GRASS Shell Scripts

This are not all scripts, please look at the scripts directory ($GISBASE/scripts/). These scripts are nice examples how to create batch jobs.

3d.view.sh	rgb.hsv.sh
blend.sh	shade.clr.sh
bug.report.sh	shade.rel.sh
d.rast.leg.sh	show.color.sh
d.zoom.last.sh	show.fonts.sh
dcorrelate.sh	slide.show.sh
grass.logo.sh	split.sh
hsv.rgb.sh	tig.rim.sh
i.oif	tiger.info.sh

```
d.rast.leg d.rast.rescale g.html2man i.image.mosaic
i.oif i.spectral i.tasscap  m.statistics ps.add.pagesize
r.edge.dig r.out.bil r.out.geotiff r.reclass.area
r.regression.line s.in.gps s.out.gps tiger.info
v.cutter.attr v.in.gps v.line2area v.xfig
```

Other Commands

help home, database, display, drivers, general, grid3d, imagery, import, misc, models, paint, photo, postscript, raster, scripts, sites, vector

NAME

3d.view.sh − Displays several 3−dimensional views of a landscape on the user's graphics monitor.
(GRASS Shell Script)

SYNOPSIS

3d.view.sh
3d.view.sh help
3d.view.sh file=*mapname* **ef**=*mapname* **vh**=*viewing_height* **sv**=*sink_value* **exag**=*exag* **lf**=*line_frequency* **back**=*background_color*

DESCRIPTION

3d.view.sh is a Bourne shell (sh(1)) script that displays several 3−dimensional views of a landscape on the user's graphics monitor. It erases the graphics monitor and then prepares it for the display of nine equally−sized frames. The user−specified raster map layer (given by **file**=*name*) is displayed using *d.rast* in the middle frame. The remaining frames are then used to display 3−d perspective views. The top middle panel is a view from the north, the top right from the north−east, the right from the east, and so on. Each is drawn with a call to the *d.3d* program. The viewing angles are calculated automatically.

OPTIONS

If options are not stated on the command line, default values will be used. These values are listed under Parameters, below.

Parameters:

file=*mapname*
: Name of raster map layer to be displayed.
Default: *elevation*

ef=*mapname*
: Name of raster map layer whose category values will supply the elevation values used to generate 3−d perspective views.
Default: *elevation*

vh=*viewing_height*
: Height (in meters) of the location from which scenes will be viewed.
Default: 30000

sv=*sink_value*
: Sink factor value, causing the image to be displayed lower, or higher, on the graphics screen.
Default: 0

exag=*vertical_exaggeration*

Vertical exaggeration factor of the values in the elevation file.
Default: 3

lf=*line_frequency*
Contour intervals at which vector grid lines will be drawn, in meters.
Default: 20

back=*background_color*
Color of the background of the display frames.
Options: red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, white, and black
Default: black

NOTES

In the *spearfish* sample data base, the user must specify a viewing height when running *3d.view.sh*. Note also that the raster elevation map layers in the PERMANENT mapset under spearfish are named *elevation.dem* and *elevation.dma*.

This program will not prompt the user for inputs; if the user types **3d.view.sh** without program arguments on the command line, default values will be used.

FILES

This program is simply a shell script. Users are encouraged to make their own shell scripts using similar techniques. See **`$GISBASE/scripts/3d.view.sh`**.

SEE ALSO

d.3d
d.rast

AUTHOR

James Westervelt, U.S.Army Construction Engineering Research Laboratory

Last changed: $Date: 2002/01/25 05:45:32 $

NAME

blend.sh − Combines the red, green, and blue color components of two raster map layers.
(GRASS Shell Script)

SYNOPSIS

blend.sh file1 file2 perc outbase

DESCRIPTION

blend.sh is a Bourne shell (sh(1)) script that extracts the red (R), green (G), and blue (B) color components from each of two raster map layers, and creates three new raster map layers whose category values respectively represent the combined red, combined blue, and combined green color values from the two input layers. Category values in each of the output map layers will fall within the range of 0 − 255.

The R,G,B values from the two input map layers (*file1* and *file2*) are not simply added together, but are instead combined by a user−named percentage (*perc*) of the R,G,B values in *file1*. Specifically, *blend.sh* executes three *r.mapcalc* statements that:

1. convert the R,G,B values in *file1* and *file2* to the range 0 − 255;
2. multiply the R, G, and B values in *file1* by a user−named percentage (*perc*);
3. multiply the R, G, and B values in *file2* by (100 − *perc*)%;
4. create three new raster map layers, whose category values represent the summed R, summed G, or summed B values resulting from (2) and (3). Resulting R, G, and B values will respectively be stored in three new raster map layers named *outbase*.r, *outbase*.g and *outbase*.b.

OPTIONS

This program runs non−interactively; the user must state all parameter values on the command line.

Parameters:

file1
: Name of a first raster map layer, whose R, G, and B color components will be combined with those of the second raster map layer (*file2*) named. The percent value (*perc*) given will apply to *file1*.

file2
: Name of a second raster map layer, whose color components will be combined with those of *file1*. The percent value (*perc*) given will apply to the R,G,B values in *file1*. The R, G, and B values in *file2* will be multiplied by (100 − *perc*)%.

perc

Percentage or amount of the color contribution in terms of color intensity. This value is multiplied by the R,G,B values in *file1*.

outbase

The root name assigned to each of the three output files created. A suffix is added to each file name, indicating which hold the red, green, and blue color values.

NOTES

blend.sh executes three *r.mapcalc* statements:

r.mapcalc "*outbase*.r = r#*file1* * *.perc* + (1.0 − *.perc*) * r#*file2*"
r.mapcalc "*outbase*.g = g#*file1* * *.perc* + (1.0 − *.perc*) * g#*file2*"
r.mapcalc "*outbase*.b = b#*file1* * *.perc* + (1.0 − *.perc*) * b#*file2*"

It uses the # operator to separately extract the red, green, and blue components in the named raster map layers, essentially allowing color separates to be made.

EXAMPLE

Typing the following at the command line:

blend.sh aspect elevation 40 elev.asp

will create three new raster map layers named *elev.asp.r*, *elev.asp.g*, and *elev.asp.b*, that, respectively, contain 40% of the red, green, and blue components of the *elevation* map layer and contain 60% of the red, green, and blue components of the *aspect* map layer.

FILES

This program is simply a shell script. Users are encouraged to make their own shell scripts using similar techniques. See **`$GISBASE/scripts/blend.sh`**.

SEE ALSO

r.colors,
r.mapcalc

AUTHOR

Dave Gerdes, U.S.Army Construction Engineering Research Laboratory

Last changed: $Date: 2002/01/25 05:45:32 $

NAME

bug.report.sh − A mechanism for writing, storing, and e−mailing bug reports on GRASS 5 commands. *(GRASS Shell Script)*

SYNOPSIS

bug.report.sh *module_name* [*module_arguments*]

DESCRIPTION

bug.report.sh is a Bourne shell (sh(1)) script which, when given a GRASS 5 module name, allows the user to complete a bug report for that program. Completed bug reports can be e−mailed to the GRASS development Team (GRASS Bug report system), saved to a file in the user's home directory, or discarded.

OPTIONS

This program is not interactive; the user must specify the name of a 5.x module on the command line. Program arguments can optionally be entered by the user.

Parameters:

module_name
: The name of an existing GRASS version 5 program tested by the user.

module_argument(s)
: Program arguments (parameters and/or flags) for the *module_name* tested by the user. The user should enter whatever command arguments were actually run when the program bug occurred.

EXAMPLE

For example, if the user wished to complete a bug report on *v.to.rast* after running the program, the user might then type:

bug.report.sh v.to.rast

A standard bug report form would then be displayed on the user's text terminal, containing the user's current GRASS region settings, and the machine name, GRASS data base, location, and mapset, on which the user is currently running GRASS. Program parameters and flag settings, and the command entered by the user on the command line, are also automatically entered on the bug report form. The user is put into a text editor and expected to enter additional information describing the nature of the bug found or the results of program testing.

The user is then asked whether the completed bug report is to be:

1 − mailed to grass−bugs@intevation.de
2 − added to the file **`grass.bugs`** in the user's home directory
3 − both 1 and 2
4 − thrown away

NOTES

This program prints whatever region settings, GRASS data base, location, and mapset are current when the user runs *bug.report.sh*. The user is therefore advised to run *bug.report.sh* immediately after experiencing a program bug, to ensure that the settings current when the bug occurred are reported on the bug report form.

Note, that the $EDITOR environment variable is used to define the editor. The "vi" editor is the default editor.

FILES

This shell script is stored under the **`$GISBASE/scripts`** directory on the user's system. The user is encouraged to examine the shell script commands stored in this directory and to produce similar scripts for their own use.

AUTHOR

James Westervelt, U.S.Army Construction Engineering Research Laboratory

Last changed: $Date: 2002/01/25 05:45:32 $

NAME

dcorrelate.sh − Graphically displays the correlation among from two to four raster map layers in the active frame on the graphics monitor.
(GRASS Shell Script)

SYNOPSIS

dcorrelate.sh *layer1 layer2* [*layer3* [*layer4*]]

DESCRIPTION

dcorrelate.sh is a C−shell (csh(1)) script that graphically displays the results of an *r.stats* run on two raster map layers. This shell script is useful for highlighting the correlation (or lack of it) among data layers (scattergram).

The results are displayed in the active display frame on the user's graphics monitor. *dcorrelate.sh* erases the active frame before displaying results.

OPTIONS

Parameters:

layer1 layer2 [layer3 [layer4]]
The names of from two to four existing raster map layers to be included in the correlation.

NOTES

This is a shell script that uses *r.stats* and the UNIX *awk* command to calculate the correlation among data layers, and uses *d.text* and *d.graph* to display the results.

If three or four map layers are specified, the correlation among each combination of two data layers is displayed.

This command is written for */bin/csh*. If your system doesn't support this shell, don't install this script.

FILES

This program is simply a shell script. Users are encouraged to make their own shell script programs using similar techniques. See **`$GISBASE/scripts/dcorrelate.sh`**.

SEE ALSO

The UNIX *awk* command

d.text, *d.graph*, *r.coin*, *r.stats*

AUTHOR

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

Last changed: $Date: 2002/06/16 15:29:18 $

NAME

d.rast.leg.sh − Displays a raster map and its legend on a graphics window.
(GRASS Shell Script)

SYNOPSIS

d.rast.leg.sh
d.rast.leg.sh help
d.rast.leg.sh *rast_map* [*num_of_lines*]

DESCRIPTION

d.rast.leg.sh is a UNIX Bourne shell macro which clears the entire screen, divides it into a main (left) and a minor (right) frames, and then display a raster map in the main frame and the map legend in the minor frame. The main frame remains active when the program finishs.

OPTIONS

The user can run the program interactively or non−interactively.

Parameters:

rast_map
: A raster map to be displayed.

num_of_lines
: Number of lines to appear in the legend. If this number is not given, the legend frame will display as many lines as number of categories in the map, otherwise, it will display the first **num_of_lines** minus 1 categories with the rest being truncated.

NOTES

The user may adjust the **num_of_lines** parameter or the size of graphics window to get an appropriate result.

FILES

See the file *d.rast.leg.sh* under **`$GISBASE/scripts`**.

SEE ALSO

d.legend
d.rast

AUTHORS

Jianping Xu,
Scott Madry,
Rutgers University

Last changed: $Date: 2002/01/25 05:45:33 $

NAME

d.zoom.last.sh − Allows the user to change the current geographic region settings interactively and tracks the last region settings. (An addition to the GRASS display program *d.zoom*).
(GRASS Shell Script)

SYNOPSIS

d.zoom.last.sh

DESCRIPTION

d.zoom.last.sh is a UNIX Bourne shell macro which allows the user to interactively adjust the settings of the current geographic region using a pointing device such as a mouse, and automatically saves the last region settings. The last region can be restored by a GRASS command:

g.region region=last

or

g.region last

This script and the above restoring command function as an 'undo' in editing.

NOTES

Only the last region is saved. When zooming multiple times, regions before the last are NOT available. Region−file 'last' is reserved for *d.zoom.last.sh*.

FILES

See the file **`$GISBASE/scripts/d.zoom.last.sh`**.

SEE ALSO

d.zoom
g.region.

AUTHOR

Jianping Xu,
John Bognar,
Rutgers University

Last changed: $Date: 2002/01/25 05:45:33 $

NAME

grass.logo.sh − Displays a GRASS/Army Corps of Engineers logo in the active display frame on the graphics monitor.
(GRASS Shell Script)

SYNOPSIS

grass.logo.sh

DESCRIPTION

grass.logo.sh is primarily a demonstration of the GRASS *d.graph* program in a UNIX Bourne shell macro that generates the U.S. Army Corps of Engineers logo. Users are encouraged to generate their own unique logos by writing similar macro shell scripts. Examine the contents of the input file called *grass.logo.sh* located in the GRASS shell script command directory (**`$GISBASE/scripts`**) to see how the GRASS logo was generated using *d.graph* graphics commands. The coordinates for this logo were taken from a drawing done on graph paper.

To view the graphics described by this file use *d.graph* or *d.mapgraph*, making sure that a copy of this file is either in your current directory or is given by its full path name.

NOTES

grass.logo.sh, like *d.rast*, will overwrite (not overlay) whatever display appears in the active graphics frame.

This program requires no command line arguments.

SEE ALSO

d.font
d.graph
d.mapgraph
d.rast

AUTHOR

James Westervelt, U.S.Army Construction Engineering Research Laboratory

Last changed: $Date: 2002/01/25 05:45:33 $

NAME

hsv.rgb.sh − Converts HSV (hue, saturation, and value) cell values to RGB (red, green, and blue) values. *(GRASS Shell Script)*

SYNOPSIS

hsv.rgb.sh *file1 file2 file3*

DESCRIPTION

hsv.rgb.sh is a Bourne shell (sh(1)) program that converts HSV to RGB using *r.mapcalc*. The Foley and Van Dam algorithm is the basis for this program. Input must be three raster files − each file supplying the HSV values. Three new raster files are created representing RGB.

NOTES

Do not use the same names for input and output.

FILES

This program is simply a shell script stored in the file *hsv.rgb.sh* under the **`$GISBASE/scripts`** directory. Users are encouraged to make their own shell script programs using similar techniques.

SEE ALSO

rgb.hsv.sh

AUTHOR

James Westervelt, U.S.Army Construction Engineering Research Laboratory

Last changed: $Date: 2002/01/25 05:45:33 $

NAME

mapcalculator.sh − TCLTK−Frontend for r.mapcalc

SYNOPSIS

mapcalculator.sh [help=[*help,man***]] [expert=***expert***] A_MAP=[***map***] B_MAP=[***map***]...**

DESCRIPTION

mapcalculator.sh is a bash−script usally called via **mapcalculator** in TCLTKGRASS. It creates a r.mapcalc call from the parameters you choose in mapcalculator. If you want to use mapcalc on the commandline you should use the original r.mapcalc command.
In **TCLTKGRASS** you can choose up to six maps [*entries A − F*] to be included in the calculation. Then you have to enter a formula and a name for the resulting raster map. You won't receive a warning, if you choose the name of an allready existing raster map, but you can protect your files from overwriting by activating the appropriate checkbox.
You can create a calculation formula in the field formula like this:
A+C or (more complex:) *exp(A+C)+(B−2)*7.*
A, B, C, etc. are your choosen raster maps. You must not insert the output file:

Right: *A+B*
Wrong: *newfile = A+ B*

Use no blanks! Blanks are inserted by the script to devide the map names from the rest of the formula.

The script won't check the syntax or logic of the formula and you will only receive the error messages from r.mapcalc, if anything went wrong. For details on creating a formula see the *r.mapcalc* manual page (man r.mapcalc).

BUGS

It is not possible to choose maps from different mapsets with the *raster* button. But you can choose maps from different mapsets by entering the name *map@mapset* in any of the map fields *[A−E]*.

You may have difficulties to calculate formulas containing *neighborhood* modifier. The only way to use *neighborhood* modifier is entring them directly into the formula field:

```
A+B*map_with_modifier\[x,y\]
```

where map_with_modifier is the name of your map. The name of your map should not contain any uppercase *A − F*, because they will be interpreted as raster map entries. Don't forget to mask the brackets with

backslashes, or you will receive errormessages form **TCLTKGRASS.**

SEE ALSO

r.mapcalc

AUTHOR

R. Brunzema

Last changed: $Date: 2002/03/01 00:08:31 $

NAME

r.dma2ll.sh − Import USGS 1:250,000 DEM files (DMA formatted) into a lat−lon region
(GRASS Shell Script)

SYNOPSIS

r.dma2ll.sh *dem_file raster_name*

DESCRIPTION

r.dma2ll.sh is a UNIX Bourne shell to ease importing of DMA−formatted USGS 1:250,000−scale Digital Elevation Models (DEM) data into a raster map in a lat−lon database. r.dma2ll.sh extracts the boundary coordinates from the DEM file automatically.

OPTIONS

None.

Parameters:

dem_file
: The name of the DMA−formatted DEM file to be imported.

raster_name
: The name of the resulting raster map. The new raster map is created in the currently selected mapset and database.

NOTES

Little error checking is performed. A temporary file is created and deleted during import. If **$TMPDIR** is set, the temporary file will be created in the directory specified. Otherwise, the temporary file is created in **/tmp**.

Environment variables **GISBASE** , **GISDBASE** , **LOCATION_NAME** and **MAPSET** are used to specify the GRASS database, location, and mapset.

Standard UNIX programs *dd* , *sed* and *awk* are assumed.

FILES

DMA−formatted DEM data files can be obtained from:

http://edcwww.cr.usgs.gov/doc/edchome/ndcdb/ndcdb.html
http://edcftp.cr.usgs.gov/pub/data/DEM/250/
ftp://edcftp.cr.usgs.gov/pub/data/DEM/250/

SEE ALSO

m.dmaUSGSread
m.rot90
m.in.bin

AUTHOR

Tom Poindexter, March, 1999
tpoindex@nyx.net

Last changed: $Date: 2002/01/25 05:45:34 $

NAME

rgb.hsv.sh − Converts RGB (red, green, and blue) cell values to HSV (hue, saturation, value) values. *(GRASS Shell Script)"*

SYNOPSIS

rgb.hsv.sh *file1 file2 file3*

DESCRIPTION

rgb.hsv.sh is a Bourne shell (sh(1)) program which converts RGB values to HSV using *r.mapcalc*. The Foley and Van Dam algorithm is the basis for the program. Input must be three raster files − each file supplying the RGB values. Three new raster files are created representing HSV.

NOTES

Do not use the same names for input and output.

FILES

This program is simply a shell script stored under the **`$GISBASE/scripts`** directory. The user is encouraged to examine the shell script programs stored here and to produce other such programs.

SEE ALSO

hsv.rgb.sh

AUTHOR

James Westervelt, U.S.Army Construction Engineering Research Laboratory

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

shade.clr.sh − Creates a color shaded relief map based on current resolution settings and sun altitude and color azimuths values entered by the user.
(GRASS Shell Script)

SYNOPSIS

shade.clr.sh
shade.clr.sh help
shade.clr.sh [**altitude**=*value*] [**r_azimuth**=*value*] [**g_azimuth**=*value*] [**b_azimuth**=*value*] [**elevation**=*name*] [**shade**=*value*|*m*|*f*]

DESCRIPTION

shade.clr.sh is a Bourne shell (sh(1)) script that creates a colored raster shaded relief map based on current resolution settings and on sun altitude and color light azimuth values entered by the user. The new shaded relief map is named <*elevation*>.*shade* and stored in the user's current mapset.

If no parameters are provided on startup, this program is interactive; thus if the user enters the command:

shade.clr.sh

The program then prompts the user to enter values for:

1. The **altitude** of the sun in degrees above the horizon (a value between 0 and 90 degrees), and
2. The **azimuth** of the red, green, and blue lights in degrees to the east of north (a value between −1 and 360 degrees).
3. The name of a raster map layer whose cell category values are to provide **elevation** values for the shaded relief map. Typically, this would be a map layer of elevation; however, any raster map layer can be named.
4. The scaling parameter, which compensates for a different horizontal **scale** than vertical scale. For example, when a latitude−longitude projection is used with an elevation map measured in meters. If 'scale' is a number then the ewres and nsres are multiplied by that scale to calculate the shading. If 'scale' is the letter M (either case) the number of meters in a degree of latitude is used as the scale. If 'scale' is the letter F (either case) then the number of feet in a degree is used. The script scales latitude and longitude equally, so it's only approximately right, but for shading its close enough. It makes the difference between a usable and unusable shade.

Specifically, *shade.clr.sh* executes the following *r.mapcalc* statement:

```
r.mapcalc << EOF
```

```
      $ELEV.shade = eval( \\
      x=($elev[-1,-1] + 2.*$elev[0,-1] + $elev[1,-1] \\
      -$elev[-1,1] - 2.*$elev[0,1] - $elev[1,1])/(8.*ewres()*$shade) , \\
      y=($elev[-1,-1] + 2.*$elev[-1,0] + $elev[-1,1] \\
      -$elev[1,-1] - 2.*$elev[1,0] - $elev[1,1])/(8.* nsres()*$shade) , \\
      slope=90.-atan(sqrt(x*x + y*y)), \\
      a=round(atan(x,y)), \\
      a=if(isnull(a),1,a), \\
      aspect=if(x!=0||y!=0,if(a,a,360.)), \\
      rang = sin($alt)*sin(slope) + cos($alt)*cos(slope) *
      cos($raz-aspect),\\
      red = int(if(rang < 0.,0.,4.9*rang)), \\
      gang = sin($alt)*sin(slope) + cos($alt)*cos(slope) *
      cos($gaz-aspect),\\
      green = int(if(gang < 0.,0.,4.9*gang)), \\
      bang = sin($alt)*sin(slope) + cos($alt)*cos(slope) *
      cos($baz-aspect), \\
      blue = int(if(bang < 0.,0.,4.9*bang)), \\
      1. + red + 5. * green + 25. * blue )
EOF
```

Refer to the manual entry for *r.mapcalc* for an explanation of the filtering syntax shown in the above expression. See, for example, the section on "The Neighborhood Modifier".

shade.clr.sh then runs *r.colors* to assign a color table to the new shaded relief map *<elevation>.shade*.

FILES

This program is simply a shell script. Users are encouraged to make their own shell scripts using similar techniques. See **`$GISBASE/scripts/shade.clr.sh`**.

SEE ALSO

An Algebra for GIS and Image Processing, by Michael Shapiro and Jim Westervelt, U.S. Army Construction Engineering Research Laboratory (March/1991) (get from GRASS web site).

shade.rel.sh
blend.sh
g.ask
g.region
r.colors
r.mapcalc

AUTHOR

Jim Westervelt, U.S. Army Construction Engineering Research Laboratory

Last changed: $Date: 2003/08/26 07:03:47 $

NAME

shade.rel.sh − Creates a shaded relief map based on current resolution settings and sun altitude and azimuth values entered by the user.
(GRASS Shell Script)

SYNOPSIS

shade.rel.sh
shade.rel.sh help
shade.rel.sh [**altitude**=*value*] [**azimuth**=*value*] [**elevation**=*name*] [**shade**=*value*|*m*|*f*]

DESCRIPTION

shade.rel.sh is a Bourne shell (sh(1)) script that creates a raster shaded relief map based on current resolution settings and on sun altitude and azimuth values entered by the user. The new shaded relief map is named *<elevation>.shade* and stored in the user's current mapset. The map is assigned a grey−scale color table.

If no parameters are provided on startup, this program is interactive; thus if the user enters the command:

shade.rel.sh

The program then prompts the user to enter values for:

1. The **altitude** of the sun in degrees above the horizon (a value between 0 and 90 degrees), and
2. The **azimuth** of the sun in degrees to the east of north (a value between −1 and 360 degrees).
3. The name of a raster map layer whose cell category values are to provide **elevation** values for the shaded relief map. Typically, this would be a map layer of elevation; however, any raster map layer can be named.
4. The scaling parameter, which compensates for a different horizontal **scale** than vertical scale. For example, when a latitude−longitude projection is used with an elevation map measured in meters. If 'scale' is a number then the ewres and nsres are multiplied by that scale to calculate the shading. If 'scale' is the letter M (either case) the number of meters in a degree of latitude is used as the scale. If 'scale' is the letter F (either case) then the number of feet in a degree is used. The script scales latitude and longitude equally, so it's only approximately right, but for shading its close enough. It makes the difference between a usable and unusable shade.

Specifically, *shade.rel.sh* executes the following *r.mapcalc* statement:

```
r.mapcalc << EOF
      $ELEV.shade = eval( \\
      x=($elev[−1,−1] + 2*$elev[0,−1] + $elev[1,−1] \\
      −$elev[−1,1] − 2*$elev[0,1] − $elev[1,1])/(8.*ewres()*$scale) , \\
```

```
      y=($elev[-1,-1] + 2*$elev[-1,0] + $elev[-1,1] \\
      -$elev[1,-1] - 2*$elev[1,0] - $elev[1,1])/(8.*nsres()*$scale) , \\
      slope=90.-atan(sqrt(x*x + y*y)), \\
      a=round(atan(x,y)), \\
      a=if(isnull(a),1,a), \\
      aspect=if(x!=0||y!=0,if(a,a,360.)), \\
      cang = sin($alt)*sin(slope) + cos($alt)*cos(slope) *
      cos($az-aspect), \\
      if(cang < 0.,0.,100.*cang), \\
      if(isnull(cang), null(), 100.*cang))
EOF
```

Refer to the manual entry for *r.mapcalc* for an explanation of the filtering syntax shown in the above expression. See, for example, the section on "The Neighborhood Modifier".

shade.rel.sh then runs *r.colors* to assign a grey−scale color table to the new shaded relief map *<elevation>.shade*, by executing the command:

r.colors **shade color**=grey

FILES

This program is simply a shell script. Users are encouraged to make their own shell scripts using similar techniques. See **`$GISBASE/scripts/shade.rel.sh`**.

SEE ALSO

An Algebra for GIS and Image Processing, by Michael Shapiro and Jim Westervelt, U.S. Army Construction Engineering Research Laboratory (March/1991) (get from GRASS web site).

shade.clr.sh
blend.sh
g.ask
g.region
r.colors
r.mapcalc

AUTHOR

Jim Westervelt, U.S. Army Construction Engineering Research Laboratory

Last changed: $Date: 2003/08/26 07:03:47 $

NAME

show.color.sh − Displays and names available primary colors used by GRASS programs, in frames on the graphics monitor.
(GRASS Shell Script)

SYNOPSIS

show.color.sh

DESCRIPTION

show.color.sh is a UNIX Bourne shell macro that displays and names available primary colors used by GRASS programs in frames on the graphics monitor. Available colors are: *red*, *orange*, *yellow*, *green*, *blue*, *indigo*, *violet*, *white*, *black*, *gray*, *brown*, *magenta*, and *aqua*.

No program arguments are required to run this program.

NOTES

The full monitor screen is made the active frame after this program ends.

This macro is located in **`$GISBASE/scripts/show.color.sh`**.

SEE ALSO

d.colormode
d.colors
d.colortable
d.display
d.frame
grass.logo.sh
show.fonts.sh

AUTHOR

David Gerdes, U.S. Army Construction Engineering Research Laboratory

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

show.fonts.sh − Displays and names available font types in the active display frame on the graphics monitor. *(GRASS Shell Script)*

SYNOPSIS

show.fonts.sh

DESCRIPTION

show.fonts.sh is a UNIX Bourne shell macro which runs the *d.erase* −*a* command, then names and displays the font types that can be selected using *d.font*. This macro also runs the GRASS commands *d.font* and *d.text*. See the manual entry for *d.font* for instructions on choosing a font type.

No program arguments are required to run this program.

BUGS

The font is set to *romans* (Roman simplex) after running *show.fonts.sh*. There is no mechanism to query the current font, so there is no way to automatically restore the font. The user will have to reset the font type using *d.font* if *romans* is not desired.

FILES

This program is simply a shell script stored under the **`$GISBASE/scripts`** directory. Users are encouraged to examine the shell script programs stored here and to produce others for their own use.

SEE ALSO

d.display
d.erase
d.font
grass.logo.sh
d.label
d.legend
d.paint.labels
d.text
d.title

AUTHOR

James Westervelt, U.S.Army Construction Engineering Research Laboratory

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.in.garmin.sh − Import gps data from garmin receiver into GRASS sites file
(GRASS Script)

SYNOPSIS

s.in.garmin.sh
s.in.garmin.sh *−h*
s.in.garmin.sh ***name****=sitesfile* ***port****=/dev/gps −v −w −r −t*

DESCRIPTION

s.in.garmin.sh allows to import waypoint, route and track data from a locally connected garmin gps receiver via the *gpstrans* program of Carsten Tschach.

Use at your own risk. This software comes with absolutely no warranty.

No checks are performed for datum, projection and format of data. You must check by yourself that your receiver, gpstrans and GRASS use the same map datum and projection (this means as it is now that you can only use a GRASS database in lat/lon projection and in wgs84 datum).

Parameters:

name*=sitesfile*
: name for new sites file

port*=/dev/gps*
: port garmin receiver is connected to

−v
: verbose output

−w
: upload Waypoints

−r
: upload Routes

−t
: upload Track

−h
: print this message

SEE ALSO

v.in.garmin.sh
gpstrans manual

AUTHOR

Andreas Lange, Andreas.Lange@Rhein−Main.de
gpstrans was written by Carsten Tschach
gpstrans is found at http://www.metalab.unc.edu/pub/Linux/science/cartography/

Last changed: $Date: 2002/06/16 15:29:18 $

NAME

slide.show.sh − Displays a series of raster/vector map layers existing in the user's current mapset search path on the graphics monitor.
(GRASS Shell Script)

SYNOPSIS

slide.show.sh [**across**=*value*] [**down**=*value*] [**prefix**=*name*] [**mapsets**=*list*]

DESCRIPTION

slide.show.sh is a UNIX Bourne shell macro which clears the entire screen, creates a series of display frames on the graphics monitor, and displays in slideshow format each of the raster/vector map layers listed in the user−specified *mapsets*. This is a shell script example which makes extensive use of GRASS and UNIX commands. Users are encouraged to examine this macro and develop similar on−line demos using their own data files.

OPTIONS

Parameters:

across=*value*
: The number of display frames across the graphics monitor screen to be used for map display.
Default: 4

down=*value*
: The number of display frames down the graphics monitor screen to be used for map display.
Default: 3

prefix=*name*
: Specify character(s) to view selected maps only.
default: * (show all maps)

mapsets=*list*
: The names of the mapsets under the user's current location whose raster map layers are to be displayed, separated by commas.
Default: All mapsets listed in the user's current mapset search path.

FILES

See the file **`$GISBASE/scripts/slide.show.sh`**.

SEE ALSO

d.display
d.erase
d.text
g.mapsets
3d.view.sh
grass.logo.sh
show.fonts.sh

AUTHOR

James Westervelt, U.S.Army Construction Engineering Research Laboratory
Vector update, fixes: Markus Neteler

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

split.sh − Divides the graphics monitor into two frames and then displays two maps in these frames.
(GRASS Shell Script)

SYNOPSIS

split.sh mapname mapname [**cmd=***GRASS_command*] [**cmd2=***GRASS_command*] [**view=horiz**]

DESCRIPTION

split.sh is a Bourne shell (*sh*) script that clears the entire graphics screen and divides it into two display frames. Map layers are then displayed in each of the two frames. This command is very useful for visually comparing maps (raster, vector, and 3−d views) and can be used by other GRASS shell macros. It is also useful for creating demos. Program parameters are given below.

OPTIONS

Parameters:

view=horiz
: The graphics screen can be split either horizontally or vertically. The default view splits the screen into two frames, one on the left and one on the right (a vertical split). Some maps (3−d views) are better represented with more width then height (horizontal split). The first map name listed on the command line will be displayed in the top or left window (depending on whether the screen was split horizontally or vertically), and The second map will be displayed in the bottom or right window.

cmd=*GRASS_command*
: The GRASS command used to display the named *mapname*s. If no command is specified by the user, *d.rast* is used by default. However, any GRASS display command (e.g., *d.3d*, *d.vect*, etc...) can be entered.

cmd2=*GRASS_command*
: This command will be used to display map data in the second frame only.
If the user fails to specify the values of both *cmd* and *cmd2*, *split.sh* will use the default command (*d.rast*) to display user−specified map layer names in both frames. If the user specifies only the value of *cmd* on the command line, then that command will be executed for both frames. If the user specifies the values of both *cmd* and *cmd2* on the command line, the *cmd* command will be executed in frame 1 and the *cmd2* command will be executed in frame 2.

EXAMPLES

split.sh soils vegcover

split.sh soils **cmd2=d.legend** "soils color=red"

split.sh elevation vegcover **cmd=**d.3d **view=**horiz

NOTES

split.sh leaves the frame that the last map was drawn in as the active frame. The order in which the options (*cmd*, *cmd2*, *view*) are placed on the command line doesn't matter, but the order is important for the map names.

FILES

This program is simply a shell script. Users are encouraged to make their own shell scripts using similar techniques. See **`$GISBASE/scripts/split.sh`**.

SEE ALSO

d.3d
d.frame
d.rast
d.sites
d.vect

AUTHOR

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

Last changed: $Date: 2003/03/16 12:25:19 $

NAME

tiger.info.sh − Provides tract number(s) and classification codes found within a given U.S. Census Bureau TIGER type1 data file.
(GRASS Shell Script)

SYNOPSIS

tiger.info.sh help
tiger.info.sh *infile*

DESCRIPTION

tiger.info.sh is a shell script which outputs tract number(s) and classification codes found within the TIGER type1 data file *infile*. Output is written to standard out, and can be captured in a file by redirecting output. This information is useful when querying (using *v.db.rim*) from the master binary vector file created by *v.in.tig.rim*. It also provides tract number(s) which can be used as input to the command *Gen.Maps*.

OPTIONS

Parameters:

infile
Name of a TIGER type1 data file.

NOTES

This command must be installed separately as part of the package of routines dealing with the import of Census (TIGER) data.

SEE ALSO

m.tiger.region
v.in.tig.rim
v.db.rim
Gen.Maps
Gen.tractmap

AUTHOR

Marjorie Larson, U.S. Army Construction Engineering Research Laboratory

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

tig.rim.sh − Generates various vector maps from a rim/TIGER data base.
(GRASS Shell Script)

SYNOPSIS

tig.rim.sh help
tig.rim.sh *dbname tractN1 tractN2 ...*

DESCRIPTION

tig.rim.sh is a shell script which queries information from a rim data base using the GRASS command *v.db.rim*, which is an interface between GRASS and RIM. The *dbname* given on the command line should be the name of a rim data base created using *v.in.tiger*. *tig.rim.sh* will create several new vector files. Three of these are: a county outline map, a map of tract boundaries within the county, and a map showing block group boundaries. For every tract number given on the input line, additional vector maps will be created showing: the tract outline, a block group boundary map for each block group within the tract, and a map showing block boundaries within each block group. Output files will be named *dbname.county*, *dbname.tract* and *dbname.bg* for the map layers showing the county outline, the tract outlines within the county, and the block group boundaries within the county. The vector file showing the boundary for an individual tract will be named *TtractN*, where *tractN* is the tract number given on the command line. The vector files created to show an individual block group boundary and block boundaries within that block group will be named using the appropriate tract number and block group number as part of the name, with suffixes of .bg and .bk respectively.

OPTIONS

Parameters:

dbname
: Name of an existing rim data base.

tractN
: Number of a tract located within this county.

NOTES

This command must be installed separately as part of the package of routines dealing with the import of Census (TIGER) data. It requires the use of *rim* and *v.db.rim*, which must be compiled first.

You must include at least one tract number on the command line for this command to function. Use

tiger.info.sh to obtain all tract numbers for a given TIGER type1 data file.

If the master binary vector file created using *v.in.tiger* is modified after it is in GRASS, this program will probably not work. In that situation, the processes from this shell script may simply be run by hand, using *v.db.rim* directly, but searching the old vector file to find lines for the new vector files without using the binary offset field (vectoff).

Vector files showing the block boundaries within a block group may contain hydrology lines, which in fact define the edge of a census block.

SEE ALSO

v.in.tig.rim
v.db.rim
tiger.info.sh

AUTHOR

Jim Hinthorne and David Satnik, GIS Lab, Central Washington University, Ellensburg, WA.

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

v.in.dxf3d.sh − Imports contour levels and master contour levels in DXF file format to GRASS vector file format. (GRASS Vector Program)

SYNOPSIS

v.in.dxf3d.sh dxf=*name* **lines**=*name,name*

DESCRIPTION

The *v.in.dxf3d.sh* data conversion program generates GRASS vector files from a DXF file with contour levels and master contour levels layers with Z values. This shell run successively the programs *v.in.dxf*, *v.in.dxf3d*, and *v.support* for both specified layers.

COMMAND LINE OPTIONS

Parameters:

dxf
: Name of the DXF input design file to be converted to GRASS vector format.

lines
: Name(s) of layer(s) in DXF input file containing the contour levels and master contour levels with Z values, and the mane(s) to be assigned to the GRASS vector files output.

SEE ALSO

v.in.dxf, *v.in.dxf3d*, *v.support*, *v.digit*

AUTHOR

The shell was writted by Evaristo Quiroga, Environmental and Territorial Analysis Center, UAB (12/95).

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

v.in.garmin.sh − Import GPS data from garmin receiver into GRASS binary vector file
(GRASS Script)

SYNOPSIS

v.in.garmin.sh
v.in.garmin.sh *−h*
v.in.garmin.sh ***name****=vectorfile* ***port****=/dev/gps −v −w −r −t −u*

DESCRIPTION

v.in.garmin.sh allows to import waypoint, route and track data from a locally connected garmin gps receiver via the gpstrans program of Carsten Tschach.

Use at your own risk. This software comes with absolutely no warranty.

No checks are performed for datum, projection and format of data. You must check by yourself that your receiver, gpstrans and GRASS use the same map datum and projection (this means as it is now that you can only use a GRASS database in lat/lon projection and in wgs84 datum).

Parameters:

name*=vectorfile*
: name for new binary vector file

port*=/dev/gps*
: port garmin receiver is connected to

−v
: verbose output

−w
: upload Waypoints

−r
: upload Routes

−t
: upload Track

−u
: run v.support on new binary vector file

−h
: print this message

SEE ALSO

s.in.garmin.sh
gpstrans manual

AUTHOR

Andreas Lange, Andreas.Lange@Rhein−Main.de
gpstrans was written by Carsten Tschach
gpstrans is found at http://www.metalab.unc.edu/pub/Linux/science/cartography/

Last changed: $Date: 2002/06/16 15:29:18 $