________  WIDTH: 0________

BACKGROUND COLOR:  white______________
BORDER COLOR: black______________
OPAQUE TO VECTORS: yes______


    AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
   (OR <Ctrl-C> TO CANCEL)
```

The label information that must be provided is:

TEXT: Up to four lines of text. Lines in multiple line labels will appear one above the next.

SKIP: yes|no. If no, label will be printed. If yes, the label will be retained in the file but not printed.

LOCATION: Determines where the text will be located on the image. The user specifies the easting and northing, and (optionally) specifies a vertical and horizontal offset (in printer pixels) from the specified easting/northing. (The vertical offset will shift the location to the south if positive, north if negative. The horizontal offset will shift the location east if positive, west if negative.) These offsets are provided to allow finer placement of
labels.

PLACEMENT: Determines which part of the label to which the location refers. If placement is unspecified, the label is centered (center), by default. Label placement may be specified as:

lower left(lower left corner of the text)
lower right (lower right corner of the text)
lower center (bottom center of the text)

upper left(upper left corner of the text)
upper right (upper right corner of the text)
upper center (top center of the text)

center (center of the text)

FONT: This specifies the font to use. The following fonts are available:

cyrilc gothgbt gothgrt gothitt greekc greekcs greekp greeks italicc italiccs italict romanc romancs romand romans romant scriptc scripts

The word standard can be used to specify the default font (which is romans).

TEXT SIZE: This determines the size of the letters. The size specifies the vertical height of the letters in meters on the ground. Thus text will grow or shrink depending on the scale at which the map is drawn. The default text size, if none is specified, is 500.

TEXT COLOR: This selects the text color. If unspecified, the label's text is drawn in black, by default. The text color can be specified in one of four ways:

1) By color name:
aqua black blue brown cyan gray green grey indigo magenta orange purple red violet white yellow

2) As red, green, blue percentages. for example: .5 .4 .7
(This form is not supported by *d.paint.labels*.)

3) By printer color number to get the exact printer color.
(This form is not supported by *d.paint.labels*.)

4) Specify none to suppress the lettering.

WIDTH: This determines the line thickness of the letters. The normal text width should be set to 1. Larger numbers can be used to simulate bold face. (*d.paint.labels* ignores this value and always uses 1. 1 is also the default width to which the width is set by paint, if none is specified by the user.)

HIGHLIGHT COLOR: The text can be highlighted in another color so that it appears to be in two colors. The text is drawn first in this color at a wider line width, and then redrawn in the text color at the regular line width. No highlight color (none) is used, by default, if unspecified by the user. To specify use of no highlight color, specify none. (See TEXT COLOR above for a list of permissible color names.)

HIGHLIGHT WIDTH: Specifies how far from the text lines (in units of pixels) the highlight color should extend. The default highlight width is set to 0 (i.e., no highlight color).

BACKGROUND COLOR: Text may be boxed in a solid color by specifying a background color. Specify none for no background. The default background color setting, if unspecified by the user, is white. (See TEXT COLOR above for a list of permissible color names).

BORDER COLOR: Select a color for the border around the background. Specify none to suppress the border. The default border color used, if unspecified, is black. (See TEXT COLOR above for a list of permissible color names).

OPAQUE TO VECTORS: yes|no. This field only has meaning if a background color is selected. yes will prevent vector lines from entering the background. no will allow vector lines to enter the background. The default setting, if unspecified by the user, is yes.

SEE ALSO
p.map, p.select, p.icons, d.paint.labels

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

p.map

NAME
p.map **-** Hardcopy color map output utility.
(GRASS Hardcopy Output Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
p.map
p.map help
p.map [input=name] [scale=mapscale]

DESCRIPTION
p.map produces hardcopy color map products on your system's color output device. Output can include a raster map, any number of vector overlays, site data, text labels, and other spatial data. This program has 2 distinct modes of operation. The command-line mode requires the user to prepare a file of mapping instructions describing the various spatial and textual information to be printed prior to running *p.map*. The interactive mode (i.e., no command-line arguments) will prompt the user for items to be mapped and does not require the user to prepare a file of instructions.

The command line parameters are:

input=name File containing mapping instructions. (or enter input=**-** to enter from keyboard).
These instructions are described in detail below.

scale=mapscale Scale of the output map, e.g. 1:25000
Default: 1panel

This parameter is provided as a convenience. It is identical to the scale mapping instruction described below.

Note: the user must select an output device using *p.select* before running *p.map*. Also, the preview device can be selected to view the output from *p.map* on the graphics monitor instead of sending it to a paper printer.

MAPPING INSTRUCTIONS
The mapping instructions allow the user to specify various spatial data to be plotted. These instructions are normally prepared in a regular text file using a system editor. Some instructions are single line instructions while others are multiple line. Multiple line instructions consist of the main instruction followed by a subsection of one or more additional instructions.

colormode
Selects the appropriate method to color raster map layers and images.

USAGE: colormode approx|best

There are two methods: approximate and best. From a user perspective, approximate can be used for raster map layers with few categories, such as soils, and best should be used for images like LANDSAT images or NHAP photos, or maps with many categories. The approximate mode treats each pixel independently, giving it the printer color that best approximates the true color. The best mode "blends" colors from pixel to pixel using a dithering technique to simulate more colors than the printer can actually print. The default, if unspecified, is best.

This example would select the approximate colormode. The assumption is that the raster map layer being printed is has few colors or that the colors would not look good dithered.

EXAMPLE: colormode approx

colortable
Includes the color table for the raster map layer in the legend below the map

USAGE: colortable [y|n]

The color table will display the colors for each raster map layer category value and the category label. To get a color table, you must have previously requested a raster map layer. Omitting the colortable instruction would result in no color table. Note: Be careful about asking for color tables for raster map layers which have many categories, such as elevation. This could result in the printing of an extremely long color table!!

This example would print a color table as part of the legend to the map.
EXAMPLE: colortable y

comments
Prints comments beneath the map . You may submit comment text line by line during *p.map* execution or a via a prepared comments file.

```
USAGE:   comments [commentfile]
    comments
    end
```

This example prints the comment "This is a comment" in the legend below the map.

```
EXAMPLE:   comments
   This is a comment.
   end
```

This example prints whatever is in the file veg.comments in the legend below the map.

```
EXAMPLE:   raster vegetation
 comments veg.comments
   end
```

Presumably, the file veg.comments contain comments pertaining to the raster map layer vegetation, such as "This map was created by classifying a LANDSAT TM image".

defpat
Defines area fill patterns to be used in the setpat instruction.

```
USAGE:   defpat name
    pattern
    color # color
    end
```

The pattern is given a name on the defpat instruction line. The pattern which follows is composed of a sequence of numbers 0-9 (and blanks, which are equivalent to 0). The blanks or zeros indicate holes in the pattern where the normal category color would show through. The other number 1-9 indicate pattern pixels and can be assigned any color. The default color for all the digits will be black unless specified with the color instruction. The color option will begin by entering the word color followed by one of the digits (1-9) in the pattern, followed by one of the NAMED COLORS. This should be repeated for each of the digits specified to avoid using black. The instruction end terminates the pattern definition. Of course, the user can define more patterns by entering more defpat instructions.

NOTE: Do NOT indent the pattern. Leading blank spaces will be interpreted as 0's.

This example creates a black horizontal line pattern.

```
EXAMPLE:   defpat horiz
1
0
0
0
color 1 black
end
```

This example creates a green vertical line pattern.

```
EXAMPLE:   defpat vert
1000
color 1 green
end
```

This example creates a red diagonal line pattern.

```
EXAMPLE:   defpat diag
00001
0001
001
01
1
color 1 red
end
```

This example creates a two-toned tree pattern with orange trunks and green leaves.

```
EXAMPLE:   defpat tree
2
222
22122
22 1 22
1

2
222
22122
22 1 22
1
color 1 orange
color 2 black
end
```

grid

Overlays a coordinate grid onto the output map.

```
USAGE:   grid spacing
   color color
   numbers # [color]
   end
```

The spacing of the grid is given (in the geographic coordinate system units) on the main instruction line. the subsection instructions allow the user to specify the color of the grid lines, whether coordinate numbers should appear on the grid lines, and if they should appear every grid line (1), every other grid line (2), etc., and what color the numbers should be. The defaults are black grid lines, unnumbered.

This example would overlay a green grid with a spacing of 10000 meters (for a metered database, like UTM) onto the output map. Alternate grid lines would be numbered with red numbers.

```
EXAMPLE:   grid 10000
   color green
   numbers 2 red
   end
```

labels
Selects a labels file for output (see manual entry for *p.labels*).

```
USAGE:   labels  labelfile|list
```

This example would paint labels from the labels file called town.names. Presumably, these labels would indicate the names of towns on the map.

```
EXAMPLE:   labels town.names
```

line
Draws lines on the output map.

```
USAGE:   line east north east north
   line x% y% x% y%
   color color
   width #
   masked [y|n]
   end
```

The beginning and ending points of the line are entered on the main instruction. These points can be defined either by map coordinates or by using percentages of the geographic region. The user may also specify line color, width in pixels, and if the line is to be masked by the current mask. (See manual entry for *r.mask* for more information on the mask.)

This example would draw a yellow line from the point x=10% y=80% to the point x=30% y=70%. This line would be 2 pixels wide and would appear even if there is a mask.

```
EXAMPLE:   line 10% 80% 30% 70%
   color yellow
   width 2
   masked n
   end
```

Of course, multiple lines may be drawn with multiple line instructions.

outline
Outlines the areas of a raster map layer with a specified color.

```
USAGE:   outline
   color  color
   end
```

Distinct areas of the raster map will be separated from each other visually by drawing a border (or outline) in the specified color (default: black). Note: it is important the user enter the instruction end even if a color is not chosen. (It is hoped that in the future the outline of a different raster map layer other than the one currently being painted may be placed on the map.)

This example would outline the category areas of the soils raster map layer in grey.

```
EXAMPLE:   raster soils
   outline
   color grey
   end
```

point
Places additional points or icons on the output map.

```
USAGE:   point east north
   point x% y%
   color color
   icon iconfile|list
   size #
   masked [y|n]
   end
```

The point location is entered in the main instruction line by giving either the map coordinates or by using percentages of the geographic region. The user may also specify the point color, the icon file to be used to represent the point location (see the manual entry for *p.icons*), the size of the icon in integer multiples of the pattern in the icon file, and whether the point is to be masked by the current mask. (See manual entry for *r.mask* for more information on the mask.)

This example would place a purple diamond (from icon file diamond) at the point (E456000 N7890000). This diamond would be the same size it is in the diamond icon file and would not be masked by the current mask.

```
EXAMPLE:   point 456000 7890000
   color purple
   icon diamond
   size 1
   masked n
   end
```

Of course, multiple points may be drawn with multiple point instructions.

raster
Selects a raster map layer for output.

```
USAGE:   raster mapname|list
```

For each *p.map* run, only one raster map layer can be requested. If no raster map layer is requested, a completely white map will be produced. It can be useful to select no raster map layer in order to provide a white background for vector images.

This example would paint a map of the raster map layer soils.

```
EXAMPLE:   raster soils
```

read
Provides *p.map* with a previously prepared input stream.

```
USAGE:   read previously prepared UNIX file
```

Mapping instructions can be placed into a file and read into *p.map*.

Note: *p.map* will not search for this file. The user must be in the correct directory or specify the full path on the read instruction. (Note to /bin/csh users: ~ won't work with this instruction).

This example reads the UNIX file pmap.roads into *p.map*. This file may contain all the *p.map* instructions for placing the vector map layer roads onto the output map.

```
EXAMPLE: read pmap.roads
```

The user may have created this file because this vector map layer is particularly useful for many *p.map* outputs. By using the read option, the user need not enter all the input for the vector instruction, but simply read the previously prepared file with the correct instructions.

region
Places the outline of a smaller geographic region on the output.

```
USAGE:   region regionfile|list
   color color
   width #
   end
```

Geographic region settings are created and saved using *g.region*. The *p.map* region option can be used to show an outline of a smaller region, which was printed on a separate run of *p.map* on other user-created maps.

The user can specify the color and the width (in pixel units) of the outline. The default is a black border of one pixel width.

This example would place a white outline, 2 pixels wide, of the geographic region called fire.zones onto the output map. This geographic region would have been created and saved using *g.region*.

```
EXAMPLE:   region fire.zones
   color white
   width 2
   end
```

scale
Selects a scale for the output map.

```
USAGE:   scale scale
```

The scale can be selected either as:

a relative ratio, e.g. 1:25000;

an absolute width of the printed map, e.g. 10 inches; the number of printed paper panels, e.g. 3 panels; the number of miles per inch, e.g. 1 inch equals 4 miles.

This example would set the scale of the map to 1 unit = 25000 units.

```
EXAMPLE:   scale 1:25000
```

setcolor
Overrides the color assigned to one or more categories of the raster map layer.

```
USAGE:   setcolor cat(s) color
```

This example would set the color for categories 2,5 and 8 of the raster map layer watersheds to white and category 10 to green. (NOTE: no spaces are inserted between the category values.)

```
EXAMPLE:   raster watersheds
 setcolor 2,5,8 white
 setcolor 10 green
```

Of course, setcolor can be requested more than once to override the default color for additional categories. More than one category can be changed for each request by listing all the category values separated by commas (but with no spaces).

setpat
Assigns a (previously defined) pattern on a raster map layer category.

```
USAGE:   setpat cat name
  orsetpat builtin
  orsetpat all
```

The user can choose to use: the name of a specific pattern created using defpat (see above); the patterns built into *p.map*; or all the patterns the user may have created.

This example assigns the vertical pattern created using defpat (see example in defpat above) to category 3 of the raster map layer vegetation and the tree pattern (see example in defpat above) to category 10.

```
EXAMPLE:   raster veg
 setpat 3 vert
 setpat 10 tree
```

This example reads a previously prepared UNIX file horiz.pat with the correct defpat instructions for creating a black horizontal pattern (see example in defpat above) and assigns that pattern to category 5 of the raster map layer soils via the setpat instruction.

```
EXAMPLE:   raster soils
 read horiz.pat
 setpat 5 horiz
```

To select the builtin patterns:

```
EXAMPLE:   raster soils
 setpat builtin
```

To select individual builtin patterns:

```
EXAMPLE:   raster soils
 setpat 5 #1
 setpat 10 #2
```

sites
Selects sites data to be placed on the output map (see manual entry for *s.menu*).

```
USAGE:   sites sitemap|list
 color color
 icon iconfile|list
 size #
 desc [y|n]
 end
```

The user may specify the color of the sites (see section on NAMED COLORS below); the icon to be used to represent the presence of a site (see the manual entry for *p.icons*); the size of the icon (number of times larger than the size it is in the icon file); and whether or not the description associated with each site is also to be printed.

This example would paint a sites map with blue windmills (from an icon file created by the user using the *p.icons* GRASS command) placed at all windmill locations (from a sites list). These windmills would be two times larger than the size of the icon in the icon file and have descriptions from the sites list file printed beside them.

```
EXAMPLE:   sites windmills
   color blue
   icon windmill
   size 2
   desc y
   end
```

text
Places text on the map.

```
USAGE:   text  east north text
```

```
 text  x% y% text
 font fontname
 color color|none
 width #
 hcolor color|none
 hwidth #
 background color|none
 border color|none
 size #
 ref reference point
 xoffset #
 yoffset #
 opaque [y|n]
 end
```

The user specifies where the text will be placed by providing map coordinates or percentages of the geographic region map. The text follows these coordinates on the same instruction line. More than one line of text can be specified by annotating the end of a line with \n (e.g. USA\nCERL).

The user can then specify various text features:

font: cyrilc gothgbt gothgrt gothitt greekc greekcs greekp greeks italicc italiccs italict romanc romancs romand romans romant scriptc scripts (The default font is romans);

color (see NAMED COLORS);

width of the lines used to draw the text (to make thicker letters);

size as the vertical height of the letters in meters on the ground (text size will grow or shrink depending on the scale at which the map is painted); the highlight color (hcolor) and the width of the highlight color (hwidth);

the text-enclosing-box background color; the text box border color;

ref. This reference point specifies the text handle - what part of the text should be placed on the location specified by the map coordinates. Reference points can refer to: [lower|upper|center] [left|right|center] of the text to be printed; yoffset, which provides finer placement of text by shifting the text a vertical distance in pixels from the specified north. The vertical offset will shift the location to the south if positive, north if negative;

xoffset, which shifts the text a horizontal distance in pixels from the specified east The horizontal offset will shift the location east if positive, west if negative;

whether or not the text should be opaque to vectors. Entering no to the opaque option will allow the user to see any vectors, which go through the text's background box. Otherwise, they will end at the box's edge.

This example would place the text SPEARFISH LAND COVER at the coordinates E650000 N7365000. The text would be a total of 3 pixels wide (2 pixels of red text and 1 pixel black highlight), have a white background enclosed in a red box, and be 500 meters in size. The lower right corner of the text would be centered over the coordinates provided. All vectors on the map would stop at the border of this text.

```
EXAMPLE:   text 650000 7365000 SPEARFISH LAND COVER
   font romand
   color red
   width 2
   hcolor black
   hwidth 1
   background white
   border red
   size 500
   ref lower left
   opaque y
   end
```

vector
Selects a vector map layer for output.

```
USAGE:   vector vectormap|list
 color [#] color
 width #
 hcolor color
 hwidth #
 masked [y|n]
 style  0-9
 end
```

The user can specify the color of the vectors; the width of the vectors lines in pixels; the highlight color (hcolor) for the vector lines; the width of the highlight color (hwidth) in pixels; whether or not the raster map layer is to be masked by the current mask (see manual entry *r.mask* for more information on the mask); and the line style. The line style allows the vectors to be dashed in different patterns and colors. This is done by typing a series of numbers (0-9) in a desired sequence or pattern. Colors for the numbers (1-9) can be assigned using the color instruction. Blanks and non-digit characters are recognized as 0's. Using 0 would allow the colors of the raster map layer (or the background color if no raster map layer was selected) to show through.

This example would paint a map of the vector map layer named streams. These streams would be a total of 3 pixels wide (the inner two pixels blue and the outer highlight pixel white). The map would not show streams outside of the current mask.

```
EXAMPLE:   vector streams
   color blue
   width 2
   hcolor white
   hwidth 1
   masked y
   end
```

This example would paint a map of the vector map layer roads. These roads would be 2 pixels wide and would be dashed blank-black-red (the blank areas would show what lies under the roads). This map would show roads inside and outside of the current mask.

```
EXAMPLE:   vector roads
   width 2
   style 001122
   color 1 black
   color 2 red
   masked n
   end
```

verbose
Changes the amount of talking *p.map* will do.

USAGE: verbose [0|1|2]

A higher value implies more chatter. The default is 2. This example sets the amount of chatter to a minimum.

EXAMPLE: verbose 0

end
Terminates input and begin painting the map.

USAGE: end

NAMED COLORS
The following are the colors that are accepted by *p.map*:

aqua black blue brown cyan gray green grey indigo magenta orange purple red violet white yellow

ICONS VS. PATTERNS
Icons and patterns as used in *p.map* are not the same thing. Patterns are defined and are normally used to cover those extended areas covered by a raster map layer category. A pattern will repeat above, below and adjacent to itself. Icons are used to represent a single point.

Patterns are supported directly within *p.map* using the defpat instruction, while icons must be created using the *p.icons* program.

EXAMPLE *p.map* INPUT FILE
The following is an example of a *p.map* script file. The file has been name spear.soils. For the purposes of illustration, the file is in two columns. This script file can be entered at the command line:

p.map input=spear.soils

```
raster soils                defpat diag
vector streams              000001
   color blue               00001
   width 2                  0001
   hcolor white             001
   hwidth 1                 01
   masked y                 1
   end                      color 1 red
vector roads                end
   width 2                  setpat 4 diag
   style 001122
   color 1 black
   color 2 red
   masked n
   end
labels town.names
region subregion
   color white
   width 2
   end
grid 10000
   color green
   numbers 2 red
   end
outline
   color black
   end
colortable y
comments
   This is a comment
   end
scale 1:25000
setcolor 6,8,9 white
setcolor 10 green
text 608000 3476004 SPEARFISH SOILS MAP
color red
width 2
hcolor black
hwidth 1
background white
border red
size 500
ref lower left
opaque y
end
line 606969 3423092 616969 3423092
color yellow
width 2
opaque yes
end
point 40% 60%
color purple
icon diamond
size 2
masked n
end
end
```

INTERACTIVE MODE
If the user simply enter *p.map* without arguments, then a simple prompting session occurs. Some, but not all of the non-interactive requests are available at this level.

SEE ALSO
p.chart, p.icons, p.labels, p.select

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

p.map.new

NAME
p.map.new **-** Color map output utility.
(GRASS Hardcopy Output Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
p.map.new
p.map.new [input= name] [scale= mapscale]

DESCRIPTION
The *p.map.new* command produces color maps for output on a color hardcopy device or a graphics monitor. Output can include a raster map, any number of vector overlays, site data, text labels, and other map elements.

This command has three modes of operation. The command-line mode requires a previously prepared file of mapping instructions describing the map elements to be printed. The interactive mode (i.e., no command-line arguments) will prompt the user for items to be mapped and does not require the user to prepare a file of instructions. The keyboard mode is started by entering a hyphen (**-**) for the input parameter. The *p.map.new* instructions would then be entered via the keyboard.

The command-line parameters are:

input=name File containing mapping instructions (or enter input= **-** to enter instructions from the keyboard). These instructions are described in detail below.

scale=mapscale Scale of the output map, e.g. 1:25000
Default: 1 panel
The options for this parameter are identical to the scale mapping instruction described below. If a scale instruction is present in an input file, it is superseded by the command-line scale parameter.

An output device can be selected using *p.select* before running *p.map.new*. Valid devices include on-line hardcopy devices, plus preview, preview2, and ppm. See manual entry for *p.select*.

The current geographic region determines the area that is mapped using *p.map.new*.

NON-INTERACTIVE MAPPING INSTRUCTIONS
Mapping instructions allow the user to specify various map elements to be plotted. These instructions are normally prepared in an ASCII text file using a system editor. All of the listed mapping instructions are usable in a prepared file in the command-line mode. Not all of them are available in the interactive and keyboard modes.

Some instructions are single line instructions while others are multiple line. Multiple-line instructions consist of the main instruction followed by a subsection of one or more additional instructions. All multiple-line instructions must be completed by the end terminator.

Some instructions, such as those using data layers, icons, or labels, access files via the current mapset search path.

barscale   Places a barscale on the output map.

USAGE: *barscale east north*
```
   barscale  x% y%
   unit  ft| mi| m| km
   length  #
   interval  #
   style  dash| tick
   width  #
   color  color
   textsize  #
   textcolor  color| none
   font  font
   background  color| none
   border  color| none
   end
```

The location of the zero point of the scale bar is entered on the first instruction line. The location can be defined either by map coordinates or by percentages of the map area, where 0% 0% is the lower left corner of the map.

The user specifies the barscale unit of measurement, the total length using that unit, and the length of one interval (a smaller length evenly divisible into the total length).

The style of the scale bar can be specified. The dash style has solid lines representing each interval, separated by gaps. The tick style has a solid total length with vertical ticks marking each interval.

The user can also specify the width of the bar in pixels, its color (see VALID COLORS NAMES), the textcolor, textsize in geographic units, font (see VALID FONT NAMES), background color, and border color.

The barscale instruction set must be completed with the end terminator.

This example would result in a scale bar representing two kilometers. Vertical ticks would be placed at the scale origin, the mid-point, and at the end. The black bar and its accompanying black text would overlay a white box trimmed be a red border.

EXAMPLE: barscale 605000 4915000
```
   unit  km
   length  2
   interval  1
   style  tick
   width  2
   color  black
   textcolor  black
   textsize  150
   background  white
   border  red
   end
```

colormode   Selects the method to portray the colors of the raster map layer or image.

USAGE: colormode approx| best

There are two options for colormode: approx and best. The approx option should be used for raster map layers with few categories, and best should be used for images like LANDSAT images or NHAP photos, or maps with very many categories. The approx mode treats each pixel independently, giving it the printer color that best approximates the true color. The best mode "blends" colors from pixel to pixel using a dithering technique to simulate more colors than the printer can actually print. If unspecified, the default is best.

This example would select the approx colormode. The assumption is that the raster map layer being printed has few colors or that the colors would not look good dithered.

EXAMPLE: *colormode approx*

colortable   Includes the color table for the raster map layer in the area below the map on hardcopy output.

USAGE: colortable [y| n]

The color table will display the colors for each raster map layer category and the category value. The colortable instruction can not precede the raster instruction in the *p.map.new* input. The color table is not shown when the output device is the color monitor.

The user should be careful about asking for color tables for raster map layers that have very many categories, such as an elevation layer. This could result in the printing of an extremely long and generally useless color table!

This example would print a color table below the data area of the map.
EXAMPLE: colortable y

comments   Prints comments beneath the map on hardcopy output.

```
USAGE:   comments  [commentfile]
    comments
    end
```

Comment text can be entered in the *p.map.new* file or from a separate, previously prepared file. Comments are not shown when the output device is the color monitor. The comments instruction set must be completed by the end terminator.

This example prints the comment "This is a comment" below the data area on the map.

```
EXAMPLE:   comments
   This is a comment.
   end
```

This example prints the text in a file called "veg.comments" in the current directory.

```
EXAMPLE:   raster  vegetation
   comments  veg.comments
   end
```

defpat   Defines an area fill pattern to be used in setpat instructions.

```
USAGE:   defpat  name
   pattern
   color  #  color
   end
```

An area fill pattern is given a name on the defpat instruction line. This name can then be used in subsequent setpat instructions. The defpat instruction can be used more than once to specify multiple patterns.

The specified pattern is composed of a sequence of numbers (0-9, and blanks, which are equivalent to 0) on one or more lines. The zeros and blanks indicate areas in the pattern where the normal category colors are visible. The other digits, 1-9, indicate pattern pixels and can be assigned any valid color.

The color option specifies a non-zero digit in the pattern, followed by a valid color name. It can be repeated for each of the non-zero digits in the pattern. The default color for all non-zero digits is black unless specified with the color option.

The defpat instruction set must be completed by the end terminator.

In the *p.map.new* input, the defpat instruction must precede any setpat instruction using the specified pattern.

Note: Indented pattern specifications will be interpreted as having leading blanks.

This example creates a black horizontal line pattern called "horiz". Each black line in the pattern would be one pixel wide and would be three pixels from neighboring lines.

```
EXAMPLE:   defpat  horiz
   1
   0
   0
   0
   color  1 black
   end
```

This example creates a green vertical line pattern.

```
EXAMPLE:   defpat  vert

   1000
   color  1 green
   end
```

The following example creates a red diagonal line pattern.

```
EXAMPLE:   defpat  diag
   00001
   0001
   001
   01
   1
   color  1 red
   end
```

This example creates a two-toned tree pattern with orange "trunks" and green "leaves".

```
EXAMPLE:   defpat  tree
  2
  222
   22122
   22 1 22
   1

   2
   222
   22122
   22 1 22
   1
   color  1 orange
   color  2 black
   end
```

end   Terminates input and begins the painting of the map to the output device.

```
USAGE:   end
```

An end instruction completes the entire input to *p.map.new*. It is normally the last line in an input file, but it can be moved forward to eliminate any instructions following its position.

The end instruction for the entire input should not be confused with end terminators that are required with all multiple-line instruction sets.

grid   Overlays a coordinate grid on the output map.

```
USAGE:   grid  spacing
   pattern  notick| tick # #
   masked  [data| nodata| all]
   style  sequence
   width  #
   color  color
   numbers  #  [color] [in| out]
   numbersbg  color| none
   textsize  #
   end
```

The spacing of the grid in geographic coordinate system units must be specified on the first instruction line. The user can specify the overall look of the grid using the pattern parameter. The notick option is for a complete net of intersecting lines. The tick option is for smaller tick marks where grid intervals intersect. The horizontal and vertical lengths, in pixels, must be specified with the tick option.

The user can control the areas covered by the grid by using the masked parameter. With the data option of masked, the grid will be seen over all areas of the map's raster layer except the no-data (category 0) areas. With the nodata option, the grid will be seen only over the no-data areas. The entire grid is seen with the all option to masked.

The grid line style can be specified using a series of 1's and 0's. The 1's represent the visible dashes and the 0's represent gaps between the dashes. The default is solid lines. The width (in pixels) and color of the grid lines can also be specified.

The user can control the placement and look of grid label numbers using the numbers, numbersbg, and textsize parameters. The numbers parameter is used to include grid labels, to specify which labels should be shown (where 1 is every grid label, 2 is every other grid label, etc.), and to specify the label color. It is also used to place the labels inside or outside the current region. The background color behind each label is specified by the numbersbg parameter. The user controls the grid label size using the textsize parameter, in geographic units.

The grid instruction set must be concluded by the end parameter.

When used in a metric location, this example would produce grid ticks every 5000 meters. The purple ticks would have

"arms" ten pixels long and would be visible over the entire map area. The purple grid numbers would be 350 meters high (to scale), inside the current region map area, and have no background color. A grid label would appear every 5000 meters.

```
EXAMPLE:   grid  5000
   pattern  tick 10 10
   masked  all
   width  1
   color  purple
   numbers  1 purple in
   numbersbg  none
   textsize  350
   end
```

The following example would produce black grid lines every 1000 meters. The lines would be visible only in the areas of category 0, and they would be dashed, with one long dash for every short gap. Every other grid label would be shown, each with a white background.

```
EXAMPLE:   grid  1000
   pattern  notick
   masked  nodata
   style  11111100
   width  1
   color  black
   numbers  2 black in
   numbersbg  white
   textsize  200
   end
```

labels   Selects a labels file for output.

USAGE: labels labelfile| list

The labels instruction includes previously prepared label specifications. See manual entry for *p.labels* for correct format of the labels file. The labels file must be accessible via the current mapset search path. The list option is available in keyboard mode. This example would paint labels from a labels file called town.names.

EXAMPLE: labels town.names

legend   Places a user-designed map legend on the output.

```
USAGE:   legend  east north
   legend  x% y%
   height  #
   width  #
   vlen  #
   textcolor  color
   textsize  #
   textwidth  #
   xspace  #
   yspace  #
   background  color
   border  color
   beginrast
   rramp  value| label vertical| horizontal
   catnum  cat description
   end
   beginvect
   vectname  vectormap description
   vecttitle  vectormap
   end
   beginsite
   sitename  sitemap description
   end
   end
```

The location of the upper left corner of the legend must be entered on the first instruction line. The location can be defined either by map coordinates or by percentages of the map area, where 0% 0% is the lower left corner of the map.

The user specifies the height and width of the boxes that will show raster category colors and/or patterns. The length of line segments that will show vector line colors and patterns is specified with the vlen parameter. The user controls horizontal spacing between legend symbols and legend text using xspace. The yspace parameter is used to control vertical spacing between legend symbols. All of these measurements are in pixels.

The user designs the legend text using the textcolor, textsize, and textwidth parameters. Colors are listed in the VALID COLOR NAMES section of this manual entry. The textsize is specified in geographic units. The textwidth is specified in pixels.

The user can specify the color for the background box containing the entire legend. If a color is chosen, underlying map elements are opaque. The user can also specify a border color for the legend box.

The user specifies the symbols to be included in the legend using the beginrast, beginvect, and beginsite parameters. Each of these parameters starts a subsection of the legend instruction that must be completed by an end terminator. These should not be confused with the end terminator for the entire legend instruction set.

If the user simply uses the beginrast parameter followed by end, all categories of the map's raster layer will be shown in individual boxes, and the legend labels will be the corresponding category names in the layer's cats file. The user can include specific categories and optional labels by using one or more catnum lines, each including a category number and the accompanying legend text. If the map's raster layer portrays a continuous range of data, a ramp in the legend might be appropriate. The ramp can be vertical or horizontal, and its accompanying text can be either the smallest and largest category values, or the cats labels associated with the smallest and largest categories. The beginrast subsection must be completed with an end terminator.

Symbols for vector data on the map can be included in the legend by using the beginvect parameter. If the user simply follows beginvect with end, all vector layers in the map will be included in the legend. The user can include specific vector layers in the legend by using the vectname line one or more times, each including a vector layer name and an accompanying description. The vector layer titles as written in dig_cats files can be included as the legend text by using the vecttitle line one or more times. The beginvect subsection must be completed with an end terminator.

Site symbols are included in the legend by using one or more sitename lines in the beginsite subsection. Each line includes the name of the site list and an accompanying description. The beginsite subsection must be completed with an end terminator.

The entire legend instruction set must be completed by an end terminator.

In the *p.map.new* input, the legend instruction can not precede the instructions for any of the map elements that are to be shown in the legend.

This example would produce a legend with five symbols: a point symbol, the colors and patterns for three raster categories, and a line representing one vector layer, in that order. The background of the legend would be white and surrounded by a red border. All text in the legend would be black.

```
EXAMPLE:   legend  589000 4921200
   height  10
   width  20
   vlen  20
   xspace  10
   yspace  7
   textcolor  black
   textsize  250
   textwidth  1
   background  white
   border  red
   beginsite
   sitename  archsites Arch. site
   end
   beginrast
   catnum  4 Sandstone
   catnum  5 Limestone
   catnum  6 Shale
   end
   beginvect
   vectname  roads Road
   end
   end
```

The following example would produce a legend with a vertical ramp showing all the colors in the map's raster layer. The labels of the first and last categories would be included.

```
EXAMPLE:   legend  589000 4921200
   height  10
   width  20
   xspace  10
```

```
  textcolor  black
  textsize  250
  textwidth  1
  background  gray
  border  black
  beginrast
   ramp  label vertical
   end
  end
```

line   Draws a line that is independent of any vector map layer on the output map.

```
USAGE:   line  east north east north
  line  x% y% x% y%
  style  sequence
  color  [# ] color
  width  #
  hcolor  color
  hwidth  #
  masked  [y| n]
  end
```

The beginning and ending points of the line are entered on the main instruction line. These points can be defined either by map coordinates or by using percentages of the geographic region, where 0% 0% is the lower left corner of the map.

The default line style is a continuous, solid line, but the user can specify a dashed line using the style parameter. The style parameter can contain a sequence of digits (0-9) that represent a colored pattern on the desired line. Colors can be assigned to each non-zero digit by using the color parameter multiple times. If the color parameter is used without a specified digit, the named color will be assigned to the entire line. Colors are listed in the VALID COLOR NAMES section in this manual entry.

The user can specify line width in pixels. A highlight color can be assigned with hcolor, and the highlight's width in pixels can be assigned with hwidth. The user can also specify if the line is to be masked by the current mask. (See manual entry for *r.mask* for more information on the mask.)

The line instruction set must be completed by an end terminator.

The line instruction can be used more than once to create multiple lines.

This example would draw a blue line from the point x= 10% y= 80% to the point x= 30% y= 70%. The line would be two pixels wide and would appear even if there is a mask.

```
EXAMPLE:   line  10% 80% 30% 70%
  color  blue
  width  2
  masked  n
  end
```

The following example would draw a line with yellow dashes on a black background.

```
EXAMPLE:   line  605000 4915000 595300 4918200
  style  1111100
  color  1 yellow
```

```
   width  1
   hcolor  black
   hwidth  1
   end
```

outline   Outlines areas of a raster map layer with a specified color.

```
USAGE:   outline
   color  color
   end
```

The outline instruction can be used to place a border around all contiguous groups of same-value cells in a raster map layer. A valid color name can be specified with the optional color parameter. The default color is black. The outline instruction set must be completed by the end terminator, even if the color parameter is not used.

The outline instruction can not precede a raster instruction in a *p.map.new* input file.

The instruction sequence in this example would outline in grey the category areas of a raster map layer called "soils".

```
EXAMPLE:   raster  soils
   outline
   color  grey
   end
```

point   Places a point symbol on the output map.

```
USAGE:   point  east north
   point  x% y%
   icon  iconfile| list
   color  color
   size  #
   masked  [y| n]
   end
```

The user enters a point symbol location on the main instruction line. The location can be defined either by map coordinates or by percentages of the map area, where 0% 0% is the lower left corner of the map.

The icon to be used can be specified with the icon parameter. The user can use any icon in an icons directory within the current mapset search path. Icons can be created using *p.icons* or by simply using a system editor. The default icon is a diamond. The list option for icon is available in keyboard mode.

The user can specify the symbol color. Colors are listed in the VALID COLOR NAMES section of this manual entry.

The icon size is a positive, floating-point scaling factor of the pattern in the icon file. A size of 1 produces an icon with the same number of pixels (at the output device's resolution) as ASCII characters in the icon file.

The user can also specify whether the point symbol is to be masked by the current mask. (See manual entry for *r.mask* for more information on the mask.)

The point instruction set must be completed be an end terminator. Multiple points may be drawn with multiple point instructions.

This example would access an icon file called "box" within the current mapset search path. The red box symbol would be placed at the point E603000, N4921750. The box would have the same number of pixels as characters in the icon file. It would not be masked by the current mask.

```
EXAMPLE:   point  603000 4921750
   icon  box
   color  red
   size  1
   masked  n
   end
```

raster Selects a raster map layer for output.

```
USAGE:   raster  rastermap| list
```

Only one GRASS raster map layer can be specified in a *p.map.new* input file. If no raster map layer is requested, a white background will be produced. The list option is available in keyboard mode.

The raster layer must be accessible within the current mapset search path. In a *p.map.new* input file, the raster instruction must precede these instructions: colortable, outline, setcolor, and setpat. It also must precede any legend instruction set that applies to the raster map layer.

This example would paint a map of the raster map layer soils.
EXAMPLE: raster soils

read Provides input to *p.map.new* from a previously prepared instruction file.

```
USAGE:   read  filename
```

Mapping instructions can be placed in a file and read as input to *p.map.new*. If a certain set of mapping instructions are used in many different maps, they can be placed in one separate file and efficiently accessed by each map's instructions using the read instruction.

Note: *p.map.new* will not search for the file to be read. The file must be in the current directory or a full path needs to be specified on the read instruction line. (Note to /bin/csh users: the tilde [~] path alias will not work with this instruction).

This example reads the ASCII file "pmap.roads" into *p.map.new*.

```
EXAMPLE: read  pmap.roads
```

region Places the outline of a geographic region on the output map.

```
USAGE:   region  regionfile| list
   style  sequence
   color  [# ] color
   width  #
   hcolor  color
   hwidth  #
   masked  [y| n]
   end
```

The user can place the outline of a saved geographic region on the map using region. The named region file must be in a windows directory within the current mapset search path. Geographic region settings can be created and saved using *g.region*. The list option is available in keyboard mode.

The default region outline style is a continuous, solid line, but the user can specify a dashed line using the style parameter. The style parameter can contain a sequence of digits (0-9) that represent a colored pattern on the desired line. Colors can be assigned to each non-zero digit by using the color parameter multiple times. If the color parameter is used without a specified digit, the named color will be assigned to the entire region outline. Colors are listed in the VALID COLOR NAMES section in this manual entry.

The user can specify the region outline width in pixels. A highlight color can be assigned with hcolor, and the highlight's width can be assigned with hwidth. The user can also specify if the outline is to be masked by the current mask. (See manual entry for *r.mask* for more information on the mask.)

The region instruction set must be completed by an end terminator.

The region instruction can be used more than once to show multiple regions.

This example would produce a white outline, two pixels wide, showing the geographic region called "fire.zones".

```
EXAMPLE:   region  fire.zones
   color  white
   width  2
   end
```

scale   Specifies the scale of the hardcopy output map.

```
USAGE:   scale  scale
```

The scale of the output map can be specified in one of several different forms:

relative ratio e.g. 1:25000

number of geographic units per map unit e.g. 1 inch equals 4 miles

absolute width of the printed map e.g. 10 inches

width in number of printed panels e.g. 3 panels

Map inches can be equated with these geographic units: miles, kilometers, and meters. Valid width units are inches, centimeters, and panels. One panel is the single-sheet maximum width available on the hardcopy medium.

The final size of the hardcopy map output is determined by the combination of the specified scale and the current geographic region.

The scale instruction does not affect output to the preview device. If used, the command-line scale parameter overrides the scale instruction.

This example would set the scale of the map to one map unit represents 25,000 geographic units.

EXAMPLE: scale 1:25000

The following example would specify an output map that would be fifteen centimeters wide.

EXAMPLE: scale 15 centimeters

setcolor   Overrides the color assigned to one or more categories of the raster map layer.

USAGE: setcolor cat(s) color

The user can assign a desired color to categories in a raster map layer by using setcolor. Categories are specified on the parameter line before a valid color name. One or more category numbers can appear on the parameter line, separated by commas (with no spaces), or in ranges using hyphens.

The setcolor instruction can be used more than once for assignment of multiple colors.

In the input *p.map.new* file, the setcolor instruction must follow the raster instruction.

Colors are listed in the VALID COLOR NAMES section of this manual entry.

In this example, the color for raster map categories 1 through 3, plus category 5, would be set to green, categories 4, 6, and 8 would be set to blue, and category 7 would be set to red, regardless of their assigned colors in the database.

```
EXAMPLE:   raster  watersheds
   setcolor  1-3,5 green
   setcolor  4,6,8 blue
   setcolor  7 red
```

setpat   Assigns a previously defined pattern to one or more raster map layer categories.

```
USAGE:   setpat  cat(s) name
   setpat  cat(s) #number
   setpat  all| builtin
```

The user can assign a pattern to categories in a raster map layer by using setpat. Categories are specified on the parameter line before the name of a pattern defined earlier using defpat, or before the number signifying a built-in *p.map.new* pattern. One or more category numbers can appear on the parameter line, separated by commas (with no spaces), or in ranges using hyphens.

The built-in patterns are defined in etc/paint/patterns in the compiled GRASS code directory. Each built-in pattern has an assigned number. These numbers can be used following a pound sign (#) on a setpat instruction line. By using the built-in option, each category in a raster map layer can be assigned the correspondingly numbered builtin pattern.

All raster map categories can be assigned the same defined pattern if the all option is used. In this case, only one pattern should be defined within the *p.map.new* mapping instruction file.

The setpat instruction can be used more than once for assignment of multiple patterns.

In the input *p.map.new* file, the setpat instruction must follow the raster instruction, as well as the defpat instruction defining the pattern that is used.

This example assigns a pattern called "vert" to categories 3 and 4 of the raster map layer "vegetation" and a pattern called "tree" to category 10.

```
EXAMPLE:   raster  veg
   setpat  3-4 vert
   setpat  10 tree
```

This example reads a previously prepared ASCII file called horiz.pat containing defpat instructions for creating a black, horizontal pattern called "horiz", and assigns that pattern to category 5 of the raster map layer "soils".

```
EXAMPLE:   raster  soils
   read  horiz.pat
   setpat  5 horiz
```

This example assigns built-in pattern 1 to category 1 of the "soils" raster layer, pattern 2 to category 2, and so on.

```
EXAMPLE:   raster  soils
   setpat  builtin
```

This example assigns built-in pattern 1 to categories 5 through 7 in the "soils" raster map layer, and built-in pattern 2 to categories 10 and 12.

```
EXAMPLE:   raster  soils
   setpat  5-7 # 1
   setpat  10,12 # 2
```

sites   Selects sites data to be placed on the output map.

```
USAGE:   sites  sitemap| list
   icon  iconfile| list
   color  color
   size  #
   desc  [y| n]
   textcolor  color
   textsize  #
   end
```

GRASS sites data can be portrayed on the map using the sites instruction. The user can specify the point symbol to be used, and whether labels are to appear next to the symbols. The sites data must be accessible via the current mapset search path. The list option is available in keyboard mode.

An icon can be specified with the icon parameter. The user can use any icon in an icons directory within the current mapset search path. Icons can be created using *p.icons* or by simply using a system editor. The default icon is a diamond. The list option for icon is available in keyboard mode.

The user can specify the symbol color. Colors are listed in the VALID COLOR NAMES section of this manual entry.

The icon size is a positive, floating-point scaling factor of the pattern in the icon file. A size of 1 produces an icon with the same number of pixels (at the output device's resolution) as ASCII characters in the icon file.

The desc parameter is used to specify whether or not the description of each site in the site_lists file is also to be printed. These labels will appear directly to the right of each site symbol. The user controls the color of the labels using the textcolor parameters. Valid colors are listed in the named colors section of this manual entry. The label size is specified in geographic units using textsize.

The sites instruction set must be completed by the end terminator.

This instruction can be used more than once to portray multiple site lists.

This example would produce point symbols representing the data in a site_lists file called "windmills". An icon called "windmill" would be placed at each site location. These symbols would be two times larger than the size of the icon in the icon file (twice as many pixels as there are characters in the icon file). Descriptions from the sites list file would not be produced in this example.

```
EXAMPLE:  sites  windmills
  icon  windmill
  color  blue
  size  2
  desc  n
  end
```

text   Places text at a user-specified location on the map.

```
USAGE:  text  east north text
  text  x% y%  text
  font  font
  size  #
  color  color| none
  width  #
  hcolor  color| none
  hwidth  #
  ref  reference_point
  rotation  #
  xoffset  #
  yoffset  #
  background  color| none
  opaque  [y| n]
  border  color| none
  end
```

The user specifies where text will be placed by providing map coordinates or percentages of the map area, where 0% 0% is the lower left corner of the map. The text follows the location information on the same instruction line. Multiple lines of text can be specified by annotating the end of a line with \n (e.g., USA\nCERL). Leading blanks an be inserted by preceding the text string with a backslash and the blanks (e.g., text 600000 4920500\See\nWall Drug).

The user can control the appearance of the text, its location, and the appearance of its background box.

The user can specify font (see VALID FONT NAMES in this manual entry), size in geographic units, color (see VALID COLOR NAMES), and width in pixels. The user can further control the text appearance by specifying a highlight color (hcolor) and the width of the highlight color (hwidth).

The text is located at the specified coordinate or percentage pair in relation to a reference point on the text string. This point, specified with the ref parameter, has two parts. The first part refers to a vertical location on the text string. Valid choices are lower, center, and upper. The second part refers to a horizontal location: left, center, and right.

The text string can be rotated at the reference point by using the rotation parameter. The value specified will be the counter-clockwise rotation in degrees from the horizontal.

The xoffset parameter provides finer placement of text by shifting the text a horizontal distance in pixels from the specified easting. The xoffset will shift the text location east if positive and west if negative. The yoffset parameter shifts the text a vertical distance in pixels from the specified northing. The yoffset will shift the location to the south if positive, north if negative.

The user can specify if a background box is present, and what color it should be. The user can also specify whether or not the background box is opaque to other map elements. The color of the border of this box can be specified.

This example would place the text "SPEARFISH LAND COVER" at the coordinates E650000, N7365000. The text would be a total of three pixels wide (one pixel of red text and one pixel of black on each side), have a white background enclosed in a red box, and be 500 meters in size (to scale). The lower left corner of the text would be placed at the coordinates provided. All other map elements would not be seen under the text.

```
EXAMPLE:   text  650000 7365000 SPEARFISH LAND COVER
   font  romand
   color  red
   width  1
   size  500
   ref  lower left
   hcolor  black
   hwidth  1
   background  white
   border  red
   opaque  y
   end
```

vector   Selects a vector map layer for output.

```
USAGE:   vector  vectormap| list
   style  sequence
   color  [# ] color
   width  #
   hcolor  color
   hwidth  #
   masked  [y| n]
   end
```

GRASS vector data can be portrayed on the map using the vector instruction. The name of the vector layer is specified on the first instruction line. The named vector layer must be accessible via the current mapset search path. The list option is available in keyboard mode.

The default vector line style is a continuous, solid line, but the user can specify a dashed line using the style parameter. The style parameter can contain a sequence of digits (0-9) that represent a colored pattern on the vectors. Colors can be assigned to each non-zero digit by using the color parameter multiple times. If the color parameter is used without a specified digit, the named color will be assigned to the entire lengths of the vectors. Colors are listed in the VALID COLOR NAMES section in this manual entry.

The user can specify the vector line width in pixels. A highlight color can be assigned with hcolor, and the highlight's width in pixels can be assigned with hwidth. The user can also specify if the vectors are to be masked by the current mask. (See manual entry for *r.mask* for more information on the mask.)

The vector instruction set must be completed by an end terminator.

The vector instruction can be used more than once to portray multiple vector data layers.

This example would include a vector map layer named "streams" in the output map. These streams would be a total of four pixels wide (two blue pixels with a white outer highlight one pixel wide on each side). The map would not show streams outside of the current mask.

```
EXAMPLE:   vector  streams
   color  blue
   width  2
   hcolor  white
   hwidth  1
   masked  y
   end
```

The following example would portray a vector map layer named "roads". These roads would be two pixels wide and would be dashed blank-black-red (the blank areas would show other map elements under the roads). The roads would be visible inside and outside of the current mask.

```
EXAMPLE:   vector  roads
   width  2
   style  001122
   color  1 black
   color  2 red
   masked  n
   end
```

verbose Sets the amount of feedback sent out by *p.map.new*.

USAGE: verbose 0| 1| 2

A higher value set using verbose results in more feedback. The default is 2.

This example sets the amount of feedback to a minimum.

```
EXAMPLE:   verbose  0
```

VALID COLOR NAMES
The following are the valid color names in *p.map.new*: aqua, cyan, indigo, red, black, gray, magenta, violet, blue, green, orange, white, brown, grey, purple, yellow

any integer from 0 through 124, representing printer color numbers (see *p.colors* manual entry)

VALID FONT NAMES
The following are the valid font names in *p.map.new*: cyrilc, greekcs, italict, romant, gothgbt, greekpromanc, scriptc, gothgrt, greeksromancs, scripts, gothitt, italicc, romand, greekc, italiccs, romans

ICONS VS. PATTERNS

Icons and patterns as used in *p.map.new* are not the same things. Patterns can only be used to cover the extended areas of a raster map layer category. A pattern will repeat above, below and adjacent to itself. Icons are used to represent single points.

Patterns can be defined directly within *p.map.new* using the defpat instruction, while icons are created outside of *p.map.new* using the *p.icons* command or a system editor.

EXAMPLE *p.map.new INPUT FILE*

The following is an example of a *p.map.new* script file. The file has been named "spear.soils". For the purposes of illustration only, the file is shown in two columns. This script file can be entered at the command line.

```
p.map.new input= spear.soils
```

```
raster soils             defpat diag
vector streams           000001
   color blue            00001
   width 2               0001
   hcolor white          001
   hwidth 1              01
   masked y              1
   end                   color 1 red
vector roads             end
   width 2               setpat 4 diag
   style 001122          text 608000 3476004 SPEARFISH SOILS MAP
   color 1 black         color red
   color 2 red           width 2
   masked n              hcolor black
   end                   hwidth 1
labels town.names        background white
region subregion         border red
   color white           size 500
   width 2               ref lower left
   end                   opaque y
grid 10000               end
   color green           line 606969 3423092 616969 3423092
   numbers 2 red         color yellow
   end                   width 2
outline                  opaque yes
   color black           end
   end                   point 40% 60%
colortable y             color purple
comments                 icon diamond
   This is a comment     size 2
   end                   masked n
scale 1:25000            end
setcolor 6,8,9 white     end
setcolor 10 green
```

INTERACTIVE MODE
If the user enters *p.map.new* on the command-line without arguments, a prompting session occurs. Some, but not all, of the non-interactive requests are available in this mode.

SEE ALSO
g.mapsets, g.region, p.chart, p.colors, p.icons, p.labels, p.ppm, p.select, r.mask

AUTHOR
Michael Shapiro, U.S. Army Construction Engineering Research Laboratories
Joo Joo Chia, U.S. Army Construction Engineering Research Laboratories

p.ppm

NAME
p.ppm **-** Reads portable pixmap (ppm) files created by PPM utilities.
(GRASS Paint/Print Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
p.ppm
p.ppm help
p.ppm [-f] [input=name]

DESCRIPTION
This program, *p.ppm*, reads a user-specified portable pixmap (ppm) file and outputs it to the currently selected printer (see *p.select*). The input ppm file should be one that has been created using the PPM utilities developed by Jeff Poskanzer. These utilities can import various image formats (including SUN raster, X Windows pixmaps, and others) into the PPM formats ppm (color pixmaps), pgm (grey scale maps), and pbm (black and white bit-maps).

If the image doesn't fit the output device, it won't get printed. If you want the image printed (but clipped), use the *p.ppm -f* option, or scale the input using ppmscale, or rotate the image using ppmrotate (if it will fit that way -- otherwise you might have to scale it as well). If the image doesn't fit, *p.ppm* will tell you what scaling value to enter to ppmscale that will make it fit.

EXAMPLES
If the user is running GRASS on a SUN machine, the following command could be used to send a monitor screen image to the printer:

```
screendump | rasttoppm | p.ppm
```

If you are running suntools, the user might type:

```
sleep 10; screendump | rasttoppm | ppmrotate 90 | p.ppm
```

The UNIX sleep command allows you time to arrange the frames before the screen dump starts. The ppmrotate is usually needed because the SUN screens are wider than they are long (and wider than 1024 pixels - which is the width of most of our printers).

If you are running X, the user might type:

```
xwd | xwdtoppm | ppmrotate 90 | p.ppm
```

NOTES
This program only supports the ppm binary format (P6).

Maximum color level is 255. If the ppm file has more color levels, use ppmcscale to reduce the number of colors.

No scaling is done. Use ppmscale to change image size.

No rotation is done. Use ppmrotate to rotate the image.

SEE ALSO
d.save, d.savescreen, p.map, p.selec

See also: The PPM utilities ppmrotate (rotates ppm images), ppmscale (scales ppm images for printing), rasttoppm (converts a SUN raster file and xwdtoppm (converts an X Windows dump

The SUN program screendump (dumps the image on the color graphics monitor into a file in SUN raster file format).

The X program xwd (dumps the image in an X window into a file in X window dump [xwd] format).

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

This program uses the PPM utilities developed by Jeff Poskanzer.

p.select

NAME
p.select **-** Selects a device (printer) for GRASS hardcopy output.
(GRASS Hardcopy Output Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
p.select
p.select help
p.select [-*lpq] [painter=name]*

DESCRIPTION
p.select allows the user to select the device for GRASS hardcopy output. The user must select a device before running the other GRASS hardcopy output functions (e.g., *p.map, p.labels, p.chart*).

OPTIONS
Flags:

-*l*	List all available painters.
-*p*	Print name of currently selected painter.
-*q*	Quietly select painter.

Parameter:
painter=name Name of painter to select.
Options: (this will be a list of available hardcopy output devices, and also preview, the user's graphics monitor)

INTERACTIVE MODE
If the user runs *p.select* without specifying program arguments on the command line, the program will prompt the user for the name of a hardcopy output device to select.

NOTE
The options available with *p.select* vary from system to system and depend upon which drivers have been compiled.

SEE ALSO
p.chart, p.labels, p.map

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

p.vrml

NAME
p.vrml **-** output of VRML code from GRASS raster maps

This version only outputs raster maps in VRML 1.0 format. The newer VRML 2.0 format will be more efficient for geographic applications, as it introduces an "ElevationGrid" node so that only the elevation points will have to be written instead of the whole geometry. The vast majority of VRML viewers currently only support VRML 1.0.

GRASS VERSION
4.x

SYNOPSIS
p.vrml elev=name [color=name] output=name

OPTIONS
Parameters:

elev	Name of elevation file.
color	Name of color file.
exag	exaggeration Default: 1.0
output	Name of new VRML file.

"elev" is a raster file to use for surface topography.

"color" (optional) raster file to use for surface color.

"exag" is an exaggeration factor for the vertical dimension.

"output" is a name for the new VRML file. If the extension "wrl" (world) is not present in the name, it will be added.

WARNING
VRML is not well suited for large geometries that can result from even a small geographic region. Most viewers seem to bog down with more than 12,000 polygons, depending on your hardware & specific viewer. Each grid cell results in two polygons, so a reasonable size region would be something less than about 75x75. For improved performance and smaller file size, leave off a color map. Since VRML is ASCII text, gzip works very well to significantly compress file size.

BUGS
Currently the region is transformed to a unit size, so real geographic location is lost. Side effects when working in a lat-lon location are that besides general distortion due to projection, a very small exaggeration factor (on order of .001) must be used to compensate for vertical units expected to be the same as map units.

NOTE
This is a preliminary release of *p.vrml*, a GRASS program to output GRASS data in the format of Virtual Reality Modeling Language (VRML).

For further information about VRML and available viewers for various platforms, see:

http://vag.vrml.org/
http://www.sdsc.edu/vrml/

TO DO
Future plans for this module are to allow draping of sites objects and vector files and using the new sites format available in floating point GRASS to embed WWW links into site objects. It will also be upgraded to support VRML 2.0 and will allow entering multiple preset "views" using the existing GRASS 3d_view file format.

Other possible additions:

- Allow animation of elevation, color, or sites based on user interaction.
- Degradation of the raster to produce TINs for improved performance.

AUTHOR
Bill Brown, U.S. Army Construction Engineering Research Laboratory

paint

NAME
paint **-** Description of hardcopy color output system for GRASS.

GRASS VERSION
4.x, 5.x

INTRODUCTION
The paint system allows the user to produce color hardcopy maps of vector, raster, and sites file data at any scale. For a discussion of the GRASS paint functions, see the manual entries for *p.chart*, *p.colors*, *p.icons, p.labels, p.map, p.screen*, and *p.select*.

PAINT DEVICES
The GRASS paint system supports multiple color printers using a device driver concept. The paint (*p.*) functions listed above send graphics requests to device-dependent paint drivers. These drivers translate the application requests into device-dependent requests to produce hardcopy maps.

INSTALLING A PAINT DRIVER
A number of paint drivers have been distributed with GRASS. The installation of a driver is a 2 step process. The first involves identifying the driver(s) which correspond to printer(s) connected to your system and compiling those drivers. The second involves telling each driver which i/o port it is to use.

1 The source code for the drivers lives in $GISBASE/../src/paint/Drivers (The variable GISBASE refers to the directory in which GRASS is installed on your system.)

The selection and compilation of the drivers is done when GRASS is compiled as a whole.

2 The port configuration is handled using the UNIX ln command.

Each driver expects to send its output to /dev/driver. For example, the tek4695 driver expects to find a tektronix 4695 (or 4696) printer on /dev/tek4695.

Suppose the printer is actually on /dev/tty10. Then, a link named /dev/tek4695 is made to /dev/tty10:
ln /dev/tty10 /dev/tek4695

NOTES
There are 2 drivers which do not use i/o ports. One is the preview driver, which sends its output to the graphics screen instead of a hardcopy printer. This driver is very handy and should definitely be compiled on your system.

The other is the null driver, which is used for debugging purposes and probably should not be compiled on your system. If you compile either of these drivers, you shouldn't create a /dev file for them.

SEE ALSO
p.chart, p.colors, p.icons, p.labels, p.map, p.screen, p.select

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

parser

NAME
GRASS parser interface **-** A set of GRASS routines that standardizes GRASS commands.

GRASS VERSION
4.x, 5.x

DESCRIPTION
A set of routines in GRASS provides a mechanism for command-line parsing. The routines standardize commands that expect command-line arguments. When a user is familiar with the general form of command-line input as defined by the parser, it simplifies the necessity of remembering (or at least guessing) required command-line arguments for any GRASS command.

The parser behaves in one of three ways:

If no command-line arguments are entered by the user, the parser searches for an interactive version of the command. If it is found, control is passed over to this interactive version. If it is not found, the parser prompts the user for programmer-defined options and flags. This prompting conforms to the same standard for every GRASS command that uses the parser routines.

If command-line arguments are entered but do not include all the options and flags that the programmer has defined as required arguments, three things happen: 1) the parser passes an error message to the user indicating which required options and/or flags were missing from the command-line, 2) then the parser displays a complete usage message for that command, and finally 3) the parser cancels execution of the command.

If all necessary options and flags are entered on the command line by the user, the parser executes the command with the given options and flags.

This page was derived from the GRASS 4.1 Programmer's Manual, Command-Line Parsing, which offers detailed information about the GRASS parsing routines.

photo.2image

NAME
photo.2image **-** An imagery function that enables you to mark known reference points (fiducial marks, reseau marks, etc.) on imagery group files, and compute the image-to-photo coordinate transformation parameters. The transformation parameters are required for the GRASS Image Processing Program *i.ortho.photo.* This is not a command in itself, but an option of the command *i.ortho.photo*.
(GRASS Image Processing Program)

GRASS VERSION
4.x, 5.

SYNOPSIS
Selected from *i.ortho.photo*

DESCRIPTION
photo.2image is an imagery function that enables you to mark fiducial or reseau points on an image to be ortho-rectified and then computes the image-to-photo coordinate transformation parameters. The coordinates of the fiducials or reseau marks can be chosen for a camera reference file. During the process of marking reference points with known photo coordinates, you may compute the RMS (root mean square) error for each reference point entered. *photo.2image* does this by calculating the transformation equation (the same one that is calculated in the GRASS program *i.points*), and then plugging these results into an equation for RMS error.

photo.2image offers a zoom option to locate precisely the fiducial or reseau point to be marked on an image.

To run *photo.2image*, a graphics monitor is required.

The procedure for marking fiducial or reseau points, entering known photo-coordinates, and analyzing the RMS error is described below.

The terminal screen displays this message:

use mouse now...

The graphics monitor displays the following screen:

```
 _______________________________________________________
| imagery                     |                          |
|_____________________________|__________________________|
|                             |                          |
|                             |                          |
|                             |                          |
|                             |                          |
|                             |                          |
|                             |                          |
|_____________________________|__________________________|
|                             |                          |
|                             |                          |
|                             |                          |
|                             |                          |
|                             |                          |
|                             |                          |
|                             |                          |
|                             |                          |
|_____________________________|_____________________________|
```

A pop-down menu like that shown below will be superimposed on the left half of the screen:

```
 _____________________________________________
| Double click on cell file to be plotted    |
|  Double click here to cancel               |
|____________________________________________|

 ________________
| Mapset imagery |
|________________|
| gs13.1|  gs21.1|
|_______|________|
| gs14.1|  gs22.2|
|_______|________|
```

Any single file in the imagery group may be used on which to mark points, and you may mark points on more than one file in the imagery group to accumulate the suggested minimum number of 4-8 fiducial or reseau points. The imagery file selected is displayed in the upper left quadrant of the screen.

CAMERA FILE

The camera reference file may be viewed by placing the mouse cross hairs on the words CAMERA. The camera reference file will be displayed in the lower left quadrant of the display:

```
-------------------------------------|
|Camera Reference File               |
-------------------------------------|
|  CAMERA NAME :   camera name       |
|  CAMERA ID   :   L1234             |
|  CAMERA CFL  :   153.021 mm.       |
|  CAMERA XP   :   0.050 mm.         |
|  CAMERA XP   :   0.056 mm.         |
|  number of fid.  4                 |
 -------------------------------------|
|                                    |
|ID   (X PHOTO) mm.  (Y PHOTO) mm.   |
_____________________________________|
|1    -105.023    110.123            |
|2107.987    109.834                 |
|3110.965   -104.329                 |
|4    -103.932   -110.352            |
|____________________________________|
```

The following menu is displayed at the bottom of the graphics display:

```
 ____________________________________________________________________________
|Quit|  Zoom|  Plot Cell|  Camera|  Analyze|    Input->KEYBOARD|  CAMERA FILE|
|____|______|___________|________|_________|___________________|_____________|
```

ZOOM

To magnify the displayed file, you must place the mouse cross hairs on the word ZOOM. The following menu will then be displayed at the bottom of the screen:

```
 ___________________________________________
| Cancel|  Box|  Point|  Select type of ZOOM|
|_______|_____|_______|_____________________|
```

You have the option of identifying the zoom window either by using the mouse to make a box, or by using the mouse to mark the center of the desired window, and adjusting the magnification factor. The terminal screen will display a mouse button menu to guide you in identifying the window. The section of the image within the zoom window will be displayed in the upper right hand quadrant.

MARKING REFERENCE POINTS
To mark the known reference points (fiducial marks, reseau marks, etc.) on the image, you must place the mouse cross hairs on the corresponding location on the image to be marked and press the left hand button on the mouse. A diamond shaped symbol will be marked on the image.

If you wish to use the camera reference file only as a comparative reference, then the keyboard can be chosen as the means to input photo coordinates corresponding to the marked reference points on the image. This is done by placing the mouse cross hairs on the word KEYBOARD and pressing the left button on the mouse.

The following menu is displayed on the graphics terminal:

```
 __________________________________________________________
| Point 1 marked at IMAGE COORDINATES      |               |
| east:  1023.77                          |               |
| north:  506.56                          |               |
|                                         |               |
|                                         |               |
|                                         |               |
|                                         |               |
|                                         |               |
|__________________________________________________________|
| Enter PHOTO COORDINATES as x and y:                       |
|__________________________________________________________|
```

You then enter the known (x,y) photo coordinates, relative to the perspective center, for the reference point marked on the image. If you do not wish to enter a coordinate, simply hit RETURN to continue; the marked reference point will disappear.

If you select the CAMERA FILE option, then reference points marked on the image will be associated with selected photo coordinates from the displayed camera reference file. In this option, when you mark a point on the image, the following menu is displayed on the graphics terminal:

```
-------------------------------------------------------------
CANCEL   Double click on the fiducial mark to be referenced
-------------------------------------------------------------
```

If you would like to select the photo coordinates of a displayed reference mark, this can be accomplished by placing the mouse cross hairs on the reference point to be selected and pressing the left button on the mouse twice. After a reference point is selected from the display, you are prompted with "Look ok? (Y/N)". If you respond with no, the reference point is ignored. If you respond with yes, the following is displayed on the terminal:

```
 ____________________________________________________________
| Point 1 marked at IMAGE COORDINATES        |               |
| east:  1023.77                            |               |
| north: 1065.41                            |               |
|                                           |               |
|                                           |            |
| Point 1 referenced at PHOTO COORDINATES   |            |
| X: -105.023                               |            |
| Y:  110.122                               |            |
|                                           |            |
|                                           |            |
|                                           |            |
|                                           |            |
|                                           |            |
|________________________________________________________|
| use mouse now...                                       |
|________________________________________________________|
```

The photo coordinates and the corresponding image coordinates are automatically saved in the photo_points file associated with the imagery group.

ANALYZE
After a number of points have been marked (4 to 8), you can check the RMS error of the points marked on the image. This is done by placing the mouse cross hairs on the word ANALYZE at the bottom of the monitor. An error report resembling the one shown below is superimposed on the monitor:

```
 ________________________________________________________
|    error  image  photo                                 |
|                                                        |
|#rowcol    photoeast    north  xy                      |
|10.0-0.9    1.01048.5   -144.8   -105.023    110.122    |
|20.4 1.0    1.32153.1   -567.2    107.987    109.834    |
|3   -1.2 -0.5.61452.8   -476.5    110.965   -104.329    |
|41.1 0.5    1.31034.0   -109.2   -103.932   -110.352    |
|                                                        |
|________________________________________________________|
|    overall   rms    error:   4.46                      |
|________________________________________________________|
```

The following menu then appears at the bottom of the monitor:

```
 _________________________________________________________________
|DONE|  PRINT|  FILE|  Double click on point to be included/excluded|
|____|_______|______|_____________________________________________|
```

The RMS error for the image is given under the column titled "error" and subtitled "row" and "col". In the above report, point number 1 is 0.0 rows and -0.9 columns from the predicted location calculated from the transformation equation. The RMS error for the photo coordinates is listed under the heading "photo". This is the RMS error for the x and y coordinates of the photo coordinates but it is presented in the table using one general value. The overall RMS error is displayed at the bottom of the screen in millimeters. Points that create high RMS error are displayed in red on the monitor (represented here in italics).

The location of the point marked on the imagery group file is given under the heading "image" and the subheadings "east" and "north". The location of the point in the photo coordinates is given under the heading "photo" and the subheadings "x" and "y". If you would like to exclude or include a point, this can be accomplished by placing the mouse cross hairs on the point number to be included (if the point is absent) or excluded (if the point is displayed) and pressing the left button on the mouse twice. When a point is excluded, it is not afterwards included in the calculation of the RMS error, or included in the final transformation matrix. However, it can be retrieved within *photo.2image* at any time by double clicking with the mouse as described above.

QUIT
To end the *photo.2image* program place the mouse cross hairs on the word QUIT; the marked reference points (including coordinates) will be saved.

NOTES
A good rule of thumb is to mark at least 4 to 8 points, which are evenly distributed over the perimeter of the imagery group file in order to obtain an accurate transformation equation for the rectification process. The RMS error may increase with more points added, but the transformation equation will be more accurate.

An RMS error of less than or equal to approximately one resolution unit (pixel) for the image being rectified is generally considered acceptable.

SEE ALSO
i.ortho.photo[2], photo.camera[2], photo.2target[2], photo.init[2], photo.rectify[2]

AUTHOR
Mike Baba DBA Systems, Inc.

photo.2target

NAME
photo.2target **-** An imagery function that enables you to mark control points on an image to be ortho-rectified, and then input the coordinates of each control point for determination of ortho-rectification parameters. The ortho-rectification parameters are required as input for the GRASS program photo.rectify. This is not a command in itself, but an option to the command *i.ortho.photo*.
(GRASS Imagery Processing Tool)

GRASS VERSION
4.x, 5.x

SYNOPSIS
Selected from *i.ortho.photo*

DESCRIPTION
photo.2target is an imagery function that enables you to mark control points on an image to be ortho-rectified and then input the coordinates of each point for calculation of rectification parameters.

Rectification is the mapping of an image from one coordinate system to another. The geometry of an image extracted into a GRASS LOCATION having an x,y coordinate system is not planimetric. To create a planimetric image, that is, to convert the x,y coordinate system into a standard coordinate system (for example, the UTM coordinate system or the State Plane coordinate system), points from a map having the standard coordinates must be associated with the same points on the image to be rectified.

The ortho-rectification parameters are computed in two phases. The first phase computes a transformation matrix between image (row,col) coordinates and photo (x,y) coordinates relative to the perspective center. The transformation matrix is computed explicitly in the option photo.2image.

The second phase, *photo.2target*, enables you to mark control points on the image and then input the standard coordinates (Easting,Northing, and elevation) to determine the parameters for a three dimensional projective transformation.

photo.init may be run before running *photo.2target* to modify the initial camera exposure station parameters, and/or modify the standard deviation of these parameters. *photo.init* is not required but generally is helpful.

During the process of marking control points and entering standard coordinates, you may compute the RMS (root mean square) error for each control point entered. *photo.2target* does this by calculating the transformation equation, and then plugging these results into an equation for RMS error.

photo.2target offers a zoom option to locate precisely the point to be marked on an image. This program also offers you the option of acquiring standard coordinates for a marked point from a map layer in the target database, or from a digitizer.

To run *photo.2target*, a graphics monitor is required.

The procedure for marking points, entering coordinates, and calculating RMS error is described below.

The terminal screen displays the following message:

```
use mouse now...
```

The graphics monitor displays the following screen:

```
 ____________________________________________________________
| imagery  filename  (mag)     |  target  filename  (mag)     |
|______________________________|______________________________|
|                              |                              |
|                              |                              |
|                              |                              |
|                              |                              |
|                              |                              |
|                              |                              |
|______________________________|______________________________|
|                              |                              |
|                              |                              |
|                              |                              |
|                              |                              |
|                              |                              |
|______________________________|______________________________|
| QUIT ZOOM PLOTCELL ANALYZE   |                              |
|______________________________|______________________________|
```

A pop-down menu like that shown below will be superimposed on the left half of the screen:

```
 _____________________________________________
| Double click on cell file to be plotted    |
|  Double click here to cancel               |
|____________________________________________|

 ________________
|   Mapset demo  |
|________________|
| gs13.1|  gs14.1|
|_______|________|
| gs21.1|  gs22.2|
|_______|________|
```

Any single file in the imagery group may be used to mark points, and you can mark points on more than one file in the imagery group to accumulate the 12 points suggested minimum. Any file in the imagery group can be rectified (using *photo.rectify*) based on the rectification parameters computed from these control points.

The imagery file you select is displayed in the upper left quadrant of the screen.

ZOOM

To magnify the displayed file, you must place the mouse cross hairs on the word ZOOM. The following menu will then be displayed at the bottom of the screen:

```
 ______________________________________________
| Cancel|  Box|  Point|  Select type of ZOOM   |
|_______|_____|_______|________________________
```

You may identifying the zoom window either by using the mouse to make a box, or by using the mouse to mark the center of the window and entering a magnification factor. The terminal screen will display a mouse button menu to guide you in identifying the window.

MARKING POINTS

To mark the points on the image that correspond to the points on a standard coordinate system map, you must place the mouse cross hairs on the corresponding location on the image to be marked and press the left hand button on the mouse. A diamond shaped symbol will be marked on the image. The terminal will display the following menu:

```
 ____________________________________________________________
| Point 1 marked at IMAGE COORDINATES |                      |
| IMAGE X:   1023.77                  |                      |
| IMAGE Y:  -164.41                   |                      |
|                                     |                      |
|                                     |                      |
|                                     |                      |
|                                     |                      |
|                                     |                      |
|____________________________________________________________|
| Enter CONTROL COORDINATES as east,north,elevation:         |
|____________________________________________________________|
```

You then enter the easting, northing, and elevation for the point marked on the image. If you wish not to enter a coordinate, simply hit RETURN to return control to the mouse; the marked point then disappears.

PLOT CELL
In addition to acquiring control points from a standard map, you have the option of acquiring the points from a cell-map in the target database. The database map is displayed by placing the mouse cross hairs on the words PLOT CELL. The following line is then displayed at the bottom of the monitor:

```
 ________________________________________________
| Cancel|  Indicate which side should be plotted|
|_______|_______________________________________|
```

Which side of the monitor is to be plotted is indicated by placing the mouse cross hairs on the half of the monitor screen that you would like to use, and pressing the left mouse button. The following pop-down menu will be superimposed on the half of the screen that was chosen:

```
 _________________________________________
| Double click on cell file to be plotted|
|  Double click here to cancel           |
|_________________________________________|

 ______________________________
|    Mapset demo               |
|______________________________|
| tm.rectified |               |
|______________|_______________|
| tm.classified|               |
|______________________________|
|  Mapset PERMANENT            |
|______________________________|
| elevation    |  geology      |
|______________|_______________|
| slope        |  soils        |
|______________|_______________|
| aspect       |               |
|______________|_______________|
| roads        |               |
|______________|_______________|
| streams      |               |
|______________|_______________|
| airfields    |               |
|______________|_______________|
```

After the map is displayed the following message appears at the bottom of the monitor:

```
 ______________________________________
| input method -->|  keyboard|  screen|
|_________________|__________|________|
```

If you wish to use the plotted map only as a comparative reference, the keyboard can be chosen as the means to input coordinates corresponding to the marked control points. This is done by placing the mouse cross hairs on the word KEYBOARD and pressing the left button on the mouse.

If you select the SCREEN option, points marked on the image will automatically be associated with the coordinates from the corresponding points on the target database map, and a corresponding elevation from the cell-file selected for elevation data. In this option, when you mark a point on the image, the following menu is displayed on the terminal:

```
 ____________________________________________________
| Point 5 marked at IMAGE COORDINATES |              |
| IMAGE X: 1023.77                    |              |
| IMAGE Y: -164.41                    |              |
|                                     |              |
|                                     |              |
| Control Point location              |              |
| East: 679132.57                     |              |
| North:    4351080.67                |              |
| Elevation:   1010.00                |              |
|                                     |              |
|                                     |              |
|                                     |              |
|___________________________________________________|
| use mouse now...                                   |
|___________________________________________________|
```

The coordinates for the target database map are automatically saved as the coordinates corresponding to the marked control point on the image.

ANALYZE
After a number of points have been marked (4 to 7), you can check the RMS error of the points marked on the image. This is done by placing the mouse cross hairs on the word ANALYZE at the bottom of the monitor. An error report resembling that shown below is superimposed on the monitor:

```
 __________________________________________________________________
|    error                                                          |
|#  east    north    target    east    northeast northelev.         |
|__________________________________________________________________|
|1  0.0-0.9  1.048.54.879132.5   351080.6   010.0                   |
|2  0.4 1.0  1.353.17.284314.7   399001.4   239.3                   |
|3 -1.2-0.5   .652.86.567841.4   457682.8   209.5                   |
|4  1.1 0.5  1.334.09.277573.8   352626.4   432.5                   |
|5 -2.714.0 14.248.6    -144.9   79132.6   351080.7   985.0         |
|                                                                   |
|__________________________________________________________________|
|overall   rms error:   4.46                                        |
|__________________________________________________________________|
```

The following menu then appears at the bottom of the monitor:

```
 ___________________________________________________________________
| DONE|  PRINT FILE|   Double click on point to be included/excluded|
|_____|____________|________________________________________________|
```

The RMS error for the image is given under the column titled "error" and subtitled "east" and "north". In the above report, point number 1 is 0.0 and -0.9 meters (east and north) from the predicted location calculated from the transformation equation. The RMS error for the target map is listed under the heading "target". This is the RMS error for the east and north coordinates of the target map but it is presented in the table using one general value. The overall RMS error is displayed at the bottom of the screen in meters. Points that create high RMS error are displayed in red on the monitor (represented here in italics).

The image coordinates of the point marked on the imagery group file is given under the heading "image" and the subheadings "east" and "north". The location of the control point in the target database is given under the heading "control" and the subheadings "east","north", and "elev". If you would like to exclude or include a control point, this can be accomplished by placing the mouse cross hairs on the control point number to be included (if the point is absent) or excluded (if the point is displayed) and pressing the left button on the mouse twice. When a point is excluded, it is not afterwards included in the calculation of the RMS error, or included in the final rectification parameters. However, it can be retrieved within *photo.2target* at any time by double clicking with the mouse as described above.

QUIT
To end the *photo.2target* program place the mouse cross hairs on the word QUIT; the marked control points (including coordinates) will be saved.

NOTES
During the course of marking control points and computing the ortho-rectification parameters, a matrix inversion error may occur. This is caused by trying to invert a non-singular normal equation matrix. When this situation arises, the status of all previously selected control points are modified, the control points are excluded. Running *photo.init* for the selected imagery group with accurate camera exposure station parameters should remedy the situation. The excluded control points may again be included as described in the section ANALYZE.

A good rule of thumb is to mark at least 12 to 15 points, which are evenly distributed over the entire imagery group file in order to obtain an accurate transformation parameters for the rectification process. The RMS error may increase with more points added, but the transformation parameters will be more accurate over the entire image.

An RMS error of less than or equal to approximately one resolution unit (pixel) for the image being rectified is generally considered acceptable.

SEE ALSO
i.ortho.photo[2], photo.camera[2], photo.2image[2], photo.init[2], photo.rectify[2]

AUTHOR
Mike Baba DBA Systems, Inc.

photo.camera

NAME
photo.camera **-** An imagery function that creates or modifies entries in a camera reference file. This is not a command in itself, but an option to the command *i.ortho.photo*.
(GRASS Image Processing Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
photo.camera option must be selected from GRASS Image Processing Program *i.ortho.photo*.

DESCRIPTION
photo.camera creates or modifies entries in a camera reference file. For ortho-photo rectification, a camera reference file is required for computation of scanned image to photo-coordinate transformation parameters.

The first prompt in the program will ask you for the name of the camera reference file to be created or modified. You may create a new camera reference file by entering a new name, or modify an existing camera reference file by entering the name of an existing camera file.

After entering the camera file name, following menu is displayed:

```
Please provide the following information

CAMERA NAME:camera name______
CAMERA IDENTIFICATION:identification___
CALIBRATED FOCAL LENGTH mm.:_________________
POINT OF SYMMETRY (X)   mm.:_________________
POINT OF SYMMETRY (Y)   mm.:_________________
MAXIMUM NUMBER OF FIDUCIALS:_________________

    AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
   (OR <Ctrl-C> TO CANCEL)
```

The camera name and identification describe the camera reference file. The calibrated focal length and the point of symmetry are used in computing the photo-to-target transformation parameters. These values should be entered from the camera calibration report (usually available from the photograph supplier). The maximum number of fiducials will determine the number of fiducial or reseau coordinate pairs to be entered below.

You are then ask to enter the X and Y photo coordinates of each fiducial as follows:

```
Please provide the following information

Fid# FID ID XY


1__  _____0.0___    0.0___
2__  _____0.0___    0.0___
3__  _____0.0___    0.0___
4__  _____0.0___    0.0___
5__  _____0.0___    0.0___
6__  _____0.0___    0.0___
7__  _____0.0___    0.0___
8__  _____0.0___    0.0___
9__  _____0.0___    0.0___
10_  _____0.0___    0.0___
```

```
        next:  end__

            AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
           (OR <Ctrl-C> TO CANCEL)
```

The input display is repeated until the number of MAXIMUM FIDUCIALS is reached. On this screen you should enter the fiducial or ressuae photo-coordinates as given in the camera calibration report. The X, and Y coordinates are in millimeters from the principle point.

SEE ALSO
i.ortho.photo[2], photo.2image[2], photo.2target[2], photo.init[2]

AUTHOR
Mike Baba DBA Systems, Inc.

photo.init

NAME
photo.init **-** An imagery function that creates or modifies the optional entries in a camera initial exposure station file for an imagery group. This is not a command in itself, but an option to the command *i.ortho.photo.*
(GRASS Image Processing Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
Selected from *i.ortho.photo*

DESCRIPTION
photo.init creates or modifies entries in a camera initial exposure station file for imagery group referenced by a sub-block. These entries include: the (XC,YC,ZC) UTM approximate coordinates of the camera exposure station; initial roll, pitch, and yaw angles (in degrees) of the cameras attitude; and the a priori standard deviations for these parameters. During the imagery program, *photo.rectify,* the initial camera exposure station file is used for computation of the ortho-rectification parameters. If no initial camera exposure station file exist, the default values are computed from the control points file created in *photo.2target*.

The following menu is displayed:

```
Please provide the following information

INITIAL XC: Meters __________
INITIAL YC: Meters __________
INITIAL ZC: Meters __________
INITIAL omega (roll) degrees:__________
INITIAL phi  (pitch) degrees:__________
INITIAL kappa  (yaw) degrees:__________

Standard Deviation XC: Meters__________
Standard Deviation YC: Meters__________
Standard Deviation ZC: Meters__________
Std. Dev. omega (roll) degrees:   __________
Std. Dev. phi  (pitch) degrees:   __________
Std. Dev. kappa  (yaw) degrees:   __________

    AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
   (OR <Ctrl-C> TO CANCEL)
```

The INITIAL values for (XC,YC,ZC) are expressed in UTM coordinates, and represent an approximation for the location of the camera at the time of exposure.

The INITIAL values for (omega,phi,kappa) are expressed in degrees, and represent an approximation for the cameras attitude at the time of exposure.

The standard deviations for (XC,YC,ZC) are expressed in meters, and are used as a priori values for the standard deviations used in computation of the ortho rectification parameters.

The standard deviations for (omega,phi,kappa) are expressed in degrees, and are used as a priori values for the standard deviations used in computation of the ortho rectification parameters.

SEE ALSO
i.ortho.photo[2], photo.camera[2], photo.2image[2], photo.2target[2], photo.rectify[2]

AUTHOR
Mike Baba DBA Systems, Inc.

photo.rectify

NAME
photo.rectify **-** An imagery function that rectifies an image by computing a coordinate transformation for each cell in the image using parameters computed by the GRASS programs *photo.2image* and *photo.2target*. This is not a command in itself, but an option to the command *i.ortho.photo*.
(GRASS Image Processing Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
Selected from *i.ortho.photo*

DESCRIPTION
photo.rectify rectifies an image by using the image to photo coordinate transformation matrix created by *photo.2image* and the rectification parameters created by *photo.2target*. Rectification is the process by which the geometry of an image is made planimetric. This is accomplished by mapping an image from one coordinate system to another. In *photo.rectify* the parameters computed by *photo.2image* and *photo.2target* are used in equations to convert x,y image coordinates to standard map coordinates for each pixel in the image. The result is an image with a standard map coordinate system, compensated for relief distortions and photographic tilt. Upon completion of the program the rectified image is deposited in a previously targeted GRASS LOCATION.

You are asked to select the file(s) within the imagery group
to be rectified:

```
Please select the file(s) to rectify by naming an output file

gs13.1 in demo   gs13.orect...
gs14.1 in demo   .............
gs21.1 in demo   .............
gs22.1 in demo   .............

(enter list by any name to get a list of existing cell files)

    AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
   (OR <Ctrl-C> TO CANCEL)
```

More than one file may be rectified at a time. Each file should have a unique output file name.

The next prompt asks you to select one of two windows:

```
Please select one of the following options
1.   Use the current window in the target location
2.   Determine the smallest window which covers the image
>
```

photo.rectify will only rectify that portion of the image that occurs within the chosen window. Only that portion will be relocated in the target database. It is therefore important to check the current window in the target LOCATION if choice number one is selected.

photo.rectify will run in the background and notify you by mail when it is finished. The process may take an hour or more depending on the size of the image, the number files, and the size and resolution of the selected window.

The rectified image will be located in the target LOCATION when the program is completed. The original unrectified files are not modified or removed.

SEE ALSO
i.ortho.photo[2], photo.camera[2], photo.2image[2], photo.2target[2], photo.init[2]

AUTHOR
Mike Baba DBA Systems, Inc.

ps.icon

NAME
ps.icon **-** Creates and modifies icons for use with *ps.map*.
(GRASS Hardcopy PostScript Output Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
ps.icon

DESCRIPTION
This program allows the user to create and maintain icons which are used by *ps.map* to depict sites.

FILES
Icon files created by the user are stored under $LOCATION/ps_icon.

SEE ALSO
ps.map

AUTHOR
Paul Carlson, USDA, SCS, NHQ-CGIS

ps.map

NAME
ps.map **-** Hardcopy PostScript map output utility.
(GRASS Hardcopy PostScript Output Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
ps.map
ps.map help
ps.map [r] [input=name] [scale=mapscale] [copies=n] output=name

DESCRIPTION
ps.map produces an output file containing a PostScript program to produce hardcopy map products on your system's PostScript output device. Output can include a raster map, any number of vector overlays, site data, text labels, and other spatial data.

This program has 2 distinct modes of operation. The command-line mode requires the user to prepare a file of mapping instructions describing the various spatial and textual information to be printed prior to running *ps.map*. The interactive mode (i.e., no command-line arguments) will prompt the user for items to be mapped and does not require the user to prepare a file of instructions.

The command line flag is:

-*r* Rotate plot 90 degrees.

The command-line parameters are:

input=name File containing mapping instructions. (or enter input=- to enter from keyboard).
These instructions are described in detail below.

scale=mapscale Scale of the output map, e.g. 1:25000
Default: 1 panel

This parameter is provided as a convenience. It is identical to the scale mapping instruction described below.

output=name Name of output the file to contain the PostScript program.

Note: the user must select a PostScript output device using *ps.select* before running *ps.map*.

MAPPING INSTRUCTIONS
The mapping instructions allow the user to specify various spatial data to be plotted. These instructions are normally prepared in a regular text file using a system editor. Some instructions are single line instructions while others are multiple line. Multiple line instructions consist of the main instruction followed by a subsection of one or more additional instructions.

Instructions that may be included in the subsection under several different main instructions are:

where x y The top left corner of the bounding box of the item to be plotted is located x inches from the left edge of the paper and y inches from the top edge of the paper. If x is less than or equal to zero, the default horizontal location is used. If y is less than or equal to zero, the default vertical location is used.

font font name The name of the PostScript font. The default is Helvetica.

fontsize font size The size of the PostScript font in 1/72 inch. The default is 10.

colortable Prints the color table for the raster map layer anywhere on the page.

```
USAGE:   colortable [y|n]
 where x y
 width table width
 cols table columns
 font font name
 fontsize font size
 color text color
 end
```

The color table will display the colors for each raster map layer category value and the category label. To get a color table, you must have previously requested a raster map layer. The default location for the colortable is immediately below any other map legend information, starting at the left margin. The default text color is black. Omitting the colortable instruction would result in no color table. Note: Be careful about asking for color tables for raster map layers which have many categories, such as elevation. This could result in the printing of an extremely long color table!!

This example would print a color table immediately below any other map legend information, starting at the left margin.

```
EXAMPLE:   colortable y
```

comments Prints comments anywhere on the page.

```
USAGE:   comments commentfile
 where x y
 font font name
 fontsize font size
 color text color
 end
```

The default location is immediately below the last item printed, starting at the left margin. The default text color is black.

This example prints in blue whatever is in the file veg.comments starting at 1.5 inches from the left edge of the page and 7.25 inches from the top of the page, using a 15/72 inch Helvetica Bold font.

```
EXAMPLE:   raster vegetation
 comments veg.comments
   where 1.5 7.25
   font Helvetica Bold
   fontsize 15
   color blue
   end
```

Presumably, the file veg.comments contain comments pertaining to the raster map layer vegetation, such as "This map was created by classifying a LANDSAT TM image".

copies Specifies the number of copies to be printed.

USAGE: copies n

Each page will be printed n times.

greyrast Selects a raster map layer for output in shades of grey.

USAGE: greyrast mapname|list

For each *ps.map* run, only one raster map layer can be requested (using either the greyrast or the raster instruction).

grid Overlays a coordinate grid onto the output map.

```
USAGE:   grid spacing
 color color
 numbers # [color]
 font font name
 fontsize font size
 end
```

The spacing of the grid is given (in the geographic coordinate system units) on the main instruction line. The subsection instructions allow the user to specify the color of the grid lines, whether coordinate numbers should appear on the grid lines, and if they should appear every grid line (1), every other grid line (2), etc., and what color the numbers should be. The defaults are black grid lines, unnumbered.

This example would overlay a green grid with a spacing of 10000 meters (for a metered database, like UTM) onto the output map. Alternate grid lines would be numbered with red numbers.

```
EXAMPLE:   grid 10000
   color green
   numbers 2 red
   end
```

header Prints the map header above the map.

```
USAGE:   header
 file header file
 font font name
 fontsize font size
 color text color
 end
```

If the header instruction or the fileR sub-instruction is absent, the header will consist of the map title and location, each centered on the page above the map. The default text color is black.

This example prints (in red) whatever is in the file soils.hdr above the map, using a 20/72 inch Courier font.

```
EXAMPLE:   header
```

```
   file soils.hdr
   font Courier
   fontsize 20
   color red
   end
```

labels   Selects a labels file for output (see manual entry for *p.labels*).

```
USAGE:   labels  labelfile|list
 font font name
 end
```

This example would paint labels from the labels file called town.names. Presumably, these labels would indicate the names of towns on the map.

```
EXAMPLE:   labels town.names
```

line   Draws lines on the output map.

```
USAGE:   line east north east north
 line x% y% x% y%
 color color
 width #
 masked [y|n]
 end
```

The beginning and ending points of the line are entered on the main instruction. These points can be defined either by map coordinates or by using percentages of the geographic region. The user may also specify line color, width in pixels, and if the line is to be masked by the current mask. (See manual entry for *r.mask* for more information on the mask.)

This example would draw a yellow line from the point x=10% y=80% to the point x=30% y=70%. This line would be 2 pixels wide and would appear even if there is a mask.

```
EXAMPLE:   line 10% 80% 30% 70%
   color yellow
   width 2
   masked n
   end
```

Of course, multiple lines may be drawn with multiple line instructions.

mapinfo   Prints the portion of the map legend containing the scale, grid and region information, on or below the map.

```
USAGE:   mapinfo
 where x y
 font font name
 fontsize font size
 color text color
 end
```

The default location is immediately below the map, starting at the left edge of the map. The default text color is black.

This example prints (in brown) the scale, grid and region information immediately below the map and starting 1.5 inches from the left edge of the page, using a 12/72 inch Courier font.

```
EXAMPLE:   mapinfo
   where 1.5 0
   font Courier
   fontsize 12
   color brown
   end
```

maploc   Positions the map on the page.

```
USAGE:   maploc  x y [width height]
```

The upper left corner of the map will be positioned x inches from the left edge of the page and y inches from the top of the page. If widthR and height (in inches) are present, the map will be rescaled, if necessary, to fit.

This example positions the upper left corner of the map 2.0 inches from the left edge and 3.5 inches from the top edge of the map.

```
EXAMPLE:   maploc 2.0 3.5
```

outline   Outlines the areas of a raster map layer with a specified color.

```
USAGE:   outline
 color  color
 end
```

Distinct areas of the raster map will be separated from each other visually by drawing a border (or outline) in the specified color (default: black). Note: it is important the user enter the instruction end even if a color is not chosen. (It is hoped that in the future the outline of a different raster map layer other than the one currently being painted may be placed on the map.)

This example would outline the category areas of the soils raster map layer in grey.

```
EXAMPLE:   raster soils
 outline
 color grey
 end
```

point   Places additional points or icons on the output map.

```
USAGE:   point east north
 point x% y%
 color color
 icon iconfile|list
 size #
 masked [y|n]
 end
```

The point location is entered in the main instruction line by giving either the map coordinates or by using percentages of the geographic region. The user may also specify the point color, the icon file to be used to represent the point location (see the manual entry for *ps.icons*), the size of the icon as a multiple of the size of the pattern in the icon file (default is 1.0, approximately equivalent to a 10 point character), and whether the point is to be masked by the current mask. (See manual entry for *r.mask* for more information on the mask.)

This example would place a purple diamond (from icon file diamond) at the point (E456000 N7890000). This diamond would be the approximately the size of a 15 point character and would not be masked by the current mask.

```
EXAMPLE:   point 456000 7890000
   color purple
   icon diamond
   size 1.5
   masked n
   end
```

Of course, multiple points may be drawn with multiple point instructions.

psfile   Copies a file containing PostScript commands into the output file.

Note: *ps.map* will not search for this file. The user must be in the correct directory or specify the full path on the psfile instruction. (Note to /bin/csh users: ~ won't work with this instruction).

USAGE: psfile filename

This example copies the file "logo.ps" into the output file.

EXAMPLE: psfile logo.ps

raster   Selects a raster map layer for output.

USAGE: raster mapname|list

For each *ps.map* run, only one raster map layer can be requested. If no raster map layer is requested, a completely white map will be produced. It can be useful to select no raster map layer in order to provide a white background for vector images.

This example would paint a map of the raster map layer soils.

EXAMPLE: raster soils

read   Provides *ps.map* with a previously prepared input stream.

USAGE: read previously prepared UNIX file

Mapping instructions can be placed into a file and read into *ps.map*.

Note: *ps.map* will not search for this file. The user must be in the correct directory or specify the full path on the read instruction. (Note to /bin/csh users: ~ won't work with this instruction).

This example reads the UNIX file pmap.roads into *ps.map*. This file may contain all the *ps.map* instructions for placing the vector map layer roads onto the output map.

EXAMPLE: read pmap.roads

The user may have created this file because this vector map layer is particularly useful for many *ps.map* outputs. By using the read option, the user need not enter all the input for the vector instruction, but simply read the previously prepared file with the correct instructions.

region Places the outline of a smaller geographic region on the output.

```
USAGE:   region regionfile|list
 color color
 width #
 end
```

Geographic region settings are created and saved using *g.region*. The *ps.map* region option can be used to show an outline of a smaller region which was printed on a separate run of *ps.map* on other user-created maps.

The user can specify the color and the width (in pixel units) of the outline. The default is a black border of one pixel width.

This example would place a white outline, 2 pixels wide, of the geographic region called fire.zones onto the output map. This geographic region would have been created and saved using *g.region*.

```
EXAMPLE:   region fire.zones
   color white
   width 2
   end
```

scale Selects a scale for the output map.

```
USAGE:   scale scale
```

The scale can be selected either as:

1. a relative ratio, e.g. 1:25000;
2. an absolute width of the printed map, e.g. 10 inches;
3.the number of printed paper panels, e.g. 3 panels (at the present time, only 1 panel is supported);
4.the number of miles per inch, e.g. 1 inch equals 4 miles.

This example would set the scale of the map to 1 unit = 25000 units.

```
EXAMPLE:   scale 1:25000
```

setcolor Overrides the color assigned to one or more categories of the raster map layer.

```
USAGE:   setcolor cat(s) color
```

This example would set the color for categories 2,5 and 8 of the raster map layer watersheds to white and category 10 to green. (NOTE: no spaces are inserted between the category values.)

```
EXAMPLE:   raster watersheds
 setcolor 2,5,8 white
 setcolor 10 green
```

Of course, setcolor can be requested more than once to override the default color for additional categories. More than one category can be changed for each request by listing all the category values separated by commas (but with no spaces).

sites Selects sites data to be placed on the output map (see manual entry for *s.menu*).

```
USAGE:   sites sitemap|list
 color color
 icon iconfile|list
 size #
 desc [y|n]
 font font name
 end
```

The user may specify the color of the sites (see section on NAMED COLORS below); the icon to be used to represent the presence of a site (see the manual entry for *p.icons*); the size of the icon (number of times larger than the size it is in the icon file); and whether or not the description associated with each site is also to be printed. If the description is to be printed, the font name may be specified. The size of the font is proportional to the icon size. This example would paint a sites map with blue windmills (from an icon file created by the user using the *p.icons* GRASS command) placed at all windmill locations (from a sites list). These windmills would be two times larger than the size of the icon in the icon file and have descriptions from the sites list file printed beside them.

```
EXAMPLE:   sites windmills
   color blue
   icon windmill
   size 2
   desc y
   end
```

text Places text on the map.

```
USAGE:   text  east north text
 text  x% y% text
 font fontname
 color color|none
 width #
 hcolor color|none
 hwidth #
 background color|none
 border color|none
 size #
 ref reference point
 xoffset #
 yoffset #
 opaque [y|n]
 end
```

The user specifies where the text will be placed by providing map coordinates or percentages of the geographic region map. The text follows these coordinates on the same instruction line. More than one line of text can be specified by annotating the end of a line with \n (e.g. USA\nCERL).

The user can then specify various text features:

font: cyrilc gothgbt gothgrt gothitt greekc greekcs greekp greeks italicc italiccs italict romanc romancs romand romans romant scriptc scripts (The default font is romans);

color (see NAMED COLORS);

width of the lines used to draw the text (to make thicker letters);

size as the vertical height of the letters in meters on the ground (text size will grow or shrink depending on the scale at which the map is painted);

the highlight color (hcolor) and the width of the highlight color (hwidth);

the text-enclosing-box background color; the text box border color;

ref. This reference point specifies the text handle - what part of the text should be placed on the location specified by the map coordinates. Reference points can refer to: [lower|upper|center] [left|right|center] of the text to be printed;

yoffset, which provides finer placement of text by shifting the text a vertical distance in pixels from the specified north. The vertical offset will shift the location to the south if positive, north if negative;

xoffset, which shifts the text a horizontal distance in pixels from the specified east. The horizontal offset will shift the location east if positive, west if negative; whether or not the text should be opaque to vectors. Entering no to the opaque option will allow the user to see any vectors which go through the text's background box. Otherwise, they will end at the box's edge.

This example would place the text SPEARFISH LAND COVER at the coordinates E650000 N7365000. The text would be a total of 3 pixels wide (2 pixels of red text and 1 pixel black highlight), have a white background enclosed in a red box, and be 500 meters in size. The lower right corner of the text would be centered over the coordinates provided. All vectors on the map would stop at the border of this text.

```
EXAMPLE:   text 650000 7365000 SPEARFISH LAND COVER
   font romand
   color red
   width 2
   hcolor black
   hwidth 1
   background white
   border red
   size 500
   ref lower left
   opaque y
   end
```

vector Selects a vector map layer for output.

```
USAGE:   vector vectormap|list
 color color
 width #
 hcolor color
 hwidth #
 masked [y|n]
 style  0-9
 end
```

The user can specify the color of the vectors; the width of the vectors lines in pixels; the highlight color (hcolor) for the vector lines; the width of the highlight color (hwidth) in pixels; whether or not the raster map layer is to be masked by the current mask (see manual entry *r.mask* for more information on the mask); and the line style. The line style allows the vectors to be dashed in different patterns. This is done by typing a series of numbers (0's and 1's) in a desired sequence or pattern. Blanks and non- digit characters are recognized as 0's. Using 0 would allow the colors of the raster map layer (or the background color if no raster map layer was selected) to show through.

This example would paint a map of the vector map layer named streams. These streams would be a total of 3 pixels wide (the inner two pixels blue and the outer highlight pixel white). The map would not show streams outside of the current mask.

```
EXAMPLE:   vector streams
   color blue
   width 2
   hcolor white
   hwidth 1
   masked y
   end
```

verbose Changes the amount of talking *ps.map* will do.

USAGE: verbose [0|1|2]

A higher value implies more chatter. The default is 2. This example sets the amount of chatter to a minimum.

EXAMPLE: verbose 0

vlegend Prints the portion of the map legend containing the vector information, on or below the map.

```
USAGE:   vlegend
 where x y
 font font name
 fontsize font size
 end
```

The default location is immediately below the legend containing the scale, grid and region information, starting at the left edge of the map. If the where instruction is present and y is less than or equal to zero, the vector legend will be positioned immediately below the map, starting x inches from the left edge of the page.

This example prints the vector legend immediately below the map and starting 4.5 inches from the left edge of the page, using a 12/72 inch Helvetica font.

```
EXAMPLE:   vlegend
   where 4.5 0
   font Courier
   fontsize 12
   end
```

end   Terminates input and begin painting the map.

USAGE: end

NAMED COLORS
The following are the colors that are accepted by *ps.map*:

aqua black blue brown cyan gray green grey indigo magenta orange purple red violet white yellow

EXAMPLE *ps.map* INPUT FILE
The following is an example of a *ps.map* script file. The file has been named spear.soils. For the purposes of illustration, the file is in two columns. This script file can be entered at the command line:

ps.map input=spear.soils output=soils.ps

```
raster soils              defpat diag
vector streams            000001
   color blue             00001
   width 2                0001
   hcolor white           001
   hwidth 1               01
   masked y               1
   end                    color 1 red
vector roads              end
   width 2                setpat 4 diag
   style 001122           text 608000 3476004 SPEARFISH SOILS MAP
   color 1 black          color red
   color 2 red            width 2
   masked n               hcolor black
   end                    hwidth 1
labels town.names         background white
region subregion          border red
   color white            size 500
   width 2                ref lower left
   end                    opaque y
grid 10000                end
   color green            line 606969 3423092 616969 3423092
   numbers 2 red          color yellow
   end                    width 2
outline                   opaque yes
   color black            end
   end                    point 40% 60%
colortable y              color purple
comments                  icon diamond
   This is a comment      size 2
   end                    masked n
scale 1:25000             end
setcolor 6,8,9 white      end
setcolor 10 green
```

INTERACTIVE MODE
If the user simply enters *ps.map* without arguments, then a simple prompting session occurs. Some, but not all of the non-interactive requests are available at this level.

SEE ALSO
ps.icons, ps.select

AUTHOR
Paul Carlson, USDA, SCS, NHQ-CGIS

ps.select

NAME
ps.select **-** Selects a PostScript device for GRASS hardcopy output.
(GRASS Hardcopy PostScript Output Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
ps.select
ps.select help
ps.select [-lpq] [painter=name]

DESCRIPTION
ps.select allows the user to select the PostScript device for GRASS hardcopy output. If the user has not selected a device before running *ps.map*, the default device configuration (shown under NOTES below, will be used.

OPTIONS
Flags:

-*l*	List all available PostScript painters.
-*p*	Print name of currently selected PostScript painter.
-*q*	Quietly select PostScript painter.

Parameter:
painter=name Name of PostScript painter to select.
Options: (this will be a list of available hardcopy output PostScript devices)

INTERACTIVE MODE
If the user runs *ps.select* without specifying program arguments on the command line, the program will prompt the user for the name of a hardcopy output PostScript device to select.

NOTES
Each PostScript device must have a device description file in the $GISBASE/etc/paint/psdevices directory. The name of the device description file is used as the name of the PostScript painter. A device description file contains the following information (default values are shown):

```
level: 2
page width: 8.5
page height: 11.0
top margin: 0.5
bottom margin: 0.5
left margin: 0.25
right margin: 0.25
resolution: 75
```

where level is the PostScript level supported by the device (1 or 2); page width and page height are the paper dimensions (in inches); top margin, bottom margin, left margin and right margin are the unprintable edges of the page (in inches); and resolution is the effective resolution (in pixels per inch).

SEE ALSO
ps.map

AUTHOR
Paul Carlson, USDA, SCS, NHQ-CGIS