GRASS Vector Commands

v.alabel
v.apply.census
v.area
v.autocorr
v.bubble
v.build.polylines
v.cadlabel
v.circle
v.clean
v.cutter
v.db.reclass
v.db.rim
v.digit
v.distance
v.dump
v.export
v.extract
v.geom
v.import
v.in.arc
v.in.arc.poly
v.in.ascii
v.in.atlas
v.in.dlg
v.in.dlg.scs
v.in.dlg2
v.in.dlg_atr
v.in.dxf
v.in.dxf3d
v.in.dxf3d.sh
v.in.gshhs
v.in.poly
v.in.sdts
v.in.shape
v.in.tig.basic
v.in.tig.lndmk
v.in.tig.rim
v.in.tiger.scs
v.in.transects
v.info
v.label
v.llabel
v.mkgrid
v.mkquads
v.mkstats
v.out.arc
v.out.ascii
v.out.atlas
v.out.dlg
v.out.dlg.scs
v.out.dxf
v.out.e00
v.out.idrisi
v.out.moss
v.out.scsgef
v.out.sdts
v.out.shape
v.patch
v.plant
v.proj
v.prune
v.psu
v.psu.subj
v.random
v.reclass
v.reclass.scs
v.report
v.rmdup
v.rm.dangles
v.rmedge
v.scale.random
v.sdts.dq.cp
v.sdts.meta.cp
v.sdts.meta
v.spag
v.stats
v.support
v.surf.rst
v.timestamp
v.to.db
v.to.gnuplot
v.to.rast
v.to.sites
v.transform
v.what

Other Commands

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

NAME

v.alabel − Bulk−labels unlabeled area features in a binary GRASS vector file.
(GRASS Vector Program)

SYNOPSIS

v.alabel
v.alabel help
v.alabel [**−i**] **map**=*name* [**value**=*value*] [**label**=*value*]

DESCRIPTION

v.alabel allows the user to bulk−label currently unlabeled polygons (area features) in a binary GRASS vector file (i.e., a *dig* file). The user must run *v.support* on the vector file before running *v.alabel* if any modifications have been made to the file since the last time *v.support* was run on it, to ensure that all area features are properly identified in the file topology.

The user must also run *v.support* on the vector file after *v.alabel* is run for labeling changes to be made evident.

v.support builds GRASS support files for (binary) GRASS vector data files. These support files supply topology (*dig_plus*) and category (*dig_att*, *dig_cats*) information that are needed by other GRASS programs (e.g., by *v.digit*, *v.to.rast*, etc.).

OPTIONS

Program parameters and flags have the following meanings.

Flags:

−i
: Label areas incrementally. For each unique, unlabeled polygon in the vector file, increment the category value by one, starting from the initial user−assigned *value*.

Parameters:

map=*name*
: Name of binary GRASS vector data file whose unlabeled areas are to be labelled. This map layer must exist in the user's current GRASS mapset.

value=*value*

The category value to be assigned to all unlabeled areas in the vector map layer. If the *−i* flag is set, *v.alabel* will increment the initial *value* by one for each unique unlabeled polygon in the vector map. Default: 1

label*=value*

The category label to be assigned to all unlabeled areas in the vector map layer. If not specified, each category added will have an empty label in the *dig_cats* file.

The user can run this program non−interactively by specifying parameter values (and optionally, the flag setting) on the command line.

Alternately, the user can simply type the command **v.alabel** on the command line. In this event, the program will prompt the user to enter parameter values and flag settings.

NOTES

A *dig_plus* file must be created for each imported vector map before it can be used in post−GRASS 3.0 versions of *digit* (now referred to as *v.digit*).

Topological information for GRASS vector files can also be built using *option 4* of the *v.import* program.

The user can bulk−label unlabeled line features in a binary vector file using *v.digit*.

SEE ALSO

v.digit
v.import
v.in.ascii
v.prune
v.support
v.to.rast

AUTHORS

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

Last changed: $Date: 2002/03/03 18:50:50 $

NAME

v.apply.census − Calculate/Import Demographics from Census STF1 Files
(GRASS Vector Program)

SYNOPSIS

v.apply.census
v.apply.census help
v.apply.census [−d] [−p] [−l] [−u]
v.apply.census input_stf1=*name*
[**out=**name] [**ef=**name] [**formula=**mapname=expression]
[**name_field=**name] [**zone=**value] [**spheroid=**name]

DESCRIPTION

This program reads a previously selected subset of STF1 or PL94−171 U.S. Census Bureau demographic records (see *m.in.stf1.tape*), and writes one of the following:

GRASS site list (coordinates and a value).
GRASS vector polygon attribute labels file (coordinates and a label value).
Text report to stdout in one of two formats.

Any arithmetic expression combining constants with any of the more than 900 hundred demographic (numeric) fields can be defined and will be evaluated (by the Unix *bc −l* utility) to compute the value for each input record.

The location coordinates for the polygon label point or the site location is obtained from certain columns (269−287) in the input records, making this program quite specific for the specified types of STF1 and PL94−171 input file records. Use with other types of input data is not advised.

OPTIONS

Flags:

−d
Output IDENTIFICATION SECTION to stdout (20 pages)

−p
Output STF1 MATRIX TABLE to stdout (30 pages)

−l
Output PL94−171 MATRIX TABLE to stdout (1 page)

−u
Output STF1 SUMLEV TABLE to stdout (4 pages)

Note:

Only the first flag given will be executed; the program exits after sending one table to stdout. Other parameters are ignored.

Parameters:

input_stf1*=name*

Path/name of STF1 or PL94 input file

out*=name*

Type of output:

`site | atts | table:Lxxx | − (stdout)`

default: −

att

for results to vector map attribute file

site

for results to site list

−

for results to stdout; this is the default

table

for results in table form to stdout with the column value(s) represented by the ':Lxxx:Lxxx' string preceeding the expression value instead of easting and northing. Lxxx is in same choice of representation as column designation in formula (see below).

ef*=name*

Path/name of text file with formula expression(s)

formula*=map=expression*

mapname=Computation with STF1A columns, constants and *bc* operators (see below). mapname of vector or site map to make is required in all cases (but ignored for '−' or 'table:' output modes).

name_field*=name*

field name for parsing stdin lines (−a to ignore)

default: −a

name is Keyword in stdin stream preceeding the number of a STF1 record to read from input_stf1 file. The number is a physical sequence number, not a record identification number. This parameter is appropriate only in special uses of this program. If '−a' is given, or this parameter is omitted, all records in STF.file will be processed.

zone*=value*

UTM zone number; default is location zone

options: 1−60

default: 10

spheroid*=name*

Spheroid for LL to UTM conversion; see m.gc.ll

default: clark66

NOTE: Only one of the "ef" or "formula" input fields may be used.

THE "formula" PARAMETER

The string after the '=' in this parameter almost always must be enclosed in quotes to protect it from Shell interpretation of characters such as * / ( ) and spaces (which may be used to increase readability, but are not

necessary). This string is always of the form *mapname=expression*.

The *mapname* string can be any legal GRASS map name (up to 14 characters). The second '=' is required and may be preceeded and followed by a space.

In the *expression*, great flexibility is provided for the computation of interesting combinations of the data fields contained in the demographic records.

The usual operators used in the *expression* include: +, −, *, /, and %. The user should review the Unix manual entry for the *bc* calculator for the complete list of available functions and other operators.

Parentheses are used in the normal algebraic manner.

The operands in the *expression* may consist of any mixture of:

integer constants real constants functions allowed by *bc* numeric fields from the demographic records

Numeric fields from both the IDENTIFICATION and MATRIX SECTIONS of the demographic records may be used. Review the User and Technical Documentation for these demographic files; or use the −*p* option of this program to print the MATRIX SECTION document (the demographic data fields). When substituting values from the numeric fields into the expression, <SPACE> characters are replaced by zero. (Spaces, which are rare in the demographic data, are usually the result of missing values or restricted information).

The numeric fields from the demographic records may be designated in one of three ways in the *expression*.

1. 'Lcccc', where 'L' is an upper case alphabetic letter which indicates the length of the numeric field in the data records: A=1, B=2, ... , I=9, J=10, O=15, etc; and 'cccc' is the starting column number for the data field of interest: 301 for 100% population count, 7221 for total number of four−room housing units, 58 for 101st Congressional District, etc. The proper specification of the Congressional District number would be 'B58' because it is a two−column field. 'I301' would indicate that the 100% poplation should be used in the calculation; it's a nine−column field.

2. 'Pnna0nnn' or 'Hnna0nnn', where 'n' is a digit and 'a' is a digit or (rarely) upper case letter. These forms represent the MATRIX field naming schemes used in the CD−ROM "dBase 3" files. They can be used in processing STF1 records extracted from the CD−ROM or TAPE distribution media. All eight characters of the field name must be used. (Note: this form cannot be used in processing the PL94−171 records.)

3. 'ID_NAME'. The STF1 and PL94−171 files use the same set of IDENTIFICATION SECTION field names (67 fields) for the "locational" information contained in the first 300 characters of each record. The field name, in all upper case letters, may be used if the numeric value of the field is needed in the *expression*. Two of the most useful fields are AREALAND and AREAWAT which allow the direct computation of population density values.

The *bc* calculator usually returns a value with a very large number of decimal places. Vector attributes are automatically truncated(!) to integers by the v.support program when the topology file is built. Site list descriptions are likewise truncated to integers by GRASS programs which use the descriptions as numeric values (e.g., sites to cell in *s.menu*).

THE 'out=table:' PARAMETER

The default operation of *v.apply.census* when tabular reports are produced to *stdout* (when not making a sites list or vector attribute file) is to print the easting and northing coordinates and then the value resulting from the *expression*. Often the user wishes to have an identifier different from the coordinates. The construction *out=table:field* will replace the coordinates with the character string indicated by *field*, which may have any of the three forms used for numeric fields in the formula *expression*, see above. Note: a special case exists for the 66 character description field which begins in column 192; the entire field will be printed if designated by either of these two forms: *out=table:ANPSADPI* or *out=table:Z192*.

Complex tables may be produced by making multiple runs of *v.apply.census*, redirecting the tabular output to files, and using the Unix tools *cut*, *paste*, and *colrm* to combine the resulting files.

EXAMPLES

[These examples assume that STF1 records for the census tracts (SUMLEV=140) in a particular county (CNTY=037) have been extracted from the distribution source (with *m.in.stf1.tpe*) and saved in a file named 037.140 in the current directory.]

1. Create a site list where the label values are the percentage of females in each input record:

   v.apply.census in=037.140 **out=**site **formula=**'pct.female = 100 * (I373 / I301)'

2. Label an existing vector map (named tract.pop) of tract boundaries with the total population of each tract (run *v.support* and build topology after creating the attributes file):

   v.apply.census in=037.140 **out=**atts **formula=**tract.pop=P0010001 (or)
   v.apply.census in=037.140 **out=**atts **formula=**tract.pop=I301

3. Produce a tabular report of the census tract numbers and the number of Hispanics per square kilometer:

   v.apply.census in=037.140 **out=**table:TRACTBNA
   formula="m=(P0080001/AREALAND)*1000"

4. Produce a tabular report of the number of people per housing unit for each tract and the coordinates of the internal (label) point:

   v.apply.census in=037.140 **formula=**"map=P0010001/H0010001"

FORMAT OF THE FORMULA TEXT FILE

Running this program with the *ef=file* command line parameter causes the named file to be read and each formula contained therein to be processed as if it were entered on the command line. The *out=* parameter may optionally be respecified in this file; each *out=* selection remains in effect until explicitly changed. The program exits after the last formula is processed.

The following is a sample formula file. Note the use of lines beginning with '#' as manditory formula separators and comment lines. Also note that expressions may be continued on successive lines; the lines are

concatenated to a maximum of 500 characters for a single formula. Blank lines are ignored.

```
popden.sqkm = 1000 * P0010001 / (AREALAND+AREAWAT)
# that computes population density in people per sq. km.
#
# next do people per sq. mile as a vector attribute file
out=att
popden.sqmi = 2.59*1000 *
P0010001 / (AREALAND+
AREAWAT)
#
# next do total population as a vector attribute file
total.pop = I301
#
# output the county identification numbers as the descriptions
# in a site list
out=sites county = CNTY
#
# output the 66 char. description and FIPS Place Code as a table
out=table:ANPSADPI
map=PLACEFP
# optional ending comment line
```

BUGS/FEATURES

Computational errors in *bc* are not handled too gracefully: a warning is output and a zero result is used.

bc tends to output lots of decimal places. The user must clean this up for output sent to stdout.

The GRASS site list output format used includes the '#' before the label value to facilitate the production of raster maps with cell values representing the results of the formula execution.

If using the "name_field" and "ef" parameters, the formula file may contain only one formula.

SEE ALSO

m.in.stf1.tape is used as a preprocessor to select subsets of STF1 or PL−171 tape format records for input to this program.

Unix utility programs such at grep or awk can also be used to select subsets of lines from the PL94−171 files, but not from the STF1 tape files due to their very long record lengths.

AUTHOR

Dr. James Hinthorne, GIS Laboratory, Central Washington University. July, 1992.

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

NAME

v.area − Display GRASS area and perimeter information for GRASS vector map.
(GRASS Vector Program)

SYNOPSIS

v.area
v.area help
v.area [**−f**] **map**=*name* [**color**=*name*]

DESCRIPTION

The GRASS program *v.area* first displays the selected vector file. Then user can select area on map by clicking with mouse within the desired area. Selected area will be highlighted in selected color on graphics display. On regular screen area information will be displayed, in square meters, hectares, acres, and square miles, Perimeter measurements, in meters, feet, and miles, are also displayed.

User can repeatedly select areas for analysis, one at a time.

Flag:

−f
: Fill selected area with selected color on graphics display, rather than simply outlining it in the *highlight* color.

Parameters:

map=*name*
: Name of the vector map layer to be displayed.

color=*name*
: Color in which perimeter will be highlighted.
Default: red

If the user simply types **v.area** without specifying program arguments on the command line, the program will prompt the user for the name of a map. Then the user is given the option of specifying a highlight color, and then, whether or not the display of selected area(s) is to be filled, rather than merely outlined, with the highlight color.

SEE ALSO

d.vect

AUTHOR

Bruce Powell, National Park Service, Denver CO.

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

NAME

v.autocorr − Calculate spatial autocorrelation statistics for GRASS vector file.
(GRASS Vector Program)

SYNOPSIS

v.autocorr
v.autocorr help
v.autocorr [**−chnqw**] **vect=***name* [**output=***name*]

DESCRIPTION

v.autocorr uses a labeled binary vector file (vect=*name*) to calculate spatial autocorrelation statistics for areas. The output may either be printed to the screen or saved to a file (vect=*name*).

Flags:

−c
Print the connectivity matrix.

−w
Print the weight matrix (a re−expression of the connectivity matrix).

−n
Suppress calculation of statistics. Useful when used with **−c** or **−w**.

−h
Do calculations for hypothesis testing. Warning: this is intensive!

−q
Quiet. Cut out the chatter.

Parameters:

vect=*name*
Name of a labeled binary vector file. The categories values must be numeric.

output=*name*
Name of the output file.

v.autocorr can be run either non−interactively or interactively. The program will be run non−interactively if the user specifies the name of a labeled binary vector file.

Alternately, the user can simply type **v.autocorr** on the command line, without program arguments. In this case, the user will be prompted for parameter values using the standard GRASS parser interface described in the manual entry for *parser*.

SEE ALSO

s.voronoi
v.stats
parser

Daniel A. Griffith, (1987). *Spatial Autocorrelation − A Primer*. Association of American Geographers.
RANLIB.C
a library of C routines for random number generation compiled and written by Barry W. Brown and James Lovato of the M.D. Anderson Cancer Center at the University of Texas, was adapted for use by this program.

BUGS

The **−w** flag does not work yet.
This program only works for fully labeled vector files (with sequential categories beginning with 1). I don't know what it will do in any other situation.
Please send all bug fixes and comments to the author.

AUTHOR

James Darrell McCauley, Agricultural Engineering, Purdue University

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

NAME

v.bubble − Creates a vector file which consists of bubble(s) for each point in a site−file and with a size relative to the z value of that point.
(GRASS Vector Program)

SYNOPSIS

v.bubble
v.bubble help
v.bubble [**−s**] **[radius=***value]* **sitefile=***name* **output=***name* **field=***value*

DESCRIPTION

"v.bubble" will create "polygon" circles or bubbles around points read from an existing "site_lists" file. The "site_lists" points will be the centers for those bubbles with one bubble created per point. The size of the bubble is relative to the z value (or "height" attribute") at that point. The maximum z value corresponds to a size of "radius". The "polygon" bubbles will be written to a binary vector file. Each "polygon" bubble will have 361 points with each point on the circumference of the bubble representing 1 degree of arc.

COMMAND LINE OPTIONS

Parameters

radius
Maximum radius corresponding with the maximum z value in the "site−file". One Unit of radius corresponds with one unit of the map.
default: 1.0

sitefile
GRASS site_lists file (input).

output
Vector file to be created (output).

field
Attribute field number to use for operation
The field number specifies the attribute field in sites list (point map) which shall be used. This is useful in case of multiple attribute sites maps.

Flags

−s

Automatically run "v.support" on newly created vector file.

NOTES

The programm is based on v.circle

SEE ALSO

s.buffer, v.circle

AUTHOR

Job Spijker (spijker@geo.uu.nl) , Utrecht University/Department of Geochemistry

Last changed: $Date: 2003/07/04 08:35:34 $

NAME

v.build.polylines − Build polylines from lines.
(GRASS Vector Program)

SYNOPSIS

v.build.polylines
v.build.polylines help
v.build.polylines [**−acoq**] **input**=*name* **output**=*name* [**type**=*name*]

DESCRIPTION

v.build.polylines builds polylines from the lines in a binary vector file. It outputs the polylines in either binary or ASCII vector format, and if requested, copies the attribute and category files from the original file.

A line is a single straight line segment defined by one start node, one end node and no other nodes. A polyline is also defined by one start node, one end node and no other nodes, but is made of two or more consecutive straight line segments. The connections between the constituent line segments of a polyline do not appear as nodes in the vector map.

Polylines provide the most appropriate representation of curved lines when it is important that nodes serve to define topology rather than geometry. Curved lines are usually digitized as polylines, but these are sometimes broken into their constituent straight line segments during conversion from one data format to another. *v.build.polylines* can be used to rebuild such broken polylines.

OPTIONS

The program can be run either non−interactively or interactively. To run *v.build.polylines* non−interactively, the user must specify *input* and *output* file names on the command line. The line *type* and flags are optional.

To run *v.build.polylines* interactively the user should simply type **v.build.polylines** on the command line, in which case the program will prompt for parameter values using the standard GRASS interface described in the manual entry for *parser* .

Flags:

−a
Make the *output* file an ASCII vector map instead of a binary vector map.

−c
Copy the attribute (`dig_att') and category (`dig_cats') files associated with the *input* file to new files with the *output* file name. You should set this flag if you wish to retain category information in the

new binary file.

−o

Silently overwrite the *output* file if it exists in the current mapset. When this flag is not set *v.build.polylines* will refuse to create the *output* file if a file of the same name exists in either the `dig' or the `dig_ascii' mapset elements.

−q

Run (relatively) quietly. Specifically, do not provide information (including warnings) about each polyline.

Parameters:

input*=name*

The name of a binary vector map layer containing lines (and possibly some polylines).

output*=name*

The name of the ASCII vector map to be created in the dig_ascii mapset element.

type*=name*

The type of each new polyline can be set from the *input* map, or alternatively all new polylines can be bulk labeled as either lines, area edges, or points.
Options: source, line, area, point.

NOTES

For full functionality *v.build.polylines* requires *v.support* and *v.in.ascii* .

If the lines that make up a polyline are of different types, then *v.build.polylines* will set the type from the first constituent line. *v.build.polylines* will issue a warning unless the flag *−q* has been set. It is possible to keep a list of all such warnings by redirecting standard output to a file.

If the lines that make up a polyline have different attribute values then *v.build.polylines* will set the attribute value of the polyline to that of the last line (this is the behaviour of *v.support* , which is used to assign the attribute values). Any warnings issued by *v.support* will be visible unless the flag *−q* has been set.

v.build.polylines correctly handles *input* maps containing lines, area edges and points. Lines and area edges will be converted to polylines of the desired type. Areas are only guaranteed to be preserved if the constituent lines of the polylines that define them are all area edges in the input map. Points will remain points provided that *type* has been set to `source'. It is possible to convert lines and area edges to points or vice versa, but this is rarely useful.

Use *v.import* to convert an ASCII *output* map to a binary vector map.

SEE ALSO

v.import, *v.support*, *v.in.ascii*, *parser*

ACKNOWLEDGEMENTS

This program was written during the author's tenure of a Leverhulme Special Research Fellowship at University College London.

AUTHOR

Mark Lake, Institute of Archaeology, University College London.

BUGS

Please email mark.lake@ucl.ac.uk if you find any bugs.

Last changed: $Date: 2003/08/20 08:10:13 $

NAME

v.cadlabel − Attaches labels to (binary) vector contour lines that have been imported to GRASS from DXF format.
(GRASS Vector Program)

SYNOPSIS

v.cadlabel
v.cadlabel help
v.cadlabel lines=*name* **labels=***name*

DESCRIPTION

v.cadlabel attaches labels to index contour lines by using the index contour lines and lables files that have been converted from DXF format to GRASS vector format by the *v.in.dxf* program. Users have the option of creating either binary or ASCII output with *v.in.dxf*. Since *v.cadlabel* works only on binary vector files, the ASCII GRASS vector (**dig_ascii**) files that are generated by *v.in.dxf* must be converted to binary GRASS vector (**dig**) files using the *v.in.ascii* program before *v.cadlabel* can be executed.

v.cadlabel searches a binary GRASS vector file of contour lines to find the contour lines that are closest to each box (contour label) in the binary GRASS vector file containing contour labels. The two contour lines that are closest to a box are tagged with the same label (an elevation value) as that of the box.

OPTIONS

Program parameters are described below.

Parameters:

lines=*name*
Name of the binary GRASS vector (**dig**) file, imported from DXF format, that contains index contour lines.

labels=*name*
Name of the binary GRASS vector (**dig_att**) file, imported from DXF format, that contains index contour line labels.

NOTES

Because line data that are created in CAD format may have unsnapped nodes or gaps, *v.cadlabel* will not always be able to label every index contour line. Also, intermediate contour lines that may be contained in the

index contour vector file (because they resided on the same DXF level as the index contour lines in the DXF design file) will not be labeled. Any lines that are not labeled with *v.cadlabel* can be labeled with the contour labeling program in *v.digit*.

If the intermediate contour lines are in a separate GRASS **dig** file, they can be patched to an labeled index contour file with the GRASS program *v.patch*, and then labeled in *v.digit*.

The user does not need to *v.patch* the labels vector file (boxes) to the lines vector file (contour lines). The purpose of the labels vector file is to determine which labels should be assigned to which contour lines (in the lines vector file) during the execution of *v.cadlabel*.

SEE ALSO

v.digit
v.in.ascii
v.in.dxf
v.out.dxf
v.patch
dxfout (Microstation tool)

AUTHOR

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

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

NAME

v.circle − Creates a vector file which consists of circle(s) which uses each point in a "site_lists" file as the center of those circle(s).
(GRASS Vector Program)

SYNOPSIS

v.circle
v.circle help
v.circle [**−s**] **sitefile**=*name* **[radius**=*value]* **[radius_uom**=*name]* **[area**=*value]* **[area_uom**=*name]* **output**=*name*

DESCRIPTION

"v.circle" will create "polygon" circle(s) around points read from an existing "site_lists" file. The "site_lists" point(s) will be the center(s) for those circle(s) with one circle created per point. The "polygon" circle(s) will be written to a binary vector file. Each "polygon" circle will have 361 points with each point on the circumference of the circle representing 1 degree of arc.

COMMAND LINE OPTIONS

Flags

−s

Automatically run "v.support" on newly created vector file.

Parameters

radius

Radius of circle(s) with "site_lists" point(s) as center(s). If radius selected then area values are not used for computations. If both radius and area selected, then radius has precedence over area.
default: 0.0

radius_uom

Radius unit of measure, ie. (m)meters, ft(feet), (mi)miles.
default: m

area

Area of circle(s) with "site_lists" point(s) as center(s). If area selected then radius values are not used for computations.
default: 0.0

area_uom

Area unit of measure, ie. sqm(square meters), ac(acres), sqmi(square miles), hec(hectares). default: sqm

sitefile

GRASS site_lists file (input).

output

Vector file to be created (output).

SEE ALSO

s.buffer, v.bubble

AUTHOR

Bruce Powell, National Park Service

Last changed: $Date: 2003/07/04 08:35:34 $

NAME

v.clean − Cleans out *dead* lines in GRASS vector files.
(GRASS Vector Program)

SYNOPSIS

v.clean
v.clean help
v.clean map=*name*

DESCRIPTION

When programs like digit delete lines, they do not really go away, but are simply marked as deleted. This is done primarily for speed. But a side effect is that vector files can end up carrying a lot of dead weight with them.

v.clean will go through a vector file and remove all dead lines in the file thus potentially reducing the size of the file.

You do *not* have to run *v.support* afterwards.

OPTIONS

Parameter:

map
Name of the GRASS vector file to be cleaned.

BUGS

None known.

SEE ALSO

v.clean, *v.digit*, *v.prune*, *v.rmdup*, *v.spag*, *v.support*

AUTHOR

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

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

NAME

v.cutter − Polygon Cookie Cutter
(GRASS Vector Program)

SYNOPSIS

v.cutter
v.cutter help
v.cutter [−o] [−l] [−a] [−q] cutter=*name* **input**=*name* **output**=*name* **type**=*name*

DESCRIPTION

This program provides a way to generate new maps based on an intersection of two existing maps. It in effect provides a way to create masked versions of vector maps. Lines, sites, and polygons are clipped correctly. By default v.cutter will extract line−type arcs.

OPTIONS

Flags:

−q
: Run quietly. Don't print percent information.

−o
: Only intersect labeled areas. Intersection of unlabeled areas may not be dependable. The flag is meaningful only for type=area or type=both.

−a
: Extract area−edge arc types only. The flag is meaningful only for type=line or type=both. By default the program uses both area−edge and line arc types.

−l
: Extract line arc types only. The flag is meaningful only for type=line or type=both. By default the program uses both area−edge and line arc types.

Parameters:

cutter=*mapname*
: Name of the binary vector file to use for the cookie cutter.

input=*mapname*
: Name of the binary vector file which is to be cut.

output=*mapname*
: Name of new vector file to be created.

type=*object type*

Type of object (area, line or both) to extract. The default is to extract lines.

NOTES

If v.cutter is only used to extract lines then it automatically runs *v.support* on the output file. If v.cutter is used to extract polygons then the user may need to run *v.spag* *−i* after running *v.cutter* to remove identical lines.

If type=area then the edge created by the cutter file will appear on closed polygons in the output. If type=line then the edge does not appear in the output.

The attributes of the output map will be generated from the attributes of the *input* map.

BUGS

There are a few rare situations that are not currently handled correctly. These mostly involve nodes or vertices intersecting exactly with lines of the opposite map. You will know you have hit one of these cases because a lot of strange text will start spitting out.

Borders between areas with the same attributes are not dissolved.

There are many places where the code could be optimized greatly.

SEE ALSO

v.spag
v.digit
v.support
v.clean

AUTHOR

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

Last changed: $Date: 2002/04/15 15:18:16 $

NAME

v.db.reclass − Changes vector category values for an existing vector map according to results of SQL queries. *(GRASS Vector Program)*

SYNOPSIS

v.db.reclass
v.db.reclass help
v.db.reclass [**−d**] **type**=*name* **input**=*name* **output**=*name* **table**=*name* [**key**=*name*] **rules**=*name* [**TITLE**=*name*]

DESCRIPTION

v.db.reclass allows a user to create a new vector map based on the reclassification of an existing vector map. The user provides the program with a rules file, input vector map name, an output vector map name, and the type of input map, table, key. There is an option (d) to dissolve common boundaries between adjoining map areas of the same re−classed category value.

Note: The dissolve option will work on only those areas which are of the same conversion category value. If a map area is inside (island) a converted area and is NOT converted to the same value, its boundaries are output to the resultant map.

COMMAND LINE OPTIONS

Flags:

−d
Dissolve common boundaries (default is no) .

Parameters:

type=*name*
Select area, line, or site.
Options: area, line, site (multiple allowed)
input=*name*
Vector input map name.
output=*name*
Vector output map name.
table=*name*
Database table.
key=*name*

Column corresponding to cats in vector map.

rules=*name*

Name of text file containing reclassification rules. Rules file may contain on each row either pair: *keyword value*
(separated by space) or comment beginning by #(hash). Definition of new category begins with kyeword *cat* followed by new category value. Keyword *where* specifies SQL where condition and optionaly *label* keyword specifies category label.

TITLE=*name*

Title for the resulting vector map.

EXAMPLE

v.db.reclass −d input=*land* **output**=*land_u* **type**=*area* **table**=*tland* **key**=*id* **rules**=*land.rcl*
the rules file contains :

```
# land reclass file
cat 1
where use = 'E13' and owner = 'Jara Cimrman'
cat 2
where use = 'E14'
label Use type E14
```

Produces a new vector area file *land_u* containing 'area' boundaries from *land* with area category values selected from database by SQL select statement: *select id from tland where use = 'E13' and owner = 'Jara Cimrman'* changed to category 1; values selected from database by SQL select statement: *select id from tland where use = 'E14'* changed to category 2. Any common boundaries are dissolved.

SEE ALSO

v.extract

AUTHOR

R.L. Glenn, USDA, SCS, NHQ−CGIS
from v.reclass to v.db.reclass rewritten by Radim Blazek

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

NAME

v.db.rim − RIM data base management/query interface for GRASS vector

SYNOPSIS

v.db.rim *data_base* −− to run the command version **v.db.rim** −− to run the interactive version (see SECTION TWO)

OVERVIEW

v.db.rim allows you to create, manage and query information about labeled lines and areas from a number of different vector maps in a batch mode or interactively. Operations are done on a data base through a command language defined below or an interactive set of menus described in SECTION TWO.

This program, like **s.db.rim**, is actually a marriage of the GRASS environment and the programmer's interface library of the relational data base management program RIM distributed publically by the University of Washington Academic Computing Services. Your system must have a FORTRAN 77 compiler to use this program.

DESCRIPTION

The vector data bases are stored in a subdirectory named 'rim/vect' in the user's current mapset. Data bases in other mapsets, selectable through the GRASS **g.mapsets** command, can be accessed for 'read−only' retrieval of records. Each mapset may have many data bases. Each data base within a mapset must have a different name; user supplied names are limited to seven (7) characters in order to maintain compatibility with the standard version of RIM. As with other GRASS commands, mapsets are searched in the mapset SEARCH_PATH order when a data base needs to be opened.

Each vector data base is composed of multi−field records (rows or tuples, in DBMS jargon). Each field and its position in the record form is defined via input to the .make command when a data base is originally defined. It is possible to add new fields or change the length of existing fields after data has been loaded, however this is not straightforward; deleting of fields is also possible, but requires even more experience and knowledge. The user needs to carefully design the data base fields and form (layout) and check the operation with a few pieces of test data before loading data for a large number of records.

One significant difference between **v.db.rim** and **s.db.rim** is that v.db.rim needs access to the original vector maps that data were imported from to successfully complete some commands. To keep track of which records were imported from which vector maps a new "table" has been added that keeps information about the reference vector maps. This table is called "Referencemaps" in the RIM data base. The v.db.rim commands .read_vect, .map and .maps allow the user to view and update this table. The .vector_map command, which creates a new GRASS binary vector file, requires that the reference vector maps for the database be

accessible. If a reference map is not accessible, any vectors represented by the data base records that were generated from that vector map will not be transferred to a new vector map. Reference vector maps may be in any mapset in the current SEARCH_PATH.

COMMANDS

[Note: For each of the "dot commands", i.e., .make, described below there is a menu choice to selected when running the interactive version. The interactive menus are described in the SECTION TWO of this document. Some display capabilities exist in the interactive version which are not directly implemented in the command version.]

The commands are given alphabetically here for easy reference. The .make command is required to create a data base. Abbreviations down to the string shown in ( ) are accepted; this is primarily for those giving **v.db.rim** commands from a terminal, but abbreviations may also be used in batch files.

Each command is introduced with an input record (line) which starts with a period and is followed by one of the words shown below; for some commands the command line also contains one or more required or optional parameters. Additional or optional input instructions/data for a command is supplied on successive lines; a .end line is needed by some commands to signify the end of these lines.

"Alphabetical Command Summary"

"!command"

This is the only **v.db.rim** command not starting with a period. "command" is a single shell command line (no more than 80 characters) which is executed by a "G_system()" call (see GRASS gis library). Many UNIX utilities (e.g., vi, ls, print) and most GRASS commands (e.g., d.rast, d.sites, g.list, g.region, r.mask) may be executed. It is permitted, and often useful, to change "region" and "MASK" within **v.db.rim.** Multiple commands may be separated by ";" in the standard UNIX way. Note that a "!cd directory; ls" will change to the specified directory and list files, but the effective working directory for **v.db.rim** will not be changed when the command terminates.

".add (.a)"

Add a new vector record (row) to the open data base. Each line following contains a field name followed by spaces and/or tabs then the value or character string to store for that field. Field information lines end with .end. Some fields may be absent and fields may appear in any order. The input of data is checked for one required field (sequence number), for field length for text type fields, and for duplicate sequence numbers. If split text fields are used in the data base layout (see .make), text data for each split field must be added as a separate line. If there are any problems, the record will not be stored and a message will be output. This format makes it relatively easy to import data from most other DBMS. The ".print −a" command, see below, outputs data in this list format.

```
Example:
 .add
 seq_num   204
 north     4690673.30
 east      601410.00
 map_num   1
 vect_type L
 reference Jones (1987)
 .end
```

".backup (.b) file_name"

The .backup command is used to dump the entire data base from the RIM binary files to a text file format (see UNLOAD in the RIM User's Manual). The file_name can be a relative path name or full path name. The file will contain the data base definition, screen layout information, and tabular data. This text file is transportable to RIM or **v.db.rim** running on any other computer; it may also be reloaded to recreate the **v.db.rim** data base. A message will be output if there is any problem writing the .backup file. Backup can only be done on data bases in the user's current mapset.
To reload your data base from the backup file (normally not necessary):

```
GRASS 4> cd $LOCATION/rim/vect     #right directory
GRASS 4> rm db_name.rimdb*         #remove data base
GRASS 4> rim                       #run RIM manually
RIM> input "path/file"             #RIM rebuilds data base from data
                                    written by .backup
RIM> exit
```

".change (.c) [−l]"

Without the "−l" flag, each line following .change is in the same format as for the .add command. The sequence number field is required and the sequence number must match an existing site in the data base. Only those fields for which lines are provided are changed in the record. After the .end the changed record is stored, if all is ok, otherwise a message is output.
If the "−l" flag (for "list") is given, the sequence number field is omitted and the specified field values are changed in all records currently selected by .find and/or .query.

".delete (.d)"

This command is used to delete data records. Deletion of records is permanent. Each line following the command should contain only a sequence number that you want to delete, with a .end line being last.
A backup of the data base or copies of the data base files are the ways to protect your valuable data. The following command sequence will delete all the records currently on v.db.rim's query list (the result of the last .query or .find command), after asking for approval.

```
.delete
.end
```

".end (.e)"

Ends multi−line input for several other commands.

".exit (.ex)"

Use .exit to end operation of **v.db.rim** cleanly. In general, do not use CTRL−C to exit unless absolutely necessary. When .exit is encountered in a batch file, input will revert back to the previous file, or the terminal, if any, which called the batch file.

".find (.f) [−m | −w] [−a | −d]"

The .find command is used to find the record(s) whose location (label point) is closest to a given point (the target). The target can be defined in one of several ways. The found records are stored on an internal query list for output by other commands; however, see note 2, below. Records are stored on the query list in order of proximity to the target location. The optional .find command line parameter specifies the current MASK (−m), if any, or the current region (−w), as a filter on the retrieved records; see notes 3 and 4, below. The append (−a) or delete (−d) options allow the "found" records to be added or deleted from the currently selected ones. When adding, duplicates will be discarded.
The single required line following the .find line gives the program the necessary target information. The following examples show the possibilities.

```
find> 602793.90 4379010.00
```

will find the one record nearest these coordinates and store it, append it or delete it on the internal query list.

```
find> 619840 4599000 10
```

will find the 10 records (or fewer, if there are not that many) closest to the given location.

```
find> record 132 10
```

will find the 10 records closest to the location (label point) for record 132 in the data base (including record 132). If record 132 does not exist, no action is taken.

```
find> distance from 472910.06 5732001.0 5000
```

will find all records within 5000 (meters in UTM coordinates) of the target location.

```
find> distance from record 16 −2500
```

will find all records greater than 2500 (meters) from the location of record 16.

Notes for .find:

1. All records found by each .find are stored on the query list in order of proximity to the target location (sorted by distance from target).

2. The number of records found is automatically printed to the active output device/file.

3. If mask is specified, the effective region is automatically set to the current region (because the GRASS mask is only defined for the current region).

4. Region and mask filtering uses the current resolution for the region to test if a point falls within a cell.

5. In the last two examples the string "distance from" must be exactly matched. Also, the word "record" must be exactly matched.

6. If the "distance from" radius is given as a negative value, points outside the target circle are selected; whereas, if a positive value is given, points inside the circle are selected.

7. The current region may be changed with !g.region or !d.zoom prior to doing a .find, and the mask may be set or removed with a variety of GRASS commands.

8. The "find>" prompt is given only when input is from a terminal.

9. The "distance" between the target location and a record for a line or an area is actually the distance between the target location and the representative point that is stored in the data base. This can lead to unexpected results when the representative point (label point) for a line or area is not near the "center" of the feature.

".help (.h)"

Prints a help screen to the output device or file. Useful to have when using **v.db.rim** from a terminal, or when writing a script file of commands.

".input (.i) [file]"

The lines in the given file are read and processed as commands or data until an end of file is reached or until a .exit command is found. Input files may call other input files, by using this command, to a nesting depth of eight. Without a file name stdin is used as the input file.

".list (.l)"

Lists the available data bases in the current mapset search path.

".make"

Using the .make command you create a new data base in the current mapset by specifying the following items which define the screen (page) layout for displaying and printing the records, as well as the information fields:

```
1)  The fixed text part of the screen layout.
2)  The positions, types, and lengths of data fields.
```

Five fields must always exist in a data base; each of these field types may only occur once in a data base layout:

```
1) Type 's'  Sequence number field (a unique integer for each record).
2) Type 'x'  Easting coordinate of the representative point (a double float).
3) Type 'y'  Northing coordinate of the representative point (a double float).
4) Type 'v'  Vector type field (a text field).
5) Type 'm'  Reference Map field (an integer).
```

The other field types, which may occur in any combination and order, are:

```
6) type 'i'  An integer field.
7) type 'f'  A double precision float field.
             (always 2 decimal places used for output)
8) type 't'  A text field.
```

Each of the fields can be positioned anywhere within the screen layout, which has a limit of 19 lines by 80 columns. A maximum of 70 fields may be defined within this space. A field is specified in the screen layout by a tilde (~), a field type character, a field name and enough trailing tildes to fill out the desired field length.

Each line following the .make command is taken to define a line of the screen layout until a .end is reached. If a mistake is made on any of the input lines, the .make will fail. The .make information may be prepared in advance as a text file (this facilitates fixing mistakes) and the .input command can be used to read in this file. An example text file for a data base screen layout follows, with some explanatory notes and restrictions.

```
.make
              Hydrology Vector Database
              =========================
 Record #:    ~sSeqnum~       Feature Name:  ~tName~~~~~~~~~~~~
 Vector type: ~vVtype         Reference Map: ~mRefMap~
 North:       ~yNorth~~~~~    East:          ~xEast~~~~~~

 Updated:     ~tUpdate_Date~~~~~~~

 Comments:
              ~tComments.1~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
              ~tComments.2~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
              ~tComments.3~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 .end
```

Notes:

1) Any text not preceded by a tilde (~) character is taken to be part of the constant or fixed text portion of the form.

2) A field definition begins with a tilde (~) character immediately followed by a single character which indicates the data type of the field (s,x,y,v,m,i,f or t). Immediately following the data type character is the field name of 1 to 16 characters. Field names can be composed of any characters from the following set: [A−Z,a−z,_,0−9]; the RIM program and library do not distinguish upper and lower case in field names, so you should avoid making names which differ only in case. Field names may not begin with a numeral [0−9]. The rest of the field length is padded with tilde (~) characters to the length desired.

3) The minimum field width is three characters; e.g., "~tA". Be sure field widths for all fields are wide enough for the values and strings you expect to store there; e.g., UTM northings require at least 11 spaces.

4) For text fields it is possible to continue a field across more than one line. This is done by appending a .1 to the field name forming first portion of this "split field", a .2 for the second portion, etc. This text field splitting affects how information is organized for input and output; the composite text string is concatenated (unused portions of fields are retained as spaces) and treated as a unit for storage and queries to the data base.

".map (.m) [map_id map_name] | [−d map_id]"

Without arguments this command outputs a list of all the reference vector maps that are stored in the reference maps table. If a map number (map_id) and a vector map name (map_name) are given the vector map is found and added to the reference maps table in the data base. If the map number (id_num) is already in that table an error is issued and no action is taken.
Finally, to delete a map from the reference maps table, use the '−d' option followed by the map number (map_id). The map information for the given map number will be displayed and the user will be asked to confirm the deletion with a 'y'. Enter a 'n' (for no) if you do not want to delete that reference map.

Remember, that if you delete a reference map for which there are still records in the data base, you cannot make a new vector map (using the .vector_map command) that includes those records unless you put that number and vector map name back in the reference map table.

".output (.o) [file or | process]"

Causes all output (except some error messages) from **v.db.rim**, including that from the .print command, to go to the named path/file (may be a full or relative path name), or to be used as standard input by the process (a pipe). If no parameter is given, output returns to stdout, usually the user's terminal. An example of the pipe usage would be

```
.output | grep "easting" | wc −l > /tmp/my_count
```

A pipe is closed whenever the .output command is given again, or on a .exit command.

".pack (.pa)"

This should be used when numerous data records have been deleted or changed to recover disk space in the RIM binary data base files. It works by doing a .backup to a temporary file; moving the data base files to new names (*.bakdb*); running RIM to rebuild the data base; and, if the rebuilt data base can be opened and read, the temporary files are deleted. The user is informed if this process fails. Packing can only be done on an open data base located in the user's current mapset.

".print (.p) [−a | −l] "

This command outputs the full record for the records currently stored on the internal query list (result of last .query or .find). Without the flag, the screen layout format is used. With the −l flag, for list format, the field name followed by the contents are output one field per line. The −a flag also outputs in the list format but with

a .add line and a .end line surrounding each record printed; data files in this form can be read with .input, thus they form one kind of backup mechanism and can be used to transfer data (not the data base layout) from one GRASS system to another. The destination for the output is set by a previous .output command (default is stdout).

".query (.q) [−m | −w] [−a | −d]"

The .query command is used to retrieve records via an SQL−like request to RIM, including a user specified "where clause." All fields for each record meeting the selection criteria are retrieved.
The optional .query command line parameters cause records whose representative points are not in the region (−w) and/or mask (−m) to be rejected, so these conditions need not be tested in the "where clause". See .find for a full explanation of the command line options.

After the query command line, any number of lines (each no more than 80 characters) may be entered to define the SQL "where" clause. A .end line is required to finish the request and begin data retrieval. See examples below.

The "distance from" clause may also be used as an additional selection criteria exactly as described in the examples and notes for .find. It must be entered as a separate line to the query prompt.

The retrieved records may be printed at time of retrieval, rather than after the completion of the query command by including a .print (.p) line with the same options for print format as in the .print command (see above); e.g. .p −a to output in the "list add" format. The .print clause must be entered as a separate line to the query prompt. This feature is most useful when working with very large data bases where retreval time is significant. See example 2 below.

Example 1

```
query> where density < 20 and (date = "10/14/89"
query> or county eq "San Marcos")
query> .end
```

Example 2

```
query> where east <600000 and name like "*Jones*"
query> distance from record 12 3000
query> .print −a
query> .end
```

Example 3

```
query>.end
```

The where and distance from clauses are each optional. If both are omitted, only the mask and region on the .query command line restrict the search; if mask and region are also omitted, all records will be retrieved (Example 3). When querying for records the where clause is processed first, the current region and mask tested (if requested), then the distance from clause is applied; a record must pass all tests to be put on the internal query list (or appended or deleted) for output by other commands.

Notes: (Also see Notes for .find)

1. The retrieved records are stored on the internal query list in the order returned from the data base by RIM, not necessarily in sequence number order or the order the data was loaded. A "distance from" clause results in

a final sorting by proximity to the target.

2. See the RIM User's Manual and the s.db.rim manual page for additional information on the "where clause" in the "select" command, especially the quotes required for matching character string (text) fields, and the allowed comparison operators. (These are also described in SECTION TWO of this manual entry.)

3. In the example where clauses above, "density", "date", "county", east", and "name" are field names (column names in RIM) defined when the user initially makes the data base.

4. Each .query or .find resets the internal query list, unless the append or delete options are used. In no case is a record allowed to be duplicated on the query list.

".read_vect (.re) vector_map_name [attribute_field [text_field]]"

This command will read an existing GRASS vector map and create a data base record for each labelled area, line, and point. The sequence number field will automatically be generated starting from one greater than the highest current number in the database. If the optional *attribute_field* is provided it must be an integer field and it will be filled with the area, line, or point attribute label. If the optional *text_field* is provided and a category description file (a dig_cats file) exists for the vector map, the category descriptions will be copied into the given text field.
Once the records have been loaded by .read_vect, use .change to add data to other fields for those records.

Note: Only **labeled** areas, lines and points are imported from the vector map.

".remove"

This command, which requires a "y" as confirmation on the next line, entirely removes the three binary files which constitute your RIM data base. Use with care. Backup files must be removed individually by the user, if desired.

".show (.sh)"

This command is used to output the screen or page layout as defined for the current data base. It serves as documentation of the data base definition and as a reminder for field names, types and lengths. By using an editor to surround the output of .show with .make and .end lines, it can be used to reload the data base definition with .input.

".site_list (.si) file_name [field_name]"

This command writes the location coordinates (representative point) and a comment to the specified file in the site_list directory in the current mapset for each record currently selected. If the site file exists, the sites are appended to the current list, otherwise, a new site list file is created. A "field name" may be optionally specified; if so, the contents of that field (retrieved from the appropriate site record) are inserted as the comment (following a '#') in the site list; the record number is used if no field name is given. Such site lists may be used as input to s.db.rim.
A comment line is inserted in the site_list file with the current date and time and the name of the data base producing the site locations. The format used for each site is:

```
easting|northing|#number or comment
```

".tables (.t)"

Prints the table structure of the currently opened RIM data base. This is the same output generated by a "list *" command when running RIM manually. The information for the table named "data" is useful for review of the user's field definitions, and the table named "Referencemaps" contains the reference map information. The information in the two other tables is for internal use by **v.db.rim.**

".vector_map (.v) file_name [attribute_field [text_field]]"

This command creates a new binary vector map by copying the vectors associated with each record on the query list from the reference vector maps into the new vector map.
If the optional *attribute_field* parameter is provided, the areas and lines in the new vector map will be labeled from the given integer (i, m or s type) field value for each record. You may supply a fixed integer value (such as 1, 908, −7, etc.) instead of a field name; each line written to the new map will be given this constant attribute/label.

If the optional *text_field* is provided, it will be used to build a category description file (dig_cats file) for the vector map. Instead of a field name, a constant text string of up to 100 characters may be given, if enclosed in single or double quotes; this string is used as the category description for each line written to the new vector file.

The header of the new binary vector file will contain the current date and indicate that **v.db.rim** was used to create the vector map.

The topology information (the dig_plus file) is not automatically built for the new vector map and the user must run **support.vect** to do so before **v.digit** can be used to edit the vector map or some other programs can be used. The vector map can immediately be displayed, from within **v.db.rim** by issuing the following command (assuming you have a graphics monitor selected):

```
!d.vect file_name c=color
```

SECTION TWO −− THE INTERACTIVE VERSION MENUS AND COMMENTS

SYNOPSIS

v.db.rim

DESCRIPTION

v.db.rim The interactive version allows you to create, manage and query information about vectors across the landscape on a data base through a series of menus (VASK screens) explained below.

THE MAIN MENU

Below is the main menu. Option 1 is the default. Note the status line at the top of the menu, and the fact that 8 records have been selected by the last find or query operation (between items 2 and 3). Note, also, that CTRL−C can be used to exit from this menu (and most other menus in the program) back to the GRASS prompt. The specifics of each menu choice are described below. Except for 6, and mouse options in 3 and 4, each choice has a direct counterpart in the command version.

```
v.db.rim                MAIN  MENU                Version 1.4
  Data base <rivers> in mapset <kittco> open. 325 records.
```

1 Open a data base 2 List available vector data bases −−−−−−−− Retrieve/Output Site Records (8 currently) −− 3 Find records in proximity to a Target point 4 Query to select records (SQL) 5 Show selected records on Terminal 6 Display maps/selected vectors on graphics terminal 7 Output selected records to Printer or File 8 Create vector/site maps from selected records −−−−−−−−−−−− Add/Edit Site Records −−−−−−−−−− 9 View a single record 10 Add a record 11 Change a record 12 Delete a single record or all selected records −−−−−−− Other functions −− Shell Command −− Exit −−−−−−−−−− 13 Make a new data base & Management Functions 14 Execute a shell command 0 Done −− Exit from v.db.rim AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE (OR <Ctrl−C> TO EXIT THIS PROGRAM) 1. Open a data base. If a data base is already open, it is closed before the requested one is opened. Only data bases in the user's current mapset may be modified; others are opened in read−only mode; this will be indicated on line 2 of the menu.

2. List available data bases. For each mapset in the current GRASS mapset search path, the names of the existing vector data bases are listed.

3. "Find" records in the data base relative to a specified target location. This is used to select records based on proximity to the target and, optionally, records within the current region and, optionally, records falling in active cells within the current GRASS mask. The label point coordinates are used for these spatial tests. Two modes of targeting are provided: the N records closest to the target, and all records within (or outside) a circle of specified radius from the target. The FIND/QUERY TARGET MENU discussed below accepts region/mask/target specifications from the user. The selected records are then displayed one at a time until CTRL−C is entered; then other operations, choices 5−8, can be done with these records. The line on the menu between 2 and 3 shows the number of records currently selected by choices 3 or 4.

4. "Query" records in the data base using an SQL−like "where clause," including specifications for region/mask/target (circle only) as in 3, above; see FIND/QUERY TARGET MENU section below. The where clause can test for ranges or matches for numeric data base fields, or matches on full strings or substrings for text fields. The selected records are then displayed one at a time until CTRL−C is entered; then other operations, choices 5−8, can be done with these records. This clause is entered on a QUERY COMMAND MENU described below.

The where clause may use parentheses ( ) to control the order of comparisons. Field names are not case sensitive within where clauses. The following comparison operators are valid for all types of fields:

```
eq   or   =            ne   or   <>
ge   or  >=            le   or   <=
gt   or   >            lt   or    <
```

String comparisons are case sensitive and are done character by character. Substrings comparisons may be done with the "like" operator as in:

```
where name like "*Jones*"
```

Note that the string being tested against the name field for each record is in quotes (single or double) and that wild card comparisons can be done in the standard way with '*' and '?' characters.

Logical comparisons may also be combined with those operators above. The permitted logical operators are:

```
and       or       not
```

The following complex example should be examined. The line breaks can occur between any tokens (words, values, operators), except within quoted strings.

```
where (name like "*Jones*" or name = "Smith")
and ( ( site < 300 and not (site = 251 or site eq 15) )
or east < 601000 )
```

5. This choice will display the records resulting from the last find/query one at a time on the terminal. Use ESC or enter a number to display another record and CTRL−C to end the display.

6. If a graphics monitor is active, the selected vectors will be displayed. The user may choose to erase the screen; display cell, vector, and/or site maps; and/or display the selected vectors from the data base; these maps are requested through the following interactive screen. Just enter ESC to skip this step. If no data base vectors are currently selected, that section of the menu will not appear; but the menu can still be used to display the other types of maps.

```
               SELECTION MENU FOR ITEMS TO DISPLAY

Enter cell and/or vector map names, if desired

 _______________  Cell file to display
 _______________  Vector file to display in color: _________
 _______________  Site list to display
                  Dpoints with: size=3_ type=box____ color=white____
                _ Display currently selected vectors (enter x)
                  Dvect red______
                _ Erase graphics screen (enter x)
                  Derase  black____

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

7. This selection results in a screen prompting for the name of the file to output the selected records to, and for optional formatting selection. If the file name is lp, the site records are sent to the printer. The optional formatting choices are for export of data in list and add formats; see the .print description in SECTION ONE of this manual page for information and examples.

8. Using this choice you can create a new GRASS vector map consisting of the vectors for the currently selected records, and/or a site list consisting of the representative points (label points) for the currently selected records, in your current mapset. A short menu prompts for the map names and other information.

If a vector map name is given you can choose an integer field (i, s, or m types), or a fixed value, to write as the label value for each vector in the dig_att file for the new map. You may also specify a field name (any type), or a fixed text string in single or double quotes, to write as the category description in the dig_cats file for the new map.

If you give a site list name, you can specify the name of a field (or fixed text string in quotes) to be used for the "#comment" in the site list (the record number is the default field). The current date and time, and the names of the mapset and data base in use are entered as an information line in the site_list file. Note that you can create a new site list or append to an existing site list, or both.

A variety of cell maps can be produced from a **v.db.rim** data base by creating new vector files then using the *v.to.rast* program, and by writing site_lists with different fields as "comments" then converting the site_lists to cell maps with *s.menu*.

9. Choices 9−12 operate on only a single record and do not use or modify the internal list of records selected by find/query (choices 3 or 4). Choice 9 is the way to view a single record, selected by record number. After

viewing, ESC will allow entry of another site number and CTRL−C will exit to the main menu.

10. Use this selection to add a new record to the data base. (A new record is one whose number does not currently exist in the open data base.) After making this selection, the data base layout will be displayed and you should enter the available information appropriate to each field; the only required entry is the site (record) number field. If values for numeric fields are not entered, zero values will be stored. Unused portions of text fields are stored as strings of spaces.

11. After making this selection and specifying the record number to change field information for, the data is entered as for choice 10, except that the record number cannot be changed. (The command version of the program has provision for making bulk changes after a find or query; see .change.)

12. To delete a single record, enter its number when requested. All records chosen by the last find/query operation may be deleted by entering "list" in place of the record number. BE CAREFUL with this, *deleted records are really gone.*

13. This choice starts a new menu with less commonly used functions. See MANAGEMENT MENU section below.

14. The program will prompt you for one−line Shell Commands until you enter just a <RETURN> to return to the main menu.

FIND/QUERY TARGET MENU

This is the screen to set up the region/mask/target information for the find choice (3) and the query choice (4), except that item B is omitted for choice (4). If a graphics monitor is not active, the "mouse" item is omitted from the menu; and, if a mask is not currently set, that line is omitted.

The choice to append or delete the selected records will only be given after a successful find or query has stored some records on the internal record list. When appending records, duplicates of those previously selected will be discorded−−they will not be stored a second time. If neither append nor delete is selected, the find or query will begin a new internal record list and the previous contents will be lost.

The choices entered on this example screen will result in all the records within a 1500 (meters) radius of the target point (to be chosen with the mouse) being selected and stored on the internal record list by find or query. They are sorted and stored in order of proximity to the target. If a specific record is used as the target, it's representative point (label point) is the target coordinates, and it is always placed first in the retrieved list. If a mouse is chosen to select the target point, a menu to display reference maps is presented, exactly as in choice (6), prior to actually activating the mouse.

```
        QUERY/FIND:  REGION/MASK/TARGET SELECTION MENU
  Data base <arch> (READONLY) in mapset <PERMANENT> open.  25 records.
    Mark requests with 'x' and enter required values.

                Respect current region   x

                Respect current MASK     x
               (forces current region)

A.  Find all sites within (or outside) a circular target   x
              and give the radius (negative for outside)  1500.00_____
                          OR
```

```
B.  Find a number of sites nearest a point   _
         and the number of sites requested   _________

    After selecting A or B, complete one(!) of these:
        1. x to select target point with mouse    x
        2. Enter site number for target point     __________
        3. Target coordinates             east    0.00________
                                          north   0.00________

 Append(a) or Delete(d) to the current FIND/QUERY list  _
 Reset to default choices for this menu _

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

QUERY COMMAND MENU

The following screen completes the information for a query (choice 4). It may be left blank if no "where clause" is required. After a successful query, the selected records are displayed one at a time by hiting ESC; CTRL−C will quit the display and return to the main menu where several choices of operation on the retrieved sites are offered. The SQL "sort by" clause may also be used after the where clause to control the order selected records are presented; however, if option A or B in the TARGET MENU has been selected it causes sorting by proximity to the target location which will override the order produced by the "sort by" clause.

```
        QUERY COMMAND CONSTRUCTION SCREEN
  Data base <wells> in mapset <grant> open.  25 records.
 The SQL select query will use the current region
 and a target clause of 'distance from 596463.15 4919041.88'

where date = 10/16/89 and ppm_Cr gt 10_____________________
_____________________________________________________________
_____________________________________________________________
_____________________________________________________________
_____________________________________________________________
_____________________________________________________________
_____________________________________________________________
_____________________________________________________________

(Enter .show on a line to review screen layout and field names.)

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

MANAGEMENT MENU

Choice 13 from the main menu presents this menu. Each item is discussed below.

```
    v.db.rim      DATA  BASE  MANAGEMENT  MENU
Data base <fires> in mapset <Yellow> open.  250 records.

  1   Make a New Data Base in Current Mapset
  2   List Available Data Bases
  3   Remove (PERMANENTLY) Data Base from Current Mapset
  4   Recover a Data Base from a RIM ASCII File
  5   Show Screen Layout of Current Data Base
  6   Backup (UNLOAD) Data Base to RIM ASCII Format File
```

```
  7   Pack the Current Data Base
  8   Read a vector map into the Current Data Base
  9   Execute a Bourne Shell Command Line

  0   Return to Main Menu

 0_ Your selection

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

1. Use this choice to create a new data base in the current GRASS mapset. See section below on MAKE A NEW DATA BASE.

2. List available data bases. Like 2 on main menu.

3. Delete an entire data base from the current mapset. The name of the data base and additional confirmation of the action are prompted for. **Be careful!**

4. Choice 6 allows backup of the definition and data parts of a data base to a transportable text file. To rebuild (or build for the first time) a **v.db.rim** data base from one of these text files do the following steps:

```
   # see if the rim directory exists.
ls $LOCATION/rim/vect
   # if the directory was not found, make it.
mkdir $LOCATION/rim
mkdir $LOCATION/rim/vect
   # change directory to it.
cd $LOCATION/rim/vect
   # have rim build and load the binary data base files.
rim
RIM> input '/path/to/your/textfile'
RIM> exit
```

The data base is thus created in the current mapset. Several **v.db.rim** commands should be run to verify the integrity of the newly created data base.

5. This merely shows the screen layout of the currently open data base. It is a useful way to quickly see the layout and review the field names and types.

6. When backing up to a text file, the RIM UNLOAD command is run with the output directed to a file of the user's choice. See 4 above. It is wise to do this operation after extensive changes or additions of data records. The resulting text file can be written to tape for preservation, or shared with other GRASS systems, if desired. Data bases may also be backed up by copying the three binary files which comprise the data base to a different directory with the UNIX cp command.

7. After deleting and adding a large number of site records, some "wasted" disk space will be present in the binary data base files. This procedure will perform an unload and a reload automatically to recover this unusable disk space. If there is any problem reopening the data base after packing, the user is notified and can recover in various ways depending on the backups which have been done.

8. Data (records) may be loaded into a data base from an existing GRASS vector map. This procedure will prompt for the vector map name and then add a record to the currently open data base for each labeled(!) line in the vector field. The user may also enter the name of an integer field in which to store the label (from the dig_att file) for each vector, and a text field in which to store the descriptive text from the dig_cats file for

each vector. The record number, vector type, map number and location coordinate fields (s,v,m,x and y types) are automatically loaded for each site record by this procedure; other fields may be later edited with the "change" function.

9. This choice is the same as choice 14 on the main menu.

MAKE A NEW DATA BASE

After entering the name of the new data base you wish to create (7 characters maximum), you then decide how to input the information required. This input may be from a text file, or may be entered directly using the editor of your choice; the former is recommended. See discussion in .make in SECTION ONE.

NOTES

1. A "date" type field should be added to future versions. This version only allows storing of dates as strings (unless the user codes them to integers), and thus only string type searches can be made for dates.

SEE ALSO

s.db.rim

The RIM Users manual by Jim Fox, Academic Computing Services, Univ. of Washington. See especially Appendix B on redistribution of RIM.
The RIM Installers manual.

AUTHORS

David Satnik and James Hinthorne, GIS Laboratory, Central Washington University.

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

NAME

v.digit2 − A menu−driven, highly interactive map development program used for vector digitizing, editing, labeling and converting vector data to raster format.
(GRASS Vector Program)

SYNOPSIS

v.digit2

DESCRIPTION

The GRASS program *v.digit2* is a menu−driven, highly interactive map development program used for inputting analog map data into a GRASS vector format. *v.digit2* contains programs for vector digitizing, editing, labeling, windowing, and converting vector data to GRASS raster format.

v.digit2 consists of two parts: an initialization procedure, and a multiple menu−driven environment.

1. The initialization procedure involves choosing a digitizer, choosing the name of a vector map layer to digitize, and, if needed, registering a map to the digitizing surface.
2. The second part of *v.digit2* is a multiple−menu environment in which digitizing, editing, labeling, and other options are available. It is within this second portion of *v.digit2* that all vector creation occurs.

NOTES

v.digit2 was written to optimize digitizing speed and performance. It has a convenient graphic display format and very convenient windowing capabilities. Features are color−coded for ease of identification and verification. Different color schemes may be user−chosen to customize a digitizing session.

Area, line, and point features may be digitized in both "stream" and "point" modes. Both a mouse and a digitizer may be used to perform windowing functions. Labeling and editing are done from within *v.digit2*, rather than through a separate set of programs.

v.digit2 has capabilities that allow users to convert vector map layers to GRASS raster format, to overlay already existing vector map layers onto the feature being digitized, and set a multitude of parameters to customize a digitizing session.

v.digit2 may be used with or without a digitizer. Many options are available if no digitizer is used.

All options available within *v.digit2* are contained within ten menus. Movement from menu to menu is done by chosing the movement options specified at the bottom of each menu.

SEE ALSO

v.import
v.in.ascii
v.out.ascii
v.out.rast
v.support

AUTHORS

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

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

NAME

v.digit − A menu−driven, highly interactive map development program used for vector digitizing, editing, labeling and converting vector data to raster format.
(GRASS Vector Program)

SYNOPSIS

v.digit

DESCRIPTION

The GRASS program *v.digit* is a menu−driven, highly interactive map development program used for inputting analog map data into a GRASS vector format. *v.digit* contains programs for vector digitizing, editing, labeling, windowing, and converting vector data to GRASS raster format.

v.digit consists of two parts: an initialization procedure, and a multiple menu−driven environment.

1. The initialization procedure involves choosing a digitizer, choosing the name of a vector map layer to digitize, and, if needed, registering a map to the digitizing surface. It is important to specify the map scale.
2. The second part of *v.digit* is a multiple−menu environment in which digitizing, editing, labeling, and other options are available. It is within this second portion of *v.digit* that all vector creation occurs.

NOTES

v.digit was written to optimize digitizing speed and performance. It has a convenient graphic display format and very convenient windowing capabilities. Features are color−coded for ease of identification and verification. Different color schemes may be user−chosen to customize a digitizing session.

Area, line, and point features may be digitized in both "stream" and "point" modes. Both a mouse and a digitizer may be used to perform windowing functions. Labeling and editing are done from within *v.digit*, rather than through a separate set of programs.

v.digit has capabilities that allow users to display raster maps in background (Customize menu, Backdrop), to overlay already existing vector map layers onto the feature being digitized (Customize menu, Overlay), and set a multitude of parameters to furtherly customize a digitizing session.

v.digit may be used with or without a digitizer. Many options are available if no digitizer is used, i.e. digitizer support by mouse.

All options available within *v.digit* are contained within ten menus. Movement from menu to menu is done by

chosing the movement options specified at the bottom of each menu.

Digitizer support

GRASS is using device independent code written originally by John Dabritz formerly of the Forest Service, which uses ascii description files to define how to communicate with the digitizer. The result is that we should be able to support just about any digitizer out there now with a minimum of work. Refer to the "digitizer definition manual" for information on the format of these files.

SEE ALSO

GRASS 4.0 map digitizing manual: v.digit

v.import, v.in.ascii, v.out.ascii, v.spag, v.out.rast, v.trim, v.support

AUTHORS

David Gerdes,
Mike Higgins,
U.S. Army Construction Engineering Research Laboratory
Improvements by Huidae Cho

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

NAME

v.distance − Calculates distance from a point to nearest line or point in vector layer.
(GRASS Vector Program)

SYNOPSIS

v.distance
v.distance help
v.distance map=*name* [**east_north=***east,north*[*,east,north,...*]

DESCRIPTION

v.distance uses a binary vector file (map=*name*) and user specified coordinates to calculate the distance from the given point(s) to the nearest line or point in vector layer.

Parameters:

map=*name*
Name of a binary vector file.
east_north=
One or multiple coordinate pairs for query

v.distance can be run either non−interactively or interactively but interactive user input is required to define the points to calculate the distance from.

EXAMPLE

To calculate the distance of given sites to the next road, run:

```
s.out.ascii −d archsites | v.distance roads
```

To calculate the distance of given sites to the next road and store the list as new sites map with distance attribute and vector attribute number, run:

```
s.out.ascii −d archsites | v.distance roads | s.in.ascii fs='|'
sites=dist
```

The output structure is: `E|N|sitesID|dist|vect_line_ATT`

e.g. `1680949.350649|5100196.753247|31|16.854858|22002`

SEE ALSO

r.distance

AUTHOR

Janne Soimasuo 1994, Finland
University of Joensuu, Faculty of Forestry, Finland

Cmd line coordinates support: Markus Neteler, ITC−irst, Trento, Italy

Last changed: $Date: 2002/02/08 13:46:12 $

NAME

v.export − Converts binary vector files into formatted ASCII files for transfer to other computer systems.
(SCS GRASS Vector Program)

SYNOPSIS

v.export
v.export help

DESCRIPTION

This program performs all of the processes that are needed to convert binary vector files into formatted ASCII files.

It also creates support files:

− an attribute flat file, which contains information for each area in the DLG file −− i.e., the DLG area number, the GRASS area label, and the GRASS category code (only created when exporting in DLG format);
− an attribute file which contains the information from the *dig_att* file (only created when exporting ASCII vector format).

EXPORT FILES
After entering the command *v.export*, the user will be asked which type of file to export:
Exports from GRASS Vector (*v.digit*) Format
1 − ASCII DLG file from GRASS Vector Format
2 − ASCII DIGIT file from GRASS Vector Format
3 − ASCII SCS−GEF file from GRASS Vector Format
4 − ASCII ARC/INFO file from GRASS Vector Format
5 − ASCII DXF file from GRASS Vector Format
If numbers 1−4 are chosen, *v.export* will respond with a request for the vector file name. After the user enters the file name the program proceeds to create the respective output format files.

GRASS Vector to DLG File
Converts binary vector files (such as those created by *v.digit*) to a DLG file and creates the attribute file. Both files are placed in the *dlg* directory under a user selected name; the attribute file has .att appended.

GRASS Vector File into ASCII Vector File
Converts a binary vector file into a readable ASCII file. Both files are placed int the *dig_ascii* directory under the same name as the given vector file, the attribute file has .att appended.

GRASS Vector to SCS−GEF File
Converts binary vector files to SCS−GEF files. Creates the SCS−GEF header, lines, text, and feature files. All

files are created and placed in a $LOCATION/gef directory as a single UNIX file under a user selected name. The following is the SCS−GEF file structure:

```
header record 1
   |      |
header record n
−head
line record 1
   |      |
line record n
−line
text record 1
   |      |
text record n
−text
feature record 1
   |      |
feature record n
−feat
```

The user will be required to use standard UNIX commands to separate this file into individual files as required by SCS−GEF specifications.

GRASS Vector to ARC/INFO(generated) File
Converts binary vector files to a "ARC ungenerate" format. A GRASS vector file to be exported to ARC/INFO must be either a line coverage (must contain only lines) or a polygon coverage (must contain only area edges). Both "ungenerate lines and points" files are created and are placed in a $LOCATION/arc directory under a user selected name.

The binary vector name will be used to name the various files that will be created for export to ARC/INFO. In the case of a labelled polygon coverage, the following three files will be created: a lines file with the suffix .lin, a label−points file with the suffix .lab, and a label−text file with the suffix .txt.

In the case of a line coverage the following two files will be created: a lines file with the suffix .lin, and a label−text file with the suffix .txt.

An unlabelled polygon or line coverage will result in a lines file (.lin suffix) only. See the DATA FILE FORMATS section of v.import for more information on these files.

GRASS Vector to DXF file
Converts binary vector files to a "DXF" format.

NOTES

Support files must be built using the GRASS program *v.support* before exporting any vector file.

Other ASCII formats are useful when importing/exporting data into and from GRASS. Such data files should be in ASCII format when transferred.

SEE ALSO

v.digit
v.import v.out.arc

v.out.ascii
v.out.dlg
v.out.dlg.scs
v.out.dxf
v.out.scsgef
v.support

AUTHOR

R.L. Glenn, USDA, SCS, NHQ−CGIS

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

NAME

v.extract − Selects vector objects from an existing vector map and creates a new map containing only the selected objects.
(SCS GRASS Vector Program)

SYNOPSIS

v.extract
v.extract help
v.extract[−d] **type**=*name* **input**=*name* **output**=*name* **new**=*value* [**list**=*range*] [**file**=*name*]

DESCRIPTION

v.extract allows a user to create a new vector map layer from an existing vector map layer. User provides the program with category numbers, input vector file name, an output vector file name, and the type of input object. There is an option (d) to dissolve common boundaries between adjoining map areas in the same category list. The user may specify a file containing category numbers or labels.

Note:
The dissolve option will work on only those areas which are in the given category list. If a map area is inside (island) a listed category area and is NOT in the given category list, its boundaries are output to the resultant map.

COMMAND LINE OPTIONS

Flags:

−d
Dissolve common boundaries (default is no).

Parameters:

type=*name*
Select area, area edge, line, or site.
Options: area, edge, line, site
input=*name*
Input vector map name.
output=*name*
Output vector map name.
new=*value*
Enter 0 (keep original category) or a desired NEW category number.
list=*range*

Category ranges.
For example: 1, 3−8, 13
For example: Abc, Def2, XyZ

file=*name*

Text file name for category range/list .

EXAMPLE

v.extract −d list=1,2,3,4 **input=**soils **output=**soil_groupa **type=**area **new=**1
Produces a new vector area file *soil_groupa* containing 'area' boundaries from *soils* with area category numbers of 1 thru 4; any common boundaries are dissolved, and all areas of the new map will be assigned category number 1.
v.extract input=soils **output=**soil_groupa **type=**area **new=**1 **file=**sample
Produces a new vector area file *soil_groupa* containing 'area' boundaries from *soils*. No common boundaries are dissolved, all areas of the new map will be assigned category number 1. The numbers can be found in the file sample of the current directory.
The format for "sample" is:
1
33
45
56

AUTHOR

R.L. Glenn, USDA, SCS, NHQ−CGIS

Last changed: $Date: 2003/08/20 08:10:13 $

NAME

v.geom − Computes constrained MinMax−Angle triangulation, constrained MinMax−Slope triangulation, constrained MaxMin−Height triangulation, constrained planesweep triangulation, constrained delaunay heuristic, and convex hull of sites and prescribed edges in 2 and 2 1/2 dimensions.

SYNOPSIS

v.geom
v.geom help
v.geom input=*name* **output**=*name* [**precision**=*value*] [**operation**=*name*]

DESCRIPTION

v.geom takes a vector file as input and computes various triangulations respecting the input edges, or the convex hull of the sites. The z−coordinate is read from the description field if it is specified, otherwise 0 is assumed. The z−coordinate is used for the MinMax−slope triangulation. For all other computations the z−coordinate is ignored.

The MinMax−angle triangulation is the triangulation for the sites which minmizes (lexicographically) the sorted vector of all the angles of triangles in the triangulation. The constrained version also minimizes this vector but under the constrained that prescribed (i.e. input) edges are part of the final triangulation. The MaxMin−height and MinMax−slope triangulations are similar. The algorithms used for the computations are not heuristics, they actually achieve the optimum.

We use a simple extension of the algorithm used to compute the Delaunay triangulation in *s.geom* to compute a triangulation which can be considered an approximatoin of the constrained Delaunay triangulation. However, this is only a (bad?) heuristic.

The output is saved in vector file format. Edge labels of input edges will also be attached to the corresponding output edges.

OPTIONS

Parameters:

input=*name*
Input vector (level 2) file.
output=*name*
Output vector file.
precision=*value*
Number of significant positions after the decimal point. (default is 0).

operation=*name*

One of the following: *sweep*, *delaunay*, *angle*, *height*, *slope*, *hull*. These correspond to the constrained planesweep triangulation, constrained Delaunay heuristic, constrained MinMax−angle triangulation, constrained MaxMin−height triangulation, constrained MinMax−slope triangulation, and convex hull, respectively. (default is constrained planesweep triangulation).

NOTE

The computation times for the various operations depends strongly on the algorithm used.

The plansweep triangulation and convex hull computation require *O(n log n)* operations in the worst case [Ed]. The Delaunay heuristic needs *O(n^2)* time in the worst case, however it performs much faster in practice. The MinMax−angle and MaxMin−height triangulations need *O(n^2 log n)* operations [BeEd, EdTa], and the MinMax−slope triangulation needs *O(n^3)* operations [BeEd].

Internally, the coordinates of the sites are stored in fix−point format. Therefore, the number of decimal digits cannot exceed 64 bit (or approx. 16 decimal digits).

It is important that the input vector file is reasonably "clean". The current implementation of *v.geom* takes care of loops (i.e. zero length edges), duplicate edges, and edges which are collinear and overlapping. However, because of the internal representation of coordinates in fix−point format it can happen that certain anomalies are introduced. For examples edges can cross although they don't in the input data. Currently, the program does not test for such cases. If it occures one of two situations will happen. Either, the planesweep algorithm terminates with a segmentation fault, or it will loop forever. For the data where we had problems these problems could be eliminated if we first used *v.spag*.

BUGS

Some fields of the header in the output file are not properly set.

REFERENCES

[BeEd] M.Bern, H. Edelsbrunner, D. Eppstein, S. Mitchel, T.S. Tan. Edge Insertion for Optimal Triangulations. *In Proc. 1st Latin American Sympos. Theoret. Informatics 1992*, 46−−60.

[Ed] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer−Verlag, Heidelberg, Germany, 1987.

[EdSh] H. Edelsbrunner, N. R. Shah. Incremental Flipping Works for Regular Triangulations. *In Proc. 8th Ann. Sympos. Comput. Geom. 1992*, 43−52.

[EdTa] H. Edelsbrunner, T.S. Tan and R. Waupotitsch. An O(n^2 log n) Time Algorithm for the MinMax Angle Triangulation. *SIAM J. Sci. Statist. Comput. 13 1992*, 994−1008.

SEE ALSO

s.geom

AUTHOR

Roman Waupotitsch

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

NAME

v.import − SCS user interface to GRASS import programs.
(SCS GRASS Vector Program)

SYNOPSIS

v.import
v.import help

DESCRIPTION

This program performs all of the processes that are needed to convert ASCII DLG files and ASCII vector files into binary vector files. It also creates support files, the dig_plus file and the dig_att file (only created when importing DLG). The dig_plus file contains topological information obtained by analyzing the vector file. The dig_att file contains attribute information 'stripped' from the DLG file. This dig_att file is created for vector files by the labeling function of the GRASS *v.digit* program.
This command must be run interactively.

IMPORT FILES
After entering the command v.import, the user will be asked which type of file to import and create support files for:

```
Imports to GRASS Vector Format and Creates Needed Support Files
   1  −   Ascii DLG file  to GRASS Vector Format
   2  −   Ascii VECTOR file  to GRASS Vector Format
   3  −   Binary VECTOR file to GRASS Vector Format
```

ASCII DLG File to GRASS Vector
Converts ASCII DLG files (such as those created in GRASS) to a vector file and creates the *dig_plus* and *dig_att* support files. The user is asked several questions:

1. The name of the DLG data file.

   NOTE: It should be available in the $LOCATION/dlg directory. If the DLG data has an attribute flat file, it should also be in $LOCATION/dlg.

2. Determine if this map is composed of Area or Line information.

   NOTE: Some machine−processed DLG files do not make the distinction between lines and area edges. For example, in a roads map, where the desired information is line data, a downtown block surrounded by roads may be processed as an area. Because of this, the user is asked to choose whether to give precedence to areas or lines. If precedence is given to lines, the user should be aware that any lines that bound unlabeled areas in the DLG file will be stored as line data. Any unlabeled areas

would therefore be lost (this is only a concern when areas are unlabeled, labeled area information will be retained). If precedence is given to areas, lines will be stored as boundaries to areas that are unlabeled.

3. Determine if you want to snap nodes to other nodes within a threshold.

   NOTE: BE CAREFUL!!! This threshold is calculated using the scale of the original DLG or *v.digit* file. If the threshold is too high, excessive snapping may occur, destroying the file. In general, users seldom need to snap nodes. If snapping of nodes is desired, the user may want to run *v.support* separately. *v.support* allows the user to set the snapping threshold.

4. Does the DLG data contain GRASS category codes?

   NOTE: Most non−GRASS computer systems will not be able to provide the necessary codes. The flat attribute file serves this purpose. If the answer to this question is NO:

   1. Enter a SUBJECT MATTER file name. A subject file will be used to assign GRASS category codes to the DLG data. It is structured the same as a *dig_cats* category file. It is suggested that a SUBJ directory be created in the GRASS location and a file containing all DLG attribute text labels by category be created. This will be required to provide consistency across several maps (quads) within one location. The user may use the *vi* text editor or the SCS macro *make_subject* to create it.

   2. Enter an ATTRIBUTE file name.

      This is the name of the flat file which will accompany a DLG from a non−GRASS system. This file contains all of the DLG area numbers with a corresponding text label.

   3. Is the DLG data from an ARC/INFO system.

      ARC/INFO DLG data is handled in a slightly different manner.

   4. Does The DLG contain a Universe Polygon.

      Some DLG files may or may NOT have this and processing will be required to handle each case differently.
      This process is done in three phases:

      1. If the DLG does NOT contain category codes, then a category file from the attribute file is created. Then the ASCII dlg file is converted to a binary *dlg* file.

         − OR −

         If the DLG does contain category codes, then the ASCII DLG file is converted to a binary DLG file.

      2. The binary *dlg* file is converted to a binary vector file, and the *dig_att* support file containing attribute information is created.

      3. The *dig_plus* support file is created by analyzing the vector file for topological information.

ASCII Vector File into GRASS Vector
Converts ASCII *v.digit* files (which are located in *dig_ascii* directory) into binary vector files and creates the *dig_plus* support file. Since a vector file keeps the distinction between lines and area edges, the user is not asked to give precedence to either. However, the user will be asked if the user wants to snap from nodes to other nodes within a calculated threshold.

This process is done in two phases:

1. The ASCII vector file is converted to a binary vector file, and the *dig_plus* support file is created.

2. The *dig_plus* support file is created by analyzing the vector file for topological information.

Binary Vector File to GRASS Vector
Creates the *dig_plus* support file.

This process is done in one phase: The *dig_plus* support file is created by analyzing the vector file for topological information.

pe of file can be created created in ARC/INFO by using the *Points* subcommand of the *Ungenerate* command. The first number on each line is a label−point ID number, the following two are easting northing coordinates for the label−point.

```
1  711539.875000 4651743.000000
2  711429.000000 4650632.000000
3  711027.625000 4651736.000000
4  711022.625000 4651519.000000
5  710482.750000 4651494.000000
6  710474.500000 4651667.000000
7  709269.750000 4651018.000000
8  709726.500000 4651604.000000
9  708926.375000 4651195.000000
10 708567.500000 4651644.000000
11 708272.750000 4651407.000000
END
```

LABEL−TEXT FILE, also known as *xxx.txt* file. This type of file can be created in ARC/INFO by using the *Display* command.

```
1   −2.30228E+07   19,399.848     1      0      0     0
2     81,079.875    1,678.826     2      1     15     3
3    955,952.500   10,229.637     3      2     19     8
4     41,530.875      926.887     4      3     17     3
5     87,900.188    1,900.909     5      4     13     3
6    166,125.125    3,512.950     6      5     15     3
7     29,460.563      824.968     7      6     17     3
8    1022769.875    9,105.707     8      7     20     9
9     51,385.500    1,075.638     9      8     17     3
10   376,834.875    4,470.027    10      9      9     2
11    65,802.688    1,575.088    11     10     16     3
```

NOTES When importing a polygon coverage, the program finds the label−point ID in a label−text file by looking for the second column in the file that contains a "1" on line 1, and a "2" on line 2.
If you are missing a label−points or a label−text file you can still import ARC/INFO data (but none of your lines or areas will be labelled).
ASCII DXF Format Files to GRASS Vector
Creates the *dig_plus*, *dig_att*, and *dig_cats* support files.

ASCII TIGER Format Files to GRASS Vector
This program imports Census line features from TIGER records type1 and type2 into GRASS vector format. Both pre−Census and post−Census data formats can be used. Specific Census Feature Class Codes (CFCC) can be extracted completely or in various combinations. These codes are described in the TIGER/line Census Files 1990 documemtation available from the Bureau of the Census. An additional feature code consisting of the three letters "BOU" may also be specified to extract a county boundary. Condensed Record 1 files may be imported with the −c flag. These files should be identified with a trailing "x" character on the filename.
The TIGER files must in sorted order before being used. This can be done by using the following command:

```
sort TGR12113.F21 −o t12113.1
sort TGR12113.F22 −o t12113.2
```

For consistency the sorted file should be written as above. It should consist of a 't' followed by the State and County FIPS code, then a '.' and then a value to identify the record number. −−>

SEE ALSO

v.in.dlg.scs
v.in.dlg
v.in.ascii
v.in.arc
v.in.dxf
v.in.tiger

AUTHOR

R.L.Glenn, USDA, SCS, NHQ−CGIS

Last changed: $Date: 2003/08/20 08:10:13 $

NAME

v.in.arc − Imports vector data in ARC/INFO ungenerate format into GRASS.
(GRASS Vector Data Import Program)

SYNOPSIS

v.in.arc
v.in.arc help
v.in.arc [**−n**] **type**=*name* **lines_in**=*name* [**points_in**=*name*] [**text_in**=*name*] **vector_out**=*name* [**idcol**=*value*] [**catcol**=*value*] [**attcol**=*value*]

DESCRIPTION

The user may wish to use GRASS programs on data files that were created by other GISs. To do this, the user must first convert data files in these systems' formats to GRASS's file format. Bringing data from other systems into GRASS is termed file *import*. Sending GRASS data files out into other systems' formats is termed file *export*.

A variety of GRASS programs exist to import and export data to and from GRASS. The *v.in.arc* program will convert vector data in ARC/INFO's "Generate" format to GRASS's vector file format, and bring it into the user's current GRASS mapset. The files to be imported to GRASS must first have been exported from ARC/INFO using the ARC/INFO *Ungenerate* command, and may represent either linear features ("line coverage") or areal features ("polygon coverage"). The *ARC/INFO User's Guide* describes how files containing linear and polygonal features can be exported from ARC/INFO, in a section detailing the use of the *Ungenerate* command.

Note 1: The ARC coverage must be set to single precision before running *Ungenerate*. If it is not, first copy it to another coverage that is set to single precision, then run *Ungenerate*.

Note 2: To use for text attributes, numeric fields with values >999 may contain no commas or TABs. Also, the first record must have all fields filled.

OPTIONS

Program parameters and the flag have the following meanings.

Flags:

−n

Neatline. Vectors representing a box (neatline) around the input vector data will be inserted into the output GRASS vector file.

Parameters:

type*=*name
Coverage type. Either polygon, or line.
Options: polygon, line

lines_in*=*name
ARC/INFO ungenerate lines file; ungenerate format input file containing line or polygon coordinates.

points_in*=*name
ARC/INFO ungenerate label−points file; ungenerate format input file containing label−point coordinates; only applies to 'polygon' type data.

text_in*=*name
ARC/INFO ungenerate label−text file; ungenerate format input file containing category numbers and (optionally) attribute text.

vector_out*=*name
Resultant GRASS vector output file.

idcol*=*value
ID Number column in label−text file. Number of label−text column containing line−ID numbers.

catcol*=*value
GRASS category column in label−text file. Number of label−text column containing category values.

attcol*=*value
GRASS attribute column in label−text file. Number of label−text column containing attribute text.

This program can be run either non−interactively or interactively. The program will run non−interactively if the user specifies the (optional) flag setting and needed parameter values on the command line, using the form:

v.in.arc [**−n**] **type**=*name* **lines_in**=*name* [**points_in**=*name*] [**text_in**=*name*] **vector_out**=*name* [**idcol**=*value*] [**catcol**=*value*] [**attcol**=*value*]

Alternately, the user can type:

v.in.arc

on the command line without program arguments; in this case, the program will prompt the user for the flag setting and parameter values in the manner shown below.

In ARC/INFO, three files are used to store polygon data:

1) a *lines file*, which contains coordinates of all the area edge lines;
2) a *label−point file*, which contains coordinates of label−points (each of which has associated with it a unique label−point ID number). One label−point is associated with each polygon defined in the *lines file*;
3) a *label−text file*, which associates each label−point ID number with a category number and category ("attribute") text.

Linear feature data are stored in two files:
1) a *lines file*, which contains geographic coordinates defining lines, each with a line−ID number; and
2) a *label−text file*, which associates each line−ID number with a category number and attribute text.

These data files are described in further detail below, under the DATA FILE FORMATS section.

INTERACTIVE MODE

The program will prompt the user for the flag setting and parameter values if the user does not specify these on the command line. First, the user will be asked to assign a name to the vector file to store program output:

```
VECTOR (DIGIT) FILENAME
Enter 'list' for a list of existing binary vector files
Hit RETURN to cancel request
>
```

Next, the user is asked to specify the COVERAGE (feature) type to be imported into GRASS. Valid coverage types are **polygon** and **line**.

```
COVERAGE TYPE
Enter "polygon" or "line"
Hit RETURN to cancel request
>
```

IMPORTING A POLYGON COVERAGE

If the user chooses POLYGON coverage, he is asked if he wishes a neatline placed around his data. (The existence of neatlines in the output file can facilitate subsequent patching of data files.)

```
NEATLINE
Do you want a neatline ?
Enter "yes" or "no"
>
```

If the user types **yes**, vectors that box the data will be inserted into the GRASS vector output file (*vector_out*); otherwise, no neatline will be inserted into the output file.

Next, the user is prompted for the name of an existing lines−file containing the geographic coordinates of the arcs forming polygon perimeters. The lines−file is created with the ARC/INFO *Ungenerate LINES* option, and is in the same format at the *prefix.pol* file created by the *v.out.arc* program. The user sees the following prompt:

```
LINES FILENAME
Enter name of the file created with the LINES
option of the ARC/INFO Ungenerate command.
Hit RETURN to cancel request
>
```

The next prompt for coverage type "polygon" asks for the name of an existing label−points file. The label−points file is created with the *Ungenerate POINTS* option, and is in the same format as the *prefix.lab* file created by the *v.out.arc* program. The user sees the following prompt:

```
LABEL−POINTS FILENAME
Enter name of file created with the POINTS
option of the ARC/INFO Ungenerate command.
Hit RETURN if there is no such file
>
```

Finally, the program asks the user for the name of an existing label−text file. This file associates each label−point ID number with a text string. It is in the same format as the *prefix.txt* file created by the *v.out.arc* program.

```
LABEL−TEXT FILENAME
Enter the name of a file that associates
label−point ID numbers with text label strings
Hit RETURN if there is no such file
>
```

v.in.arc then scans the label−text file to find the numbers of lines and columns, the column headers (if any), and the first three lines of actual data in the file. It displays this information to standard output to help the user determine which columns will hold the ID, Category value, and Attribute text data in the new vector output file. A sample of the program's output is shown below:

```
The LABEL−TEXT file has been scanned. There are 132
lines in the file and 8 columns in the file

Column headers of the LABEL−TEXT file:
  rec#\ AREA PERIMETER SOILS#\ SOILS−ID SOIL−CODE DRAIN_CODE TXTUR−CODE

Here are the first three lines :
     1   −2.30228E+07  19399.848     1      0      0      0      0
     2     81079.875    1678.826     2      1     15      3      3
     3    955952.500   10229.637     3      2     19      8      8
```

The column of category values must contain only integer values. The attribute text column can contain a floating point number, an integer, or a word (text string).

Finally, the user is prompted to enter line ID, category value, and attribute text column numbers.

```
Enter the number of the column that should be used
for line IDs (probably the column with −ID) :

Enter the number of the column that is to be used
for GRASS category values:

Enter the number of the column that should be used
for GRASS attribute text:
```

Once these column numbers have been entered, *v.in.arc* will begin converting the ARC/INFO "Generate" format files into GRASS vector file format.

IMPORTING A LINE COVERAGE

The user will also be prompted for input when importing ARC/INFO files containing linear features ("line coverage") data. Like polygon data, linear features are constructed by the series of arcs (aka, vectors) defining their perimeters. If the user selects LINE coverage, the prompts seen by the user will be different in two respects from those for POLYGON coverage. First, the user will not be asked whether or not a neatline is desired; and second, no label−points file name will be requested. In other respects, the treatment of LINE coverage is identical to that for POLYGON coverage.

The user is prompted for the name of the lines−file containing the geographic coordinates of these arcs. The lines−file must first have been created with the ARC/INFO *Ungenerate LINES* option, and is in the same format as the *prefix.lin* file created by the GRASS *v.out.arc* program.

DATA FILE FORMATS

Following are examples of the data files discussed above.

LINES FILE, also known as *prefix.lin* or *prefix.pol* file.

This type of file can be created in ARC/INFO by using the *lines* subcommand of the *Ungenerate* command. Each line (aka, arc) is defined by a line−ID number, followed by a list of at least two easting and northing coordinate pairs, followed by a line with the word "END". The file is terminated with the word "END".

The line−ID number is important only for line coverage data. For a line coverage, the line−ID number is the number that associates each line with its attribute data.

```
3
711916.000000   4651803.000000
711351.875000   4651786.000000
END
3
709562.500000   4651731.000000
709617.250000   4651624.000000
709617.250000   4651567.000000
709585.000000   4651503.000000
709601.125000   4651470.000000
709696.875000   4651503.000000
709720.500000   4651574.000000
709823.750000   4651575.000000
709893.125000   4651741.000000
END
3
710296.875000   4651491.000000
710295.125000   4651470.000000
710223.000000   4651454.000000
710154.500000   4651463.000000
END
END
```

LABEL−POINTS FILE, also known as *prefix.lab* file.

This type of file can be created in ARC/INFO using the *Points* option of the *Ungenerate* command. The first number on each line is a label−point ID number, and the following two numbers are (respectively) the easting and northing coordinate pair representing the geographic location of the label−point.

```
1       711539.875000   4651743.000000
2       711429.000000   4650632.000000
3       711027.625000   4651736.000000
4       711022.625000   4651519.000000
5       710482.750000   4651494.000000
6       710474.500000   4651667.000000
7       709269.750000   4651018.000000
8       709726.500000   4651604.000000
9       708926.375000   4651195.000000
10      708567.500000   4651644.000000
11      708272.750000   4651407.000000
END
```

LABEL−TEXT FILE, also known as *prefix.txt* file.

The ARC/INFO *Display* command can be used to create this type of file.

```
1       -2.30228E+07    19399.848       1       0       0       0
2       81079.875       1678.826        2       1       15      3
3       955952.500      10229.637       3       2       19      8
4       41530.875       926.887         4       3       17      3
5       87900.188       1900.909        5       4       13      3
6       166125.125      3512.950        6       5       15      3
7       29460.563       824.968         7       6       17      3
8       1022769.875     9105.707        8       7       20      9
9       51385.500       1075.638        9       8       17      3
10      376834.875      4470.027        10      9       9       2
11      65802.688       1575.088        11      10      16      3
```

NOTES

ARC/INFO data can be imported even if a label−points and/or a label−text file are missing; however, the lines and/or areas imported will not be labeled.

v.in.arc can handle label−text files both with and without header lines.

Import files are read from any directory, v.in.arc can handle relative or absolute file names with path.

SEE ALSO

v.in.shape, *v.in.dxf*, *v.out.arc*, *v.support*

AUTHOR

Dave Johnson
DBA Systems, Inc.
10560 Arrowhead Drive
Fairfax, Virginia 22030

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

NAME

v.in.arc.poly − Import areas from ARC/INFO ungenerate format.
(GRASS Vector Data Import Program)

SYNOPSIS

v.in.arc.poly *ARC/INFO−file GRASS−vectorfile*

OBJECTIVES:

The two programs and the shell−script are made for importing area data created by ARC/INFO−UNGENERATE to GRASS. The main problems when using v.in.arc followed by v.support directly are errors in the assignment of the area labels to the polygons. This is caused by polygons with identical starting points (first data pair), which are used for assigning the nodes. During this node creation in v.support the node of polygon 1 may be located within the area of polygon2 due to internal approximation processes. In this case, no area labels are assigned. To solve this problem, the whole file has to be scanned and searched for identical starting points. The coordinates of either one of the polygon−pairs with identical starting points have to be permutated. Then, during v.support, different nodes will be created and the assignment of the area labels will be done correctly.

STRUCTURE OF INPUT DATA

We have got our ARC/INFO files in the following format:

```
          1    3457065.250000    6095126.000000
    3457198.000000    6096357.500000
    3457459.750000    6095341.500000
    3457321.500000    6094720.000000
    3457247.500000    6094485.500000
    3457247.500000    6094485.500000
    3456976.250000    6094512.000000
    3456552.250000    6094553.500000
    3456552.250000    6094553.500000
    3456630.250000    6094656.000000
    3457078.250000    6095642.500000
    3457198.000000    6096357.500000
END
.
.
.
END
```

The first line specifies the polygon ID and the label points. The following lines define the x and y coordinates (here in a Gauss Krueger projection) until the keyword END.

Thus, all informations needed for v.in.arc, i.e. the polygons, the label points and the label text, are collected in this file.

IN_ARC

The first program "in_arc.c" extracts these informations from the original file and creates the lines file, the label point file and the label text file. During this process, each polygon gets a unique ID, starting with 1. The correlation to the original ID is stored in the label text file.

PERMUT

Based on this data structure, the program "permut" scans the whole polygon file for identical nodes and removes these nodes by permutation of the data points. This is done in several iteration steps, until all identical nodes are removed. The program uses a temporary file, which will be copied after each iteration step to the original file. Thus after finishing the program, the original file contains the permutated polygons.

v.in.arc.poly

This shell script has been written to import the ARC/INFO data automatically into GRASS, using these two conversion programs. It will be called by:

```
v.in.arc.poly ARC/INFO-file GRASS-vectorfile
```

First, this shell script runs in_arc to extract the lines, label point and label text information. It places the created files (*.ply, *.pnt, *.txt) into the /arc directory of the current mapset. Second, the program permut will be called, replacing the *.ply file created by in_arc. In the steps 3 and 4, the data will be imported using v.in.arc and v.support, resulting in the GRASS−vectorfile specified when calling v.in.arc.poly.

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

NAME

v.in.ascii − Converts ASCII vector map layers into binary vector map layers.
(GRASS Vector Data Import Program)

SYNOPSIS

v.in.ascii
v.in.ascii help
v.in.ascii input=*name* **output**=*name*

DESCRIPTION

v.in.ascii converts a vector map in ASCII format to a vector map in binary format.

OPTIONS

The user can run this program non−interactively by specifying all program options on the command line, in the form:

v.in.ascii input=*name* **output**=*name*

Parameters:

input=*name*
Name of an ASCII vector file to be converted to binary vector file.
output=*name*
Name given to binary vector output file.

If the user runs *v.in.ascii* without giving program arguments on the command line, the program will prompt the user for *input* and *output* file names.

NOTES

After running this program, GRASS support files must be built for the binary *output* file before the user can use the file in *v.digit*. The user can run *v.support* to create GRASS support files for the *output* file.

Note, that the coordinates order is NORTH EAST.

To import category labels, the dig_cats file needs to be specified in `OTHER INFO` field.

Example ASCII vector file (store in dig_ascii/). Note the blank before entering vertex coordinates:

```
ORGANIZATION:
DIGIT DATE:   Oct 24 2001
DIGIT NAME:   nds_gem
MAP NAME:     nds_gem
MAP DATE:
MAP SCALE:    500000
OTHER INFO:   /home/neteler/grassdata5/langeoog/PERMANENT/dig_cats
ZONE:         0
WEST EDGE:    3399965.343
EAST EDGE:    3403360.343
SOUTH EDGE:   5957127.30045067
NORTH EDGE:   5960625.30045067
MAP THRESH:   19.05
VERTI:
A  13
 5958812.48844435 3400828.84221011
 5958957.29887089 3400877.11235229
 5959021.65906046 3400930.7458436
 5959048.47580612 3400973.65263665
 5959069.92920264 3401032.64947709
 5959064.56585351 3401123.82641232
 5958994.84231481 3401188.1866019
 5958914.39207784 3401188.1866019
 5958823.21514261 3401107.73636493
 5958753.49160391 3401005.83273144
 5958753.49160391 3400941.47254186
 5958780.30834956 3400887.83905055
 5958812.48844435 3400828.84221011
A  8
 5959010.9323622 3401338.36037757
 5959096.7459483 3401370.54047235
 5959091.38259917 3401450.99070932
 5958973.38891828 3401520.71424803
 5958882.21198305 3401467.08075671
 5958903.66537958 3401375.90382148
 5958941.2088235 3401332.99702844
 5959010.9323622 3401338.36037757
```

The associated dig_att file needs following format (store in dig_att/):

```
A 3401180.463379 5959003.209139          1
A 3401516.741397 5958966.635071          2
```

The associated dig_cats file needs following format:

```
# 2 categories


0.00 0.00 0.00 0.00
1:forest
2:water
```

To (re−)set map metadata like ORGANIZATION etc. later, *v.digit* can be used.

The GRASS program *v.out.ascii* performs the function of *v.in.ascii* in reverse; i.e., it converts vector files in binary format to ASCII format. These two companion programs are useful both for importing and exporting vector files between GRASS and other software, and for transferring data between machines.

The input from *v.in.ascii* has to be placed into **$LOCATION/dig_ascii** (this directory must be created if not existing).

SEE ALSO

v.digit, *v.import*, *v.out.ascii*, *v.support*, *GRASS ASCII formats*

AUTHORS

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

Last changed: $Date: 2002/03/11 14:04:03 $

NAME

v.in.dgn − Imports Microstation DGN vector files.
(GRASS Vector Data Import/Processing Program)

SYNOPSIS

v.in.dgn
v.in.dgn help
v.in.dgn [**−c**] **dgn**=*name* **output**=*name* [**level**=*value[,value,...]*] [**color**=*value[,value,...*] [**startcat**=*value*] [**type**=*name*]

DESCRIPTION

v.in.dgn creates new GRASS vector map in the current mapset. v.in.dgn supports **line, line string, shape and text** DGN element types. For other types warning is printed on standard error output. Texts are imported as category labels or categories. v.support must be run on vector after import.

COMMAND LINE OPTIONS

Flags:

−c
Use texts as category values (dig_att) instead of category labels (dig_cats).

Parameters:

dgn=*name*
Full path to DGN file.

output=*name*
Name for output vector map.

level=*value*
Comma separated list of levels to be imported. Default value −1 specifies all levels. Values 1−63 allowed.

color=*value*
Comma separated list of colors to be imported. Default value −1 specifies all colors. Values 0−255 allowed.

startcat=*value*

First category wich will be used for texts (if −c flag is not specified).

type=*name*
Type which will be assigned to lines and categories in vector file (**line** or **area** available).

BUGS

Text position for any other than lower−left justification is not exactly the same as in Microstation.

SEE ALSO

v.support

AUTHORS

Radim Blazek

NAME

v.in.dlg2 − Converts an ASCII or binary USGS DLG−3 (bdlg) file to a binary GRASS vector (**dig**) file.
(GRASS Vector Data Import Program)

SYNOPSIS

v.in.dlg2
v.in.dlg2 help
v.in.dlg2 [**−bl**] **input=***name* **output=***name*

DESCRIPTION

This program converts an ASCII or binary USGS DLG−3 (*dlg.old* or *bdlg.old*) file into a binary GRASS vector (**dig**) file.

v.in.dlg2 also creates a **dig_att** file containing the label information 'stripped' from the DLG−3 file. However, the user must run *v.support* (or *v.import* option 4) on the *output* file created by *v.in.dlg2* to create a **dig_plus** file containing the file topology, before using the *output* file in *v.digit*.

The user can avoid this two−step process by converting the ASCII or binary DLG−3 file to binary GRASS vector format using option 1 or 2 of the GRASS program *v.import*.

OPTIONS

Flags:

−b
Input is a binary DLG−3 file (default is ASCII).

−l
Give precedence to line information (default is area).

Parameters:

input=*name*
Name of USGS DLG−3 Optional format input file.

output=*name*
Name to be assigned to the binary GRASS vector files created.

If the user simply types **v.in.dlg2** without specifying parameter values on the command line, the program will prompt the user to enter these.

NOTES

Area vs Line Precedence:

Some machine−processed DLG−3 files do not make the distinction between line edges and area edges. For example, in a roads map, where the desired information is line edge data, a downtown block surrounded by roads may be processed as an area. Because of this, the user is asked to choose whether to give precedence to areas or lines. If precedence is given to lines, the user should be aware that any lines that bound unlabeled areas in the DLG−3 file will be stored as line data. Any unlabeled areas would therefore be lost (this is only a concern when areas are unlabeled; labeled area information will be retained). If precedence is given to areas, lines will be stored as boundaries to areas that are unlabeled.

Building support files with *v.support*:

When you run *v.support* you will have the option of snapping the nodes in your vector file that fall within a certain threshold of one another. WARNING: the default threshold is calculated using the scale of the original DLG−3 file. If the threshold is too high, excessive snapping may occur, destroying the file!! With *v.support*, the user has the option of snapping or not snapping nodes, and further, of setting a particular snapping threshold.

SEE ALSO

v.digit, *v.import*, *v.support*

AUTHOR

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

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

NAME

v.in.dlg − Converts an ASCII USGS DLG−3 Optional file to a binary GRASS vector (**dig**) file.
(GRASS Vector Data Import Program)

SYNOPSIS

v.in.dlg
v.in.dlg help
v.in.dlg [**−l**] **input**=*name* **output**=*name* [**matt**=*name*] [**base**=*val*]

DESCRIPTION

This program converts an ASCII USGS DLG−3 (*dlg*) file into a binary GRASS vector (**dig**) file. Store the DLG−file into $LOCATION/dlg/ (create this directory, if not there) before importing.

Warning: The program reads DLG−3 Optional format only.

v.in.dlg also creates a **dig_att** file containing the label information 'stripped' from the DLG−3 file (the first minor attribute for each record unless **matt** is specified).

If the **matt** is specified, *v.in.dlg* creates an additional attribute file containing identifiers for every record with correspon ding multiple attributes. In this case **matt** file contains identifiers starting with base **base** for the attributes stored in **matt** file (as opposed to the first minor attributes with no **matt** file). The example of **matt** with **base** = 34 would be:

```
34      0       0
35      180     201
36      180     208
        170     240
        190     201
37      160     220
```

With the corresponding *dig_att* looking like this:

```
A 648467.190000 4456367.320000     34
L 667989.290000 4458393.520000     35
L 651002.470000 4473793.390000     36
L 663816.680000 4471412.080000     37
```

However, the user must run *v.support* (or *v.import* option 4) on the *output* file created by *v.in.dlg* to create a **dig_plus** file containing the file topology, before using the *output* file in *v.digit*.

The user can avoid this two−step process by converting the ASCII DLG−3 file to binary GRASS vector format using option 1 of the GRASS program *v.import*.

OPTIONS

Flag:

−l
: Give precedence to line information (default is area).

Parameters:

input*=name*
: Name of USGS DLG−3 Optional format input file.

output*=name*
: Name to be assigned to the binary GRASS vector files created.

matt*=name*
: Name of file with multiple attributes (optional).

base*=val*
: Identifier base for multiple attributes (default is 1).

If the user simply types **v.in.dlg** without specifying parameter values on the command line, the program will prompt the user to enter these.

NOTES

Area vs Line Precedence:

Some machine−processed DLG−3 files do not make the distinction between line edges and area edges. For example, in a roads map, where the desired information is line edge data, a downtown block surrounded by roads may be processed as an area. Because of this, the user is asked to choose whether to give precedence to areas or lines. If precedence is given to lines, the user should be aware that any lines that bound unlabeled areas in the DLG−3 file will be stored as line data. Any unlabeled areas would therefore be lost (this is only a concern when areas are unlabeled; labeled area information will be retained). If precedence is given to areas, lines will be stored as boundaries to areas that are unlabeled.

Building support files with *v.support*:

When you run *v.support* you will have the option of snapping the nodes in your vector file that fall within a certain threshold of one another. WARNING: the default threshold is calculated using the scale of the original DLG−3 file. If the threshold is too high, excessive snapping may occur, destroying the file!! With *v.support*, the user has the option of snapping or not snapping nodes, and further, of setting a particular snapping threshold.

SEE ALSO

v.digit, *v.import*, *v.support*

AUTHOR

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

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

NAME

v.in.dlg.scs − Developed to handle DLG−3 ASCII import of data, specifically a DLG WITHOUT category/attribute codes. DLG files with this affliction will require a flat ASCII file having a 1 to 1 correspondence between DLG area number and a text label.
(SCS GRASS Vector Program)

SYNOPSIS

v.in.dlg.scs
v.in.dlg.scs help
v.in.dlg.scs [**−uf**] [**dlg=***name*] [**bdlg=***name*] [**att=***name*]

DESCRIPTION

Under normal circumstances the *v.in.dlg* program will handle the requirements of reading DLG data and creating vector maps from it. However, *v.in.dlg* assumes that the DLG−s file will contain major/minor category numbers; this is NOT always the case. In some instances, the user may want label names with the DLG data; the current DLG−3 specification does not provide that. SCS has developed this program to meet that need.

Notes:
This program is normally NOT called from the command line. *v.import* will create the command string, then execute it.
This program converts an ASCII DLG file to a binary DLG file with attribute codes in the major minor fields, and creates a *dig_cats* file with the correct code/label correspondence. The program *v.a.b.dlg* must be run after this program to create the GRASS vector files.
It is assumed that the DLG data file contains only one set of geographic information; ie. areas, or lines, or points. This program WILL FAIL if mixed data is encountered.
Degenerate lines are accepted in this program as point data.
DLG.att File format
The DLG.att attribute file format is simple:
field 1 − DLG [area|line|point] number
field 2 − Label

COMMAND LINE OPTIONS

Flags:

−u
DLG file contains universe area.

−f

An attribute file is included.

Parameters:

dlg=*name*
ASCII DLG (*dlg*) file name.
bdlg=*name*
Binary DLG (*bdlg*) file name.
att=*name*
DLG attribute (*dlg.att*) file name.

SEE ALSO

v.digit, v.import, v.in.dlg

AUTHOR

R.L. Glenn, USDA, SCS, NHQ−CGIS P.W.Carlson, USDA, SCS, NHQ−CGIS

Last changed: $Date: 2002/03/19 09:14:37 $

NAME

v.in.dxf3d − Converts the Z values of DXF files to attribute GRASS vector file format.
(GRASS Vector Program)

SYNOPSIS

Bv.in.dxf3d R
Bv.in.dxf3d help R
v.in.dxf3d dxf=*name* **[lines=***name[,name,...]]*

DESCRIPTION

The v.in.dxf3d data conversion program generates GRASS dig_att files from a DXF file with Z values.

This program, in conjunction with v.in.dxf, is ideal for automaticaly importing, to GRASS vector file, layers with Z values (isolines and level contours) from DXF format files.

Firts should be run the v.in.dxf program for import the DXF polylines to binary GRASS vector file format (dig). Later, the DXF Z values of the isolines layers with v.in.dxf3d (dig_att) must be imported, and finally run v.support to attach at this layers their elevations. The v.in.dxf3d.sh script can be used to make all this automaticaly, for a maximun of two isolines layers (normally contours and master contours) from one DXF file.

The v.in.dxf program only recognizes, by now, the Z values from polylines entities in the DXF format.

COMMAND LINE OPTIONS

Parameters

dxf

Name of the DXF input design file from where will be extracted the Z values.

lines

Name(s) of layer(s) in DXF input file containing isoline data with Z values, and the name(s) to be assigned to the GRASS vector data (dig_att) files output.

BUGS

The program only recognizes the Z values if they are in the "30" field of the POLYLINE entity section.

If the input DXF file is in MS−DOS text format, with CR/LF instead of Unix LF, the program doesn't run. To avoid this problem, import to Unix your DXF−MSDOS format files with FTP in ascii mode or with a

command able to convert this files (DOS−COPY, mcopy, etc).

SEE ALSO

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

AUTHOR

The original program dxf3d2gras.bas written in GWBASIC (MS−DOS) by Evaristo Quiroga, Hidrologic and Extern Geodinamic Unity of the University Autonoma of Barcelona (6/93).

The program was rewritten in C for Grass−Unix environment by Evaristo

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.dxf − Converts files in DXF format to ASCII or binary GRASS vector file format.
(GRASS Vector Data Import Program)

SYNOPSIS

v.in.dxf
v.in.dxf help
v.in.dxf [**−a**] **dxf**=*name* [**lines**=*name*[,*name*,...]] \ [**labels**=*name*[,*name*,...]] [**prefix**=*name*]

DESCRIPTION

The *v.in.dxf* data conversion program generates GRASS *dig*, *dig_ascii*, and *dig_att* files from a file in DXF format. Each layer in the DXF input file is converted to a separate *dig* (or *dig_ascii*) layer. For each DXF layer containing text, a *dig_att* file is also created. These output files are placed in the *dig*, *dig_ascii*, and *dig_att* directories under the user's current GRASS mapset.

Output from this program is designed to be used as input to the program *v.cadlabel*.

The *v.in.dxf* program will only recognize points, lines, polylines, and text in the DXF format, and will translate these to GRASS vector format; other types of data are ignored.

Flags:

−a
Output an ASCII GRASS vector (*dig_ascii*) file rather than a binary GRASS vector (*dig*) file.

Parameters:

dxf=*name*
Name of the DXF input design file to be converted to GRASS vector format.

*"**lines**=name[,name,...] or **lines**=in_name:out_name[,in_name:out_name,...]"*
Name(s) of layer(s) in the DXF input file containing line data, and (optionally) the name(s) to be assigned to the GRASS vector data (*dig* or *dig_ascii*) files output.
Default: Convert each layer containing data in the \Idxf file to a GRASS vector data (*dig* or *dig_ascii*) file.

*"**labels**=name[,name,...] or **labels**=in_name:out_name[,in_name:out_name,...]"*
Name(s) of layer(s) in the DXF input file containing text labels, and (optionally) the name(s) to be assigned to the GRASS vector attribute (*dig_att*) files output.
Default: Convert each layer containing text labels in the *dxf* map to a GRASS vector attribute (*dig_att*) file.

***prefix**=name*

Prefix assigned to the *dig* or *dig_ascii* and *dig_att* output file names.
The names of the GRASS vector (*dig*, *dig_ascii*, and *dig_att*) files output are constructed as *prefix.extension*, where *prefix* is the *prefix* name specified by the user and *extension* is the number of the DXF layer from which the data were obtained. If the user does not specify a *prefix* name, the output files take their prefix from the prefix of the input DXF map layer. For example, for the DXF file named *streams.dxf* containing line data on layer , the GRASS vector map layer output would be named *streams.*.

EXAMPLES

lines=15

Outputs line data in DXF layer 15.

lines=15,16

Outputs line data in DXF layers 15 and 16.

lines=ROADS,WATER

Converts line data in DXF layers ROADS and WATER.

lines=15:16

Outputs line data in DXF layer 15, and places it in the *dig* (or *dig_ascii*) file for DXF layer 16.

The below examples are given for a DXF design file named *cont.dxf* containing contour lines and contour line labels, in which:

index contour lines are in DXF layer 9,
intermediate contour lines are in DXF layer 11, and
index labels and some intermediate contour lines are in DXF layer 12.

v.in.dxf can be run with default values, as shown below:

v.in.dxf dxf=cont.dxf

Here, this is equivalent to running the following command:

v.in.dxf dxf=cont.dxf **lines=**9,11,12 **labels=**12

Either of the above commands will produce three GRASS *dig* files (named *cont.9, cont.11,* and *cont.12*) and one *dig_att* file (named *cont.12*).

In our example, however, the *cont.12* file contains intermediate contour lines that the user would like to add to the *dig* file *cont.11*. Our user also wishes to use a different file prefix than the default prefix *cont.* The user therefore types the following command:

v.in.dxf dxf=cont.dxf **lines=**9,11,12:11 **labels=**12 **prefix=**contour

The above command will generate three *dig* files (named *contour.9, contour.11, contour.12*), and will create one *dig_att* file containing text labels (called *contour.12*). No contour lines will appear in the *dig_att* file.

NOTES

Output Filenames:
The output filename, *prefix.extension*, conforms with the GRASS limit of 14 characters. The entire prefix name is used, a '.' inserted, and as much of the extension name is used as the 14 character limit will permit. Excess characters are truncated. To minimize the possibility of creating output files with the same names (resulting in loss of data from the DXF file), use the prefix option to abbreviate the DXF file name. This will leave the majority of characters available for differentiating between layer names.

Translation:
This data translation program does not contain any of the quality control functions available in *v.digit* that will prevent data in an improper format from being input to a GRASS data base. If present, DXF entities are placed in output file(s) corresponding to the layers on which they occurred in the DXF design file input.

Editing:
If the user asks *v.in.dxf* to output ASCII vector (*dig_ascii*) files, they must be converted to binary vector format before they are usable by most GRASS vector commands. The user can convert GRASS vector files from ASCII to binary format by running such programs as *v.support* or *v.in.ascii*. After conversion to binary format the vector files can be displayed (e.g., with *d.vect*); however, the user must run *v.support* on the binary vector files before they can be edited in *v.digit*. The files output by *v.in.dxf* will preserve the data in whatever form they exist in the DXF file. This means that output files may contain unsnapped nodes, overshoots, gaps, and replicated lines. The data, and the file header information (including the owner's name, map's name, date, and scale, and UTM zone) for the GRASS vector files output may require editing by the user in *v.digit*.

Attributes:
The *v.in.dxf* program attaches attributes only to DXF text data that are converted to GRASS vector data (such as contour line labels). Attributes are not attached to converted DXF line data. For each layer of text data in the DXF design file, *v.in.dxf* generates a vector file consisting of rectangular boxes (lines) that are drawn around the DXF text data, and uses the text values to create a GRASS attribute file for the boxes. The vector and attribute files can then be used to label contour lines with the *v.cadlabel* program.

SEE ALSO

v.cadlabel, *v.digit*, *v.in.ascii*, *v.out.dxf*, *v.support*

AUTHORS

Original *dxf2dig* program written by Jan Moorman, U.S. Army Construction Engineering Research Laboratory (6/89)

Revised by Dave Gerdes, U.S. Army Construction Engineering Reseach Laboratory (12/89)

Revised and appended by Jan Moorman, U.S. Army Construction Engineering Research Laboratory (7/90)

Code for arcs and circles from National Park Service GIS Division written by Tom Howard.

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

NAME

v.info − Outputs basic information about a user−specified vector map layer.
(GRASS Vector Program)

SYNOPSIS

v.info
v.info help
v.info input=*name*

DESCRIPTION

v.info reports some basic information about a user−specified vector map layer and the topology status. This map layer must exist in the user's current mapset search path. Information about the map's boundaries, projection, data type, topology presence, category number, number of lines, areas and islands, data base location and mapset, and history are put into a table and written to standard output. The types of information listed can also be found in the cats, cellhd, and hist directories under the mapset in which the named map is stored.

The program will be run non−interactively if the user specifies the name of a vector map layer on the command line, using the form:

v.info input=name
where *input* is the name of a vector map layer on which the user seeks information. The user can save the tabular output to a file by using the UNIX redirection mechanism (>); for example, the user might save a report on the *streams* map layer in a file called *stream.rpt* by typing:

r.info input=*streams* > *stream.rpt*

Alternately, the user can simply type *v.info* on the command line, without program arguments. In this case, the user will be prompted for the name of a vector map layer using the standard GRASS *parser* interface. The user is asked whether he wishes to print the report and/or save it in a file. If saved, the report is stored in a user−named file in the user's home directory.

SEE ALSO

r.info, *v.report*, *v.stats*, *v.support*, *v.what*

AUTHOR

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 $

NAME

v.in.gshhs − import Global Self−consistant Hierarchical High−resolution Shoreline (GSHHS) data
(GRASS Vector Import Program)

SYNOPSIS

v.in.gshhs [−g] [−s] [−a] input=*name* **output**=*name* [**n**=*value*] [**s**=*value*] [**e**=*value*] [**w**=*value*]

DESCRIPTION

The *v.in.gshhs* program imports Global Self−consistant Hierarchical High−resolution Shoreline (GSHHS) data available from http://www.soest.hawaii.edu/wessel/gshhs/gshhs.html. Coordinates are originally lat/long and, if required, reprojected during import to the current GRASS projection. The user may supply bounding coordinates by manually entering north, south, east, and west values, or optionally using the "−g" flag to use the current GRASS region as the bounding coordinates.

Flags:

−g
Obtain north, south, east, and west bounding coordinates from the current GRASS region

−s
Automatically run *v.support* after import

−a
Import vector data as area−line type (default is line)

Parameters:

input
The name of the GSHHS shoreline file
There are five files (resolutions) available: crude, low, intermediate, high, and full

output
The name of the GRASS vector file to be created

n
North limit for imported vector

n
North limit for imported vector

s
South limit for imported vector

e
East limit for imported vector

w

West limit for imported vector

NOTES

Imported vector data is assigned labels (eg.land, lake, etc) derived from GSHHS data. Solid lines that exit and re−enter the import area are broken at the point of exit.

A GSHHS map shift sometimes reported may be related to the map datum. The GSHHS reference ellipsoid is the same as for the GSHHS data sources WVS ("World Vector Shoreline") and WDBII ("World Databank II") for which the reference ellipsoid is unknown. The GSHHS data set is provided in Latitude/Longitude.

EXAMPLE

The following example imports an intermediate resolution shoreline into GRASS using the current GRASS region and automatically runs *v.support* after import:

```
v.in.gshhs −gs input=gshhs_i.b output=shoreline.int
```

SEE ALSO

Appendix K in: The Generic Mapping Tools. Technical Reference and Cookbook.

AUTHORS

This program is based on *gshhstograss*
Author: Simon Cox
Author: Paul Wessel
Author: Bob Covill

Last changed: $Date: 2003/08/20 08:10:13 $

NAME

v.in.mif − Import of MapInfo vector files
(GRASS Vector Data Import Program)

SYNOPSIS

v.in.mif [−l] input=*name* **output**=*name* [**logfile**=*name*] [**scale**=*name*] [**attribute**=*name*]

Flag:

−l
List fields of coverage

Parameters:

input
Name of .MIF/.MID file to be imported
output
Name of target vector file
logfile
Name of file where log operations
scale
Set initial scale [1:200]
Default: 200
attribute
Name of attribute to use as category

NOTE

The MapInfo vector map is read from current directory.

AUTHOR

David D. Gray

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

NAME

v.in.poly − Create a vector file of polygons centered on given locations
(GRASS Vector Program)

SYNOPSIS

v.in.poly
v.in.poly help
v.in.poly [**−t**] [**input**=*name*] **vect**=*name* **radius**=*value* [**segments**=*value*]

DESCRIPTION

This program creates a vector map of polygons of specified radius around center points which may be input as coordinate pairs from a file or from stdin.

COMMAND LINE OPTIONS

Flags

−t
Do not automatically build topology for the new map
Parameters

input=*name*
Name of input file (omit or use − for input from stdin)
default: − (stdin)
vect=*name*
Name of new vector map to create
radius=*value*
Radius of polygon's circumscribed circle
default: none
segments=*value*
Number of straight line segments bounding the polygon
default: set to give perimeter point spacing of 0.02% of the width of the current region based on the radius given. The smallest number of segments that will be automatically used is six(6). Values down to 3 may be explicitly selected on the command line or in input lines (see below). A value of 3 results in an equilateral triangle, 4 a square, etc.

DISCUSSION AND ADDITIONAL INPUT LINE PARAMETERS

If input is from a keyboard, a prompt will be given for each input line. The input lines from the file or stdin should look like this:

```
easting northing [label] [category desc]
            or
easting northing [#label] [category desc]
```

The "#label" for is for center points piped from s.out.ascii, like this:

```
s.out.ascii -d  sitefile | v.in.poly vect=newvect radius=500
```

The polygon (area) label and category description are optional. "label" is an integer and the "category desc" is any text string. If the label value is missing, the label value used is one greater than the last and the description is "n−sided polygon," where n is the number of bounding segments. In this way a series of sequentially numbered polygons may be created by just giving the easting and northing center coordinates.

Two optional command lines may be interspersed with the input lines containing the coordinate and optional label and category descriptions. These lines begin with ".S" or ".R". Note the restrictions on values of radius and segments parameters below.

```
.S 500
```

Changes the number of straight line segments which bound the polygon to a new value. With large values (greater than 20) the polygons will approximate circles. The minimum value of 3 will create an equilateral triangle. Very large values are allowed, but the resulting "circles" may have more definition than is needed and will take lots of storage space for the vector map.

```
.R 100
```

Changes the radius of the circle to a new value. Radius must be greater than 0.0; however, very small values may give meaningless circles and v.support may not be able to construct the topology if the points on the perimeter are too closely spaced.

BUGS

Circles in Lat−Long locations are not really round. Really large Lat−Long circles or polygons may look oddly misshappen when displayed.

SEE ALSO

v.to.rast can be used to convert the polygons to raster maps for masking, etc.

AUTHOR

Dr. James R. Hinthorne, GIS Laboratory, Central Washington University. April 1992.

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

NAME

v.in.scsgef − Converts SCS Geographic Exchange Format (SCS−GEF) ASCII data into a GRASS vector file. *(SCS GRASS Vector Program)*

SYNOPSIS

v.in.scsgef
v.in.scsgef help
v.in.scsgef [**−o**] [**gef=***name*] [**output=***name*] [**cat=***name*]

DESCRIPTION

v.in.scsgef allows a user to create a GRASS vector file from a SCS−GEF format ASCII file.
1. The program will first request the name of the SCS−GEF file to be read in, it expects to find the data in the current directory. It is suggested to create a *gef* directory and put all SCS−GEF data there.

2. The program will then request the name of a GRASS vector file.

3. The program will then request the name of a SUBJECT file. A subject file will be used to assign GRASS category codes to the SCS−GEF data. It is structured the same as a *dig_cats* category file. It is suggested that a SUBJ directory be created in the GRASS location and a file containing all SCS−GEF text labels by category be created. This will be required to provide consistancy across several maps (quads) within one location. The user may use the *vi* text editor or the SCS macro *make_subject* to create it.

4. The program will then read the SCSGEF header information, interactively present information that was available, and request additional data of the user. These questions are :
Name of their organization. (from SCS−GEF)
Digitized Date. (from SCS−GEF)
Map Name. (from SCS−GEF)
Map Location. (from SCS−GEF)
Other Information. (from SCS−GEF)
State FIPS code.
County FIPS code.
Present GEF Coord. System (table, stplane, ll, utm).
Coordinate System Desired (utm, stplane, ll, albers).

The program will then actively read the SCS−GEF data file and process it,

scripts contains SCS macro *make_1_gef*. This macro makes one file out of the three (3) files found in SCS−GEF (see SCS−GEF technical specifications for more information). The macro must be run on each data set BEFORE *v.in.scs.gef*.

COMMAND LINE OPTIONS

Flags:

−o

The SCS−GEF is in the OLD format (24 char).

Parameters:

gef*=name*

ASCII SCS−GEF file name.

output*=name*

Vector file name.

cat*=name*

Category file name.

SEE ALSO

make_1_gef, *make_subject*, v.import

AUTHOR

R.L. Glenn, USDA, SCS, NHQ−CGIS

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

NAME

v.in.sdts − Imports SDTS vector data, conforming to the Topological Vector Profile, into GRASS, creating GRASS vector map(s) and associated attribute files ready to be installed in a relational database.
(GRASS Vector Data Import/Processing Program)

SYNOPSIS

v.in.sdts
v.in.sdts help
v.in.sdts [**−il**] **catd**=*name* [**output**=*name*] [**dbpath**=*name*] [**domain**=*name*] [**map**=*name*] [**theme**=*name*] [**manifold**=*name*]

DESCRIPTION

v.in.sdts creates one or more GRASS vector maps in the current mapset from a Spatial Data Transfer Standard dataset conforming to the Topological Vector Profile (TVP). The program generates GRASS *dig*, *dig_att*, and *dig_cats* files. Also, if requested, files of attributes in database−ready form are created, along with scripts to create an approprate SQL−compliant relational database, and load the attribute files into the new database. Special database−ready files of tables linking the attributes to the GRASS vector map layer or layers are also generated.

The source SDTS dataset must be in the user's current directory. The files that make up the dataset are listed in the dataset's Catalog/Directory file (CATD); this file is specified by the user with the **catd** parameter.

v.in.sdts creates maps in your current mapset, and will only import map data if there is correspondence between the current mapset's coordinate system and that of the transfer set; in addition, for UTM (and State Plane), Zone designations must match. These specifications can be displayed be running *v.in.sdts* in "Information only" mode. "Information only" mode is automatically put in effect when there is a mis−match between source and target coordinate systems.

An SDTS dataset may consist of one or several distinct map layers (or "2−D manifolds", in SDTS terminology), coinciding with one or more partitions of the earth's surface. If a dataset contains more than one map layer, the grouping of object data into individual map layers, and of groups of map layers, is specified in the Catalog/Spatial Domain (CATS) file, in terms of "domain", "map", "theme", and/or "manifold" ("aggregate object"). If available, this information is displayed to the user in "Information Only" mode. The user can then either
(1) import all the map layers in a transfer at once, or
(2) select a subset of the transfer consisting of one or more maps by specifying a domain name, map name, etc.

COMMAND LINE OPTIONS

Flags:

−i

"Information−only" mode. Information about the dataset and any individual map layers in the dataset are displayed. No map layers or attribute files are generated. Information displayed includes basic identifying data (TITLE of transfer dataset, map date, dataset creation date, scale, coordinate system, etc.). For individual maps, any names found in the CATS file specifying map, theme, domain, manifold, are given. Bounding coordinates for each map layer are also printed.
The program will also run in "information only" mode if (1) no output name is specified, or (2) the coordinate system, or, in the case of UTM and State Plane, Zone, of the dataset to be imported does not match the current mapset.

−l

Import object link table(s) only; do not create attribute tables. If this flag is set, and if **dbpath** is set, only the vector map (*dig*, *dig_att*, and *dig_cats*) and the file containing the database−ready table linking the vector map with the attribute tables will be created; the attribute files themselves will not be created. This option is useful if the user wants to selectively import data layers from an SDTS dataset with multiple maps. One map could be imported with its object link table and the full set of attributes; subsequent layers imported with the "**−l**" option would avoid recopying the full set of attributes.

Parameters:

catd*=*name

Full name of SDTS file containing the Catalog/Directory (CATD) module for the source dataset. The file name format is specified by SDTS and the TVP as *xxxxCATD.DDF*, where *xxxx* are 4 digits or upper−case letters or any combination therof. The CATD file must be located along with the rest of the SDTS dataset in the current directory. The CATD file contains a listing of all the dataset files, and is thus the necessary starting point for the transfer process.
Note that the same four−character prefix of the CATD file is used for all files in the SDTS dataset. This prefix is also used by *v.in.sdts* for the naming of the output attribute files (see *The GRASS−SDTS User Guide* for details.)

output*=*name

name for output vector map layer. If the SDTS dataset contains multiple maps, and if no particular one is specified, causing all the maps to be imported, maps will be distinguished by name plus numeric suffix. If not specified the module runs in "information mode" (−i flag) and no output is written.

dbpath*=*name

full path to location for placement of database−ready attribute files preparatory to their installation in a relational database. Path must exist and be writable by the user. Setting the **dbath** parameter causes database−ready files to be created; otherwise they are not created.

domain*=*name
map*=*name
theme*=*name
manifold*=*name

if one or more domain, map, theme, or manifold ("aggregate object") names are given in the SDTS dataset Catalog/Spatial Domain (CATS) file, map layers so designated can be selected with the appropriate parameter. "Information only" mode lists any such names found in the CATS file.

SPATIAL OBJECTS IN SDTS AND GRASS

SDTS and the Topological Vector Profile define two basic types of spatial objects: simple spatial objects, i.e., lines, polygons, nodes, etc.; and composite objects, which are made up of one or more other simple and/or composite spatial objects. SDTS composite objects, which GRASS cannot handle directly, are imported as records in RDBMS−ready tables. Details on the mapping of simple and composite spatial objects between SDTS and GRASS may be found in the *GRASS−SDTS User Guide*.

SDTS ATTRIBUTE IMPORT

Only a brief explanation of SDTS attributes and v.in.sdts's handling of them is given here. See the *GRASS/SDTS User Guide* for details.

SDTS is capable of substantial attribute complexity. SDTS distinguishes between two basic kinds of attributes: primary attributes are related directly to spatial objects (simple or composite), while secondary attributes are related to primary or to other secondary attributes. In SDTS, attributes are stored in relational tables, and spatial objects may be linked to multiple attributes in one or more different attribute tables. The schema of an SDTS dataset−−the number and kind of attribute fields and attribute tables, and the relationships among attributes and objects−−is not predefined by the Standard or the Profile, but is determined by the producer of the dataset.

For most kinds of data likely to be available through SDTS, optimal access requires use of GRASS with a relational database management system. In support of this, *v.in.sdts* imports SDTS attribute tables in a form ready for use with your RDBMS. It also produces SQL−compatible scripts to set up the relational database and install the data.

dig_att and dig_cats files: The program does generate *dig_att* and *dig_cats* files, and for relatively simple SDTS datasets, i.e., those with one−to−one object−attribute relationships with all object attributes in a single attribute table, an associated relational database is not necessary. In addition, for more complex datasets, the *dig_att* and *dig_cats* files are constructed in such a way as to facilitate post−import incorporation of selected data from the attribute files for use without recourse to a relational database. Specifically, the contents of the generated *dig_att* and *dig_cats* files are as follows:

dig_att

Contains an entry for each attributed non−composite object (line, polygon, point). each entry will be assigned a unique category integer value. These integers, or feature−IDs (FID), also uniquely identify the same spatial objects, in the relational database object link table. *dig_cats* For datasets with one−to−one object−attribute relationships and a single object−related attribute module, only one database−ready "object−attribute" file is created, and the *dig_cats* records are given the same content, as the generated database−ready file. Record structure is as follows:

```
FID | obj_code | attr_code | attr. field 1 |...| attr. field n |
```

(*obj_code* and *attr_code* are codes, derived from record IDs in the SDTS dataset, which function as keys in the import relational database. See *The GRASS−SDTS User Guide* for details.)

For other datasets, the *dig_cats* file is identical in content to the generated GRASS−database object link table, and records would have the structure:

```
FID | obj_code     or    FID | obj_code | attr_code
```

SDTS IMPORT AND USE OF A RELATIONAL DATABASE

Full discussion of this topic may be found in the *GRASS−SDTS User Guide*.

FILE NAMES

vector map name: if the SDTS dataset contains a single map layer, or if a a single map layer from a multiple−map dataset, the name specified in output is used as is. Otherwise, the name is extended with integers to specify the individual layers. *relational database file names:* see the GRASS−SDTS User Guide.

SEE ALSO

m.sdts.read, *v.in.sdts*, *v.out.sdts*, *v.sdts.dp.cp*, *v.sdts.meta.cp*, *v.sdts.meta*

AUTHORS

David Stigberg, U.S.Army Construction Engineering Research Laboratory
Tin Qian, University of Illinois

Last changed: $Date: 2002/06/17 17:20:39 $

NAME

v.in.shape − Read an ArcView Shapefile
(GRASS Vector Program)

SYNOPSIS

v.in.shape

v.in.shape[−loud] input=*name* [**output=***name*] [**verbose=***debug level*] [**logfile=***name*] [**snapdist=***snap distance*] [**scale=***orig. scale*] [**attribute=***category number*] [**label=***category label*]

DESCRIPTION

The *v.in.shape* program is designed to import ArcView Shapefiles. *v.in.shape* will be run non−interactively if the user specifies program arguments on the command line. Alternately, the user can simply type:

v.in.shape

on the command line without program arguments. In this case, the user will be prompted for parameter values using the standard GRASS user interface described in the manual entry for *parser*.

FEATURES

By default Grass files created have the name extracted from the basename of the shapefile. This can be over−ridden by supplying a value to the *output* option, which must be a legal name.

The current version is more robust than the module available up to Spring 2002. The emphasis is now to concentrate on maintaining an effective shape file import filter. To assist this, features that are really extensions of the basic aim have been removed. Selection of features with particular attributes can still be extracted to a new file with *v.extract*. There is also less need for posterior error correction now, as the current module filters out some potentially problematic shapes at the point of import, and so, will always insert incoming data into the final map. If a map fails to build because of topological errors, the problem can be tracked down using *v.digit* and fixed. Formerly, the module attempted many kinds of fixes, however there were many complaints about imported maps being modified or corrupted by the process, so we now leave the fixing of such errors to the user.
The snap distance, which always has a small default value if not supplied, creates a microgrid of cells in the import map region. Only one vertex is ever assigned to a particular cell. If a later vertex is added that is in the same cell, it is considered co−incident with the first. When a link is added between two vertices, it is recorded and the same link is not added twice. The snap distance is also used to define the co−incidence of other spatial entities, for example edges of bounding boxes. This snap distance has no relation to the snap distance defined

for the dig_plus map. (This still needs some improvement).

OPTIONS

Flags:

−l

Force an area (polygon theme) import to be imported as lines **as is**. This option can be used to create an outline of the original boundaries even if the map is bad. This option is ignored for other types of import.

−o

Allow over−write of existing vector map.

−u

If the record number is used as the attribute, several 'parts' of the same 'shape' will have the same value when these are split on import to GRASS which does not currently support compound objects. This may be what you want, but it might not. To assign a unique and sequential record number, as attribute, to each area choose this option. So, this option is ignored if the *attribute* is set.
Note: This can cause a subtle difference even if applied to coverages without compound objects, as it always applies a sequential ID value to each successively imported object. Without the option, it is the record number in the original that is used, which can skip values if there are **NULL** shapes, because GRASS ignores the latter. This may be important if you plan to use the attribute as a row ID in an external database or *dbmi*.

−d

List fields in DBF file

Parameters:

input*=filename*

Name of input shape file. Provide a full path name or the name of a file in the current directory. Any of the full pathname, basename, or prefix only will suffice.

output*=filename*

Name of vector map to be created (default: prefix of shape file). By default prefix of shape file name is used.

verbose*=integer*

Number between 0 (minimal report of fatal errors and essential warnings) and 3 (very verbose log − reports most data transfer operations and some intermediate steps). The default is **1**.

logfile*=filename*

Name of file[path] where log info will be written. By default, log info is directed to *stderr*, which is also used (with a warning) if the log filepath is invalid or not writable.

snapdist*=fp−number*

A grid resolution can be defined within which adjacent vertices will snap.
Note: The vertices do not snap to the grid. All the vertices inside a grid cell snap to one of their number (usually the first). This value defaults to 1.0e−10 ground units. In fact, it will never allow a value that is less. This is still not quite right, but errors caused by it are astronomically unlikely, and are easily fixed manually, which must be set off against the import process increasing in duration by a factor of several if the snapping procedure was perfectly correct. Really effective snapping must await the development of spatial query functions. Therefore it will not be changed at this stage.

scale*=integer*

This sets an original scale that will be specified in the header of the vector map file produced. It can be edited later with **v.digit**. The value defaults to 1:2400. The value affects some behaviour of *v.digit*

including proper building, so may have to be played with. Of course it can be reset by *v.digit* following import.

attribute*=name*

Name of the input field to use as the category number in dig_att. Defaults to using the record ID number as a category value if no value is assigned or a non−numeric field is given. If the field is floating−point the value is rounded to the nearest integer.

label*=name*

Name of the input field to use as the category label in dig_cats. Only writes out results if a meaningful category field is given, otherwise no action is taken. If the same attribute is re−assigned a new category, the value is over−written.

BUGS AND CAVEAT

There is no support for projection since the projection information is not stored in SHAPE files.

Multipatch data is not yet supported. Point data is now imported, but in the GRASS 5.0.x vector model, these vector 'points' have limited utility.The module *s.in.shape* imports site data in a form that may be found more useful.

The filtration process creates a large temp file to store the imported data and a significant amount of metadata about the imported points. This can be large in large files. It is deleted if the import proceeds in a normal fashion, but will be left 'dangling' on abnormal termination. This however is a quite general problem in the current GRASS environment model. If disk space is available, this module will now import an unlimited size of map, though it is currently slow on large maps, as there are many disk read/write operations. This is compounded by the lengthy build process, which however affects all vector map builds.

Area and perimeter fields in input data may no longer be quite correct if the lines have been adjusted to correct topology problems. But if the user has not modified the file, the values should be correct. If in doubt, GRASS's own modules may be used to generate dimensional data, for example *v.report*

EXAMPLE

Example:

```
v.in.shape in=map.shp out=map attribute=OBJART label=ALIASFOLIE
scale=25000
```

This command imports the shape file "map" into GRASS at scale of 1:25000 (since the scale is not stored in the shape file). "attribute" is the field name containing the record ID (if not set, internal numbering will be done). "label" is the field containing the associated text labels. Use the "−d" flag to get the list of fields from the "map.dbf" file.

SEE ALSO

m.in.e00, g.mapsets, g.region, v.digit, v.proj, s.in.shape, v.support, v.to.rast, v.out.shape v.extract v.report

AUTHORS

Frank Warmerdam (warmerda@home.com)
Based on Shapelib (http://gdal.velocet.ca/projects/shapelib/).

Markus Neteler
added category support

David Gray
preprocessing to provide correct handling of polygon edges, labels and correction of some topological errors. Also some new options q.v.
Spring 2002: Rewrite of most of the code.

Last changed: $Date: 2002/06/17 17:22:46 $

NAME

v.in.tig.basic − Create GRASS vector map from TIGER files
(GRASS Vector Program)

SYNOPSIS

v.in.tig.basic v.in.tig.basic help
v.in.tig.basic [−pqt] t1*=TIGER.1* ***t2****=TIGER.2* ***out****=name [****zone****=utm_zone]*
[**spheroid**=*spheroid*] [**tlid**=*file*]

DESCRIPTION

This program creates a GRASS vector map in the current mapset (UTM or Lat−Long locations only) with labelled line segments constructed from the end points (nodes) from the Type 1 TIGER file records and shape points from the Type 2 TIGER file. The nodes and shape points are matched by the TIGER record number (TLID field). In the attribute (label) file which is built (in the *dig_att* directory), the lines are given the "area boundary" type and line labels are the record numbers (9 digits!).

The Type 1 file may contain the records for a complete county, a subset for a county, or those from more than one county concatenated. The Type 2 file must contain all those records corresponding to the Type 1 records, but may contain extra records which will not be used. Typically, a subset of Type 1 records is used and a full county Type 2 file is used.

The program should function well independent of the termination character(s) at the end of each line in the input files. <LF>, <CR>, neither, one, or both are acceptable. Different distribution media apparently have different record delimiting characters.

The −p flag should not normally be used. It causes the program to build a disk file pointer table for all of the Type 2 records each time the program runs, rather than using the table built previously. Detection, naming and verification of the pointer file are automatic. It is created in the "tmp" space in the user's location.

OPTIONS

Flags:

−p
Create new disk file for Type 2 pointers every time

−q
Perform functions quietly

−t
Build topology (dig_plus) file when done (can't be quiet)

Parameters

t1*=name*
TIGER Type 1 path/file name
t2*=name*
TIGER Type 2 path/file name
out*=name*
Name of vector map to create
zone*=value*
UTM zone number; default is location zone
options: 1−60
default: (current zone)
spheroid*=keyword*
Spheroid for LL to UTM conversion; see m.gc.ll
default: clark66
tlid*=name*
Path/file for list of TLID numbers to process from TIGER.1 File
default: nonE

THE tlid PARAMETER

The file specified by the *tlid* parameter may be of any size. Any number at the beginning of a line will be interpreted to be the TLID of a Type 1 record which is to be included in the output map. Other information on these lines is ignored, as are lines which do not begin with a number. If fewer than 5000 numbers are in this file, it is only read once and execution speed is greatly enhanced. Normally, a full Type 1 file will be specified by the *t1* parameter when a *tlid* file is used.

BUGS/RESTRICTIONS

Caution: This program will overwrite an existing vector map of the same name without warning.

The Type 2 input file must be on a device capable of "seeking," i.e., not a tape drive. CD−ROM is OK (but program will run much faster from disk files).

Input lines in a *tlid* file are limited to 300 characters in length. If the system can allocate space to store all the TLID numbers read from this file, it is only read once and execution speed is greatly enhanced; on a typical system, space for more than 10,000 numbers can be allocated.

SEE ALSO

v.apply.census will generate polygon labels for Census tracts, block groups, etc., from STF1 or PL94−171 files.
m.in.stf1.tape and/or unix text tools can be used to extract/concatenate useful subsets of TIGER Type 1 records prior to running **v.in.tig.basic**.

AUTHOR

Dr. James Hinthorne, GIS Laboratory, Central Washington University

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

NAME

v.in.tiger.scs − Converts ASCII TIGER data files from the U.S. Dept. of Commerce Bureau of the Census. *(SCS GRASS Vector Program)*

SYNOPSIS

v.in.tiger.scs
v.in.tiger.scs help
v.in.tiger.scs [**−cv**] **tig1**=*name* **tig2**=*name* **out**=*name* **cfc**=*name*[,*name*,...]

DESCRIPTION

This program imports Census line features from TIGER records type1 and type2 into GRASS vector format. Both pre−Census and post−Census data formats can be used. Specific Census Feature Class Codes (CFCC) can be extracted completely or in various combinations. These codes are described in the TIGER/line Census Files 1990 documemtation available from the Bureau of the Census. An additional feature code consisting of the three letters "BOU" may also be specified to extract a county boundary. Condensed Record 1 files may be imported with the −c flag. These files should be identified with a trailing "x" character on the filename.

COMMAND LINE OPTIONS

Flags:

−c
Condensed TIGER file.

−v
Verbose output.

Parameters:

tig1=*name*
TIGER file 1.

tig2=*name*
TIGER file 2.

out=*name*
New vector file name.

*"**cfc**=name,name,..."*
Specific Census Feature Class (CFCC) codes.

EXAMPLE

To extract all Primary (A1) and Secondary (A2) roads from a county's TIGER files the following command would be used:
v.in.tiger.scs tig1=t12113.1 **tig2**=t12113.2 **out**=roads **cfc**=A1,A2
To extract all the Hydrographic features in a county's TIGER files with verbose output:
v.in.tiger.scs −v tig1=t12113.1 **tig2**=t12113.2 **out**=hydro **cfc**=H
To extract the county boundary the command would be:
v.in.tiger.scs tig1=t12113.1 **tig2**=t12113.2 **out**=bou **cfc**=BOU

NOTES

The TIGER files must in sorted order before being used. This can be done by using the following command:

```
sort TGR12113.F21 −o t12113.1
sort TGR12113.F22 −o t12113.2
```

For consistency the sorted file should be written as above. It should consist of a 't' followed by the State and County FIPS code, then a '.' and then a value to identify the record number.

The CFCC code 'BOU' used to extract the County Boundary should be used alone as it will result in a polygon AREA being created.

Currently output is in UTM only.

SEE ALSO

v.import

AUTHOR

Paul H. Fukuhara, USDA SCS National Cartographic Center

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

NAME

v.in.tig.lndmk − Create GRASS vector map from TIGER files
(GRASS Vector Program)

SYNOPSIS

v.in.tig.lndmk v.in.tig.lndmk help v.in.tig.lndmk t1=TIGER.1

v.in.tig.lndmk [−is] t1=*TIGER_1* [**t2**=*TIGER_2*] [**t7**=*TIGER_7*] [**t8**=*TIGER_8*]
[**tI**=*TIGER_I*] [**tP**=*TIGER_P*] [**input**=*file*] [**vect**=*name*] [**site**=*name*]
[**zone**=*value*] [**spheroid**=*name*]

DESCRIPTION

This program creates a GRASS site or vector map of Census "Landmark" features in the current mapset (UTM or Lat−Long locations only) with labelled points, areas or lines constructed from the TIGER file records. Each point or area Landmark is a record in the Type 7 TIGER/Line file for a county.

The user may select the Landmark Features to include in the new map by specifying CFCC's (Census Feature Classification Code), or by specifying text string(s) to match with the Landmark Feature Name field. [Many Landmark records do not have an entry in the Feature Name field, so use the string matching selection with care.]

Point Landmarks are generated only from information in the Type 7 file. Area Landmarks, which often comprise several individually labelled polygons, are generated from data in TIGER file Types 7, 8, I, P, 1 and 2 (in order of use).

When making "line" maps, this program functions as a selection front end to the *v.in.tig.basic* program; see that programs documentation for the information on how these line maps will be labelled.

TIGER/Line files from multiple counties should not be mixed or concatenated when using this program.

The program should function well independent of the termination character(s) at the end of each line in the input files. <LF>, <CR>, neither, one, or both are acceptable. Different distribution media apparently have different record delimiting characters.

OPTIONS

PARAMETERS

t1=*name*

Path/name of Type 1 Tiger File

t2*=name*
Path/name of Type 2 Tiger File

t7*=name*
Path/name of Type 7 Tiger File

t8*=name*
Path/name of Type 8 Tiger File

tI*=name*
Path/name of Type I Tiger File

tP*=name*
Path/name of Type P Tiger File

input*=name*
file path/name for input commands

vect*=name*
Name of vector map to create

site*=name*
Name of site map to create

zone*=value*
UTM zone number; default is location zone
options: 1−60
default: 12

spheroid*=keyword*
Spheroid for LL to UTM conversion; see m.gc.ll
default: clark66

Flags:

−i

Force interactive mode

−s

Input commands from stdin
Notes: **−i** flag implied if neither flag set and no **input** file specified. Omitted Tiger file names will be derived from the TIGER.1 name given by suitably changing the last character.

THE COMMAND MODE OF OPERATION

Starting *v.in.tig.lndmk* with the *−s* flag invokes the command mode of operation. Commands are expected from *stdin*, that is, the a terminal or a file redirected with '|' or '<'. When using a terminal for input, the prompt lists the available commands thus:

.p(oints) .a(reas) .l(ines) .c(ode)list
.s(trings) .m(atch)both .e(nd) .ex(it) >>

"Syntax for the dot commands"

.points [site_map]
lines containing requested CFCC Codes
(CFCC>> prompt will be given if input is from a terminal) **.end**
.areas [vect_map]
lines containing requested CFCC Codes

CFCC>> prompt will be given if input is from a terminal) **.end**
.lines [vect_map]
lines containing requested CFCC Codes
(CFCC>> prompt will be given if input is from a terminal) **.end**
.strings [output_file]
1 to 10 lines of 30 chars to match with Landmark names;
use '!' as first character to require exact case match;
(*String #n:* prompt will be given if input is from a terminal) **.end .matchboth [no]**
.codes [P|A|L[=output_file]]

.exit

"The details of the dot commands"

Note: Always use the **.strings**, **.matchboth** and/or **.codes** commands prior to creating a map with **.points**, **.areas** or **.lines**.

.points mapname begins the process of making a GRASS4.1 site map of point locations. This site map name takes precedence over the site=name parameter that may be given on the command line.

Specification of the CFCC's, if any, for the features you want included in the map should be done on the line(s) following **.points** and should conclude with a line containing **.end**, which begins the map generation process. The CFCC's should be of the form Dnn (*e.g.*, D00; D10 D85 | D23, H12); several may be given on one line, separated by any common separator character. Requested CFCC's may be abbreviated to D1 (meaning D10 through D19) or just D (meaning D00 through D99).

If matching text strings are desired (see *.strings\'fR), they must be specified prior to the* **.points** command.

.areas mapname begins the process of making a GRASS4.1 vector map of polygons (areas). This vector map name takes precedence over the vect=name parameter that may be given on the command line. Specification of the CFCC's, if any, for the features you want included in the map should be done on the line(s) following **.areas** and should conclude with a line containing **.end**, which begins the map generation process.

If matching text strings are desired (see *.strings\'fR), they must be specified prior to the* **.areas** command.

.lines mapname begins the process of making a GRASS4.1 vector map of lines. This option uses only the Type 1 and 2 TIGER files. It searches the Type 1 records for the specified CFCC's and matches any specified text strings to the Feature Name (FENAME) field in the Type 1 records. Specification of the CFCC's, if any, for the features you want included in the map should be done on the line(s) following **.areas** and should conclude with a line containing **.end**, which begins the map generation process.

If matching text strings are desired (see *.strings\'fR), they must be specified prior to the* **.lines** command.

.codes searches the Type 7 Landmark file and sends to stdout, or to a file if one specified, the CFCC and the Classification description of each CFCC present. If a 'P' or 'A' is specified, both the point and area codes present are listed. 'L' lists the CFCC codes contained in the Type 1 file. The default type, if no type is specified, is 'P'. This command is provided as a useful aid to the user.

.strings records up to 10 following lines (until ".end") as text strings to match to the Landmark Feature Name Field in the Type 7 records (Type 1 for making ".line" maps). A match is made if any of the specfied strings is a *substring* of the Feature Name. If exact case matching is required, '!' should be the first character of the

entered text strings. Example: The Feature Name "Walter Reed Army Hospital" would be matched by any of the following strings.

```
!Walt
hosp
Reed
army hosp
```

If the optional filename parameter is given, Type 7 records which match the specified strings are reported to that file, otherwise they are reported to stdout.

.matchboth is used to specify that a Landmark record (or Type 1 record for ".lines") must match both one of the specified CFCC's *and* one of the specified text strings to be selected. A match to *either* condition is the default mode, and can be explicitly set or reset by the optional "no" parameter; *i.e.*, **.matchboth no**.

.exit is used to exit from the program.

THE INTERACTIVE MODE OF OPERATION

The following "VASK" menus will be encountered in using the interactive mode of this program. Comments following each menu will help you decide how to answer the questions and create products.

GRASS IMPORT FROM CENSUS LANDMARK FEATURE RECORDS

STEP 1: Select Type(s) of features to be extracted

```
There are Point and Area (polygon) Landmark feature types.
  in the Landmark (Type 7) Records

There may also be some Line (vector) Landmark features
  in the Basic Data (Type 1) Records

Mark one feature type you wish extracted:

               Point   x
               Area    _
               Line    _

View CFCC Codes for selected type  x

or  Exit program  _

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

Comments: This is the "Main Menu." You must select (by entering 'x') **one** of the Point, Area or Line choices. The viewing of the CFCC codes is optional, but is usually done initially.

The following screen results from a request to View CFCC Codes after selecting Point features:

```
  TIGER Type 7 file <tgr53037.f47>
  examined.  15 unique CFCC codes found in 123 records.

  Mark 'x' to see list on screen  _
```

```
     Enter file name (home directory) to save list to file  _______________

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

Comments: You may request to see the list on the screen, or have the list sent to a file in your home directory, or both. If you request neither, the program returns to the Main Menu. Both point and area landmarks are reported if either "Point" or "Area" was selected on the Main Menu. If "Line" was requested on the Main Menu, all the CFCC codes in the Type 1 (Basic Data Record) file will be tabulated (this can be a useful product for other purposes); remember that the Type 1 file is large and this processing may take a few minutes or more.

The following is typical output of CFCC Codes:

```
TIGER Type 7 file <tgr53037.f47>
Number of Each Area Landmark CFCC
     6 D00 Landmark Feature, Class Unknown or Not Elsewhere Classified
     3 D10 Military installation
     1 D28 Campground
     1 D31 Hospital
     5 D43 Educational institution
     1 D81 Golf course
     7 D82 Cemetery
    14 D85 State or local park or forest
     7 H00 Water Feature, Class Unknown or Not Elsewhere Classified
     8 H11 Perennial stream
     9 H30 Lake or pond
    32 H31 Perennial lake or pond
     1 H32 Intermittent lake or pond
Total Area Records: 95

Number of Each Point Landmark CFCC
     1 D00 Landmark Feature, Class Unknown or Not Elsewhere Classified
     5 D28 Campground
     1 D31 Hospital
    10 D43 Educational institution
     4 D44 Religious institution
     5 D71 Lookout tower
     1 D81 Golf course
     1 D82 Cemetery
Total Point Records: 28
```

After reviewing the list of available CFCC Codes, select "Point, Area or Line" again on the Main Menu and something like the following two menus will appear. On the first one, select any number of CFCC Codes (categories) that you desire to be included in a new map.

```
        LANDMARK FEATURE CATEGORY SELECTION MENU: POINT LANDMARKS
        D00 _ Landmark Feature, Class Unknown or Not Elsewhere Classified

        D28 x Campground
        D31 _ Hospital
        D43 _ Educational institution
        D44 _ Religious institution
        D71 _ Lookout tower
        D81 _ Golf course
        D82 _ Cemetery

        Done: _  Start over: _
```

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

Comments: In this case "Point" was selected on the Main Menu, and the user has indicated that a GRASS site map of Campgrounds is to be made.

Now you have the opportunity to enter character strings which will be matched against the Feature Name field (in the Type 7 record in the case of "Point" or "Area" maps, and in the Type 1 record for "Line" maps).

```
  Step 1C  Enter text(s) to match in finding Landmark Features.
           (Use ! as first char to require exact case match.)

                       _____________________________________
                       _____________________________________
                       _____________________________________
                       _____________________________________
                       _____________________________________
                       _____________________________________
                       _____________________________________
                       _____________________________________
                       _____________________________________
                       _____________________________________

             Check for any Landmark Records that match your
             text strings before proceeding to next step?  _

             File to save matches to (omit for screen display):
            ____________________________________________________

          AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
                 (OR <Ctrl-C> TO ERASE AND START OVER)
```

Comments: See *.strings* discussion in the command line section of this manual page for examples of what can be entered on the lines. If no entries are made then no string matching to the Feature Name field will be done.

BUGS/RESTRICTIONS

Some area features are included in the Type 7 (Landmark) file which do not fit the definition given of "Landmark" (CFCC's beginning with 'D'), as a mechanism to have them included in the Tiger/Line data. Many Hydrography area features fall in this category for some counties. Also, there may be records in both the Type 1 and Type 7 files for some Landmark features, such as airports, and to make a complete airport map for a county both sets must be extracted.

In the first trials of this program is was discovered that the TIGER/Line files have several defects in topological completeness when area landmark polygons are created. These defects express themselves when v.support is run (automatically) to attach labels to the polygons and is unable to attach one or more label points. Some editing with v.digit should be able to resolve these problems.

Caution: This program will overwrite an existing vector map of the same name without warning. Site_list maps will not be overwritten.

Since this program opens many TIGER files at once, the set for a county must reside on a disk or CD−ROM, not tape. Only file types 1, 2, 7, 8, I and P are used so only those would have to be copied from tape to disk.

SEE ALSO

v.in.tig.basic is called automatically to produce the area and line maps.
v.apply.census will generate polygon labels for Census tracts, block groups, etc., from STF1 or PL94−171 files.

AUTHOR

Dr. James Hinthorne, GIS Laboratory, Central Washington University, 1992.

APPENDIX

"List of Census CFCC Landmark 'D' Codes"

```
[Extracted from file CODESTIG.ASC on TIGER distribution CD−ROM.]
     CFCC    CLASSIFICATION D = LANDMARK FEATURES
     D00     Landmark Feature, Classification Unknown or Not Elsewhere
              Classified
     D10     Military installation
     D20     Multihousehold and transient quarters
     D21     Apartment building or complex
     D22     Rooming or boarding house
     D23     Trailer court or mobile home park
     D24     Marina
     D25     Crew of vessel
     D26     Housing facility for workers
     D27     Hotel, motel, resort, spa, YMCA, or YWCA
     D28     Campground
     D29     Shelter or mission
     D30     Custodial facility
     D31     Hospital
     D32     Halfway house
     D33     Nursing home, retirement home, or home for the aged
     D34     County home or poor farm
     D35     Orphanage
     D36     Jail or detention center
     D37     Federal penitentiary, state prison, or prison farm
     D40     Educational or religious institution
     D41     Sorority or fraternity
     D42     Convent or monastery
     D43     Educational institution
     D44     Religious institution
     D50     Transportation terminal
     D51     Airport or airfield
     D52     Train station
     D53     Bus terminal
     D54     Marine terminal
     D55     Seaplane anchorage
     D60     Employment center
     D61     Shopping center or major retail center
     D62     Industrial building or industrial park
     D63     Office building or office park
     D64     Amusement center
     D65     Government center
     D66     Other employment center
     D70     Tower
     D71     Lookout tower
```

```
D80     Open space
D81     Golf course
D82     Cemetery
D83     National park or forest
D84     Other federal land
D85     State or local park or forest
D90     Special purpose landmark
D91     Post office box ZIP code
```

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

NAME

v.in.tig.rim − Imports Census Bureau line data (TIGER files) to GRASS vector format.
(GRASS Vector Data Import Program)

SYNOPSIS

v.in.tig.rim
v.in.tig.rim help
v.in.tig.rim dbname=*name* **in1file**=*name* **in2file**=*name* **zone**=*value*

DESCRIPTION

v.in.tig.rim imports Census line data (called TIGER) and creates a "master" binary vector file containing a large amount of data. Various map layers can then be created by querying information from the master vector file using *v.db.rim* or one of the *Gen.* shell scripts listed in the SEE ALSO section, below. The database name (*dbname*) given on the command line will be the name of the rim data base, and the master vector file in GRASS will be named "dbname.Master". The master vector file will include all information from the type1 and type2 TIGER files given on the command line as *in1file* and *in2file*. If the user simply types *v.in.tig.rim* on the command line, all parameters will be queried using the standard GRASS parser described in the manual entry for *parser* .

OPTIONS

Parameters:

dbname=*name*
Vector/rim data name (with a maximum of 7 characters).
in1file=*name*
TIGER type1 input file name.
in2file=*name*
TIGER type2 input file name.
zone=*value*
Universal Transverse Mercator (UTM) zone in which these data are located.
Options: −60 − 60

NOTES

TIGER data are presented in latitude/longitude format, and are converted to UTM (Universal Transverse Mercator) coordinates as part of this importing routine. The spheroid used in the conversion is clark 66, as it is the most consistent with the original data.

This command must be compiled separately, and requires the use of *rim* and *v.db.rim* which contain FORTRAN code. The user must have access to a FORTRAN compiler in order to compile and use this command, since it calls both *rim* and *v.db.rim*.

If the user does not know the UTM zone for this data input file, the command *m.tiger.region* should be run first to determine the zone.

v.support must be run separately on the output file if needed.

FILES

Source code for RIM is located under $GISBASE/../src.related/rim

Source code for *v.db.rim* is located under $GISBASE/../src.garden/grass.rim/v.db.rim

Source code for *v.in.tig.rim* is located under $GISBASE/../src.garden/grass.tig.rim/v.in.tig.rim

Source code for *m.tiger.region* is located under $GISBASE/../src.alpha/misc/m.tiger.region

SEE ALSO

tig.rim.sh, *m.tiger.region*, *v.db.rim*, *tiger.info.sh*.

AUTHOR

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

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

NAME

v.in.transects − import transect data to a GRASS vector map
(GRASS Vector Program)

SYNOPSIS

v.in.transects
v.in.transects help
v.in.transects *[−**a**] **input**=name [**fs**=*"character"] **length**=value*
[**units**=*name*] [**decl**=*name*] **vect**=*name* [**TITLE**=*"*name*"]

DESCRIPTION

v.in.transects imports "transects" into a GRASS vector map. A transect is a line (or an area) which is described by a starting coordinate, a direction or azimuth, (or backward, forward, rightside, and leftside distances from the line transect), and a length (in meters or feet). The information describing the transects must be in a file prepared by the user before running **v.in.transects.** The format of this file is described below in the section **"TRANSECT FILE FORMAT".**

OPTIONS

Flags:

−a

Run for area transects. Default is line transects, i.e., this flag is not used.

Parameters:

input

transect information file
This is the file containing the transect information to be imported. The format of this file is described below in the section **"TRANSECT FILE FORMAT"**

fs

input data field separator
The transect file is organized one transect per line with at least 4 fields. The *fs* option specifies the character that is used in the transect information file to separate the fields. If not specified, fields are assumed to be separated by blanks (white space). (Quote the delimeter whenever you are not sure whether it has special meaning in UNIX or not, to avoid the misunderstanding by GRASS.)

length

transect length
This is the length of the transects (default is in grid units, e.g., meters). It is assumed that all transects have the same length. If they do not, then more than one import process must be run to create two or more vector files and the results patched together (using v.patch).

units

units of the length
default: meter
This is the unit of the transect. It can be meter or foot.

decl

declination − angle (in degrees) to be added to input azimuth angles
default: 0
Each transect has a direction or azimuth angle associated with it. The map projection may have a declination associated with it and if the azimuth angles embedded in the transect input file do not account for this declination, it may be specified here.

vect

Vector map to be created

TITLE

Title for resultant vector map
default: Transect map
If TITLE is more than one word, it should always be quoted.

TRANSECT FILE FORMAT

The format for the transect file consists of one record or line per transect with 4 mandatory fields and a 5th optional field for line transects. The first field is the GRASS category number to be assigned to the transect. The second and third fields are the easting and northing (respectively) of the starting coordinate for the transect. The fourth field is the azimuth in degrees clockwise from north of the transect. Following the fourth field is an optional fifth field that is the category label for the transect. The following is a simple example with 3 transects:

```
1 709818   5453991   246.0   P CLGR 4 73 1 21 0 0 0 0 1
2 698350   5464162   128.0   P CLGR 0 55 0 36 0 0 0 0 1
3 704615   5461874   190.5   P DEGR 0 34 4 15 0 0 0 0 0
```

Note that the fifth field (i.e., the label) is really everything after the azimuth, not just the P.

This file could be formatted as follows:

```
1:709818:5453991:246.0:P:CLGR:4:73:1:21:0:0:0:0:1
2:698350:5464162:128.0:P:CLGR:0:55:0:36:0:0:0:0:1
3:704615:5461874:190.5:P:DEGR:0:34:4:15:0:0:0:0:0
```

In this case the fields are separated by a colon so *fs=:* must be specified on the command line. The labels (starting with the P) would retain the colons (i.e., they are not removed from the label even though they act to define the first 4 fields).

When area transects are required, the transects file should include four (4) more fields for backward, forward, rightside, and leftside distance from the corresponding line transect. The format will be as the following:

```
1   709818   5453991   246.0   10.0   15.0   20.0    5.0   P CLGR 4 73 1 21 0 0 0 0 1
2   698350   5464162   128.0    5.0   15.0   10.0   25.0   P CLGR 0 55 0 36 0 0 0 0 1
3   704615   5461874   190.5   15.0   20.0   10.0    5.0   P DEGR 0 34 4 15 0 0 0 0 0
```

Note that delimiter could be other than white space in transect file. Vector files of line transects can be generated from the above area transects file by not using *−a* flag. However, the label will include four (4) more items.

NOTES

The resulting vector map is a complete GRASS vector map: it will have a category file with the labels from the input file and it will have the topology file already built. (v.support is run automatically by **v.in.transects** as the final step in creating the GRASS vector map.)

AUTHORS

Tao Wen, University of Illinois at Urbana−Champaign, Illinois.
Michael Shapiro, U.S.Army Construction Engineering Research Laboratory

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

NAME

v.label − makes a label−file from a GRASS vector.
(GRASS Vector Program)

SYNOPSIS

v.label
v.label help
v.label map=name [type=type] text=name label=name [xoffset=value] [yoffset=value] [reference=name] [font=name] [size=value] [color=name] [width=value] [hcolor=name] [hwidth=value] [background=name] [border=name] [opaque=name] [space=value]

DESCRIPTION

v.label makes a label−file from a GRASS vector file. Labels are read from text file which contains category and label on each row separated by space. Example of text file:

```
1 Kamenice
2 Cerna Desna
3 Bila Desna
```

In current version only lines are labeled and label is created along line.

PARAMETERS

map
: Name of a vector

text
: Full path to text file containing labels

label
: Name of a paint−label file

xoffset
: Offset label in x−direction
default: 0

yoffset
: Offset label in y−direction
default: 0

reference
: Reference position
options: center,left,right,upper,lower
default: center

font
Font
default: standard

size
Label size (in map−units)
options: 1−1000
default: 100

color
Text color
options: aqua,black,blue,brown,cyan,gray,green,grey,indigo, magenta,
orange,purple,red,violet,white,yellow
default: black

width
Line width of text (only for p.map output)
options: 1−100
default: 1

hcolor
Highlight color for text (only for p.map output)
options: none,aqua,black,blue,brown,cyan,gray,green,grey, indigo,magenta,
orange,purple,red,violet,white,yellow
default: none

hwidth
Line width of highlight color (only for p.map output)
options: 0−100
default: 0

background
Background color
options: none,aqua,black,blue,brown,cyan,gray,green,grey, indigo,magenta,
orange,purple,red,violet,white,yellow
default: none

border
Border color
options: none,aqua,black,blue,brown,cyan,gray,green,grey, indigo,magenta,
orange,purple,red,violet,white,yellow
default: none

opaque
Opaque to vector (only relevant if background color is selected)
options: yes,no
default: yes

space
Space between letters (in map−units)
options: 1−100000
default: 100

NOTE

Rotated labels produced by this module are supported by ps.map only.

AUTHOR

Philip Verhagen (original s.label)
Radim Blazek

NAME

v.llabel − Bulk−labels unlabeled points or lines in a binary GRASS vector file.
(GRASS Vector Program)

SYNOPSIS

v.llabel
v.llabel help
v.llabel [**−in**] **map**=*name* [**type**=*name*] [**value**=*value*] [**label**=*value*]

DESCRIPTION

v.llabel allows the user to bulk−label currently unlabeled points or lines (not areas) in a binary GRASS vector file (i.e., a *dig* file). The user must run *v.support* on the vector file before running *v.llabel* if any modifications have been made to the file since the last time *v.support* was run on it, to ensure that all lines are properly identified in the file topology.

The program also runs *v.support* on the vector file after labelling so that the changes will be made evident.

v.support builds GRASS support files for (binary) GRASS vector data files. These support files supply topology (*dig_plus*) and category (*dig_att*, *dig_cats*) information that are needed by other GRASS programs (e.g., by *v.digit*, *v.to.rast*, etc.).

OPTIONS

Program parameters and flags have the following meanings.

Flags:

−i
: Label incrementally. For each unique, unlabeled line or point in the vector file, increment the category value by one, starting from the initial default or user−assigned *value*.

−n
: Do not run *v.support*. There may occasionally be instances where the user prefers not to run *v.support* immediately.

Parameters:

map=*name*
: Name of binary GRASS vector data file whose unlabeled lines are to be labelled. This map layer must exist in the user's current GRASS mapset.

type=*name*
Select type of primitives to be labeled. Primitives include points, line arcs, area edge arcs and points. v.llabel does not label areas.
Options: point, edge, line
Default: all

value=*value*
The category value to be assigned to all unlabeled points or lines in the vector map layer. If the −*i* flag is set, *v.llabel* will increment the initial *value* by one for each unique unlabeled point or line in the vector map.
Default: 1

label=*value*
The category label to assign to all unlabeled points or lines in the vector map layer. If unspecified, all added categories will be given an empty label the the *dig_cats* file.

The user can run this program non−interactively by specifying parameter values (and optionally, the flag setting) on the command line.

Alternately, the user can simply type the command **v.llabel** on the command line. In this event, the program will prompt the user to enter parameter values and flag settings.

NOTES

A *dig_plus* file must be created for each imported vector map before it can be used in *v.digit*.

Topological information for GRASS vector files can also be built using *option 4* of the *v.import*. program.

The user can bulk−label unlabeled line features in a binary vector file using *v.digit*.

SEE ALSO

v.digit, *v.in.ascii*, *v.support*, and *v.alabel*

AUTHORS

James Darrell McCauley, Agricultural Engineering, Purdue University
Dave Gerdes, U.S.Army Construction Engineering Research Laboratory

Last changed: $Date: 2002/04/13 14:38:29 $

NAME

v.make.subj − Create a "subject" file from all category labels found in a set of listed vector maps.
(SCS GRASS Vector Program)

SYNOPSIS

v.make.subj
v.make.subj help
v.make.subj map=*name*[,*name*,...] **subj**=*name*

DESCRIPTION

Program to read the *dig_cats* file for each listed map. The program then creates (or appends to, if the SUBJ file exists) a "subject" file of all of the labels, giving a unique category value to each.

This program was designed to create a common category file from maps of areas that have the same category labels, but different category values; and that must be joined/merged together.

COMMAND LINE OPTIONS

Parameters:

map=*name*
Vector map−−source for composite.
subj=*name*
New SUBJ file name.

SEE ALSO

v.merge

AUTHOR

R.L. Glenn , USDA, SCS, NHQ−CGIS

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

NAME

v.merge − Merges vector map files.
(SCS GRASS Vector Program)

SYNOPSIS

v.merge
v.merge help
v.merge [**−mv**] **map=***name*[,*name*,...] **output=***name* **subj=***name*

DESCRIPTION

Program to read the *dig_cats* file for each named map. The program then compares the category label (NOT the value) to the labels in the named subject file. If the label is found in the subject file, the corresponding attribute values of the map are changed to the subject file category value. When all maps are completed a patch operation is performed.
This program was designed to merge maps for areas that have the same category labels, but different category values associated with those labels.
There are two flags for *v.merge*:

Flags:

−m
: Use RAM to hold subject file category values. This may be faster than the default (use of disc files); however, memory may not be large enough for all of the subject file.

−v
: By default, only the name of the mapset processing is sent to standard output. In verbose mode, the user is given additional information on program operations.

Parameters:

*"**map=**name,name,..."*
: Vector map−−source for composite.

output=*name*
: New name assigned to vector composite map.

subj=*name*
: SUBJ file name.

NOTES

The program *v.make.subj* can be run prior to using *v.merge*. *v.make.subj* will read the category labels of each map in its list and create a subject file of labels and values. Users may opt to create the subject file by other means if they wish.

v.rmedge must be run on any mapsets in the *v.merge* list prior to issuing the *v.merge* command. Common edges will need to be removed prior to the *v.merge* operation.

SEE ALSO

v.make.subj
v.rmedge

AUTHOR

R.L. Glenn, USDA, SCS, NHQ−CGIS

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

NAME

v.mkgrid − Creates a (binary) GRASS vector map of a user−defined grid.
(GRASS Vector Program)

SYNOPSIS

v.mkgrid
v.mkgrid help
v.mkgrid map=*name* **grid**=*rows,columns* **coordinate**=*x,y* **box**=*length,width* [**angle**=*degree*]

DESCRIPTION

v.mkgrid will create a binary format, vector map representation of a regular coordinate grid.

Flags:

−q
Quiet. Cut out the chatter.

Parameters:

map=*name*
Name to be assigned to binary vector map layer output.
grid=*rows,columns*
Number of ROWS and COLUMNS to appear in grid.
coordinate=*x,y*
Lower left EASTING and NORTHING coordinates of vector map layer output.
box=*length,width*
LENGTH and WIDTH of boxes in grid.
*[**angle**=degree]*
optional rotate grid counter−clockwise about the origin (**coordinate**).

If the user simply types **v.mkgrid** on the command line without specifying parameter values, the program will prompt the user for inputs using the standard interface described in the manual entry for *parser*.

NOTES

The new binary vector map output is placed under the *dig* directory in the user's current mapset, and can be used like any vector map layer. Run *v.support* to build the topology information for the vector map before using *v.mkgrid* map layer outputs in the *v.digit* program.

Since the grid is computer−generated, the corner coordinates will be exact and can be used when patching together several *v.mkgrid* grids.

This is NOT to be used to generate a vector map of USGS quadrangles, because USGS quads are not exact rectangles. To generate a vector map of USGS quads, use the program *v.mkquads*.

The program ignores the current GRASS geographic region settings. It will create the complete grid even if part of this grid falls outside the current geographic region.

Rotating the grid should not usually be necessary under normal usage, but this option is available.

SEE ALSO

v.digit, v.mkquads, v.patch, v.support and parser

AUTHOR

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

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

NAME

v.mkgrid − Creates a (binary) GRASS vector map of a user−defined grid.
(GRASS Vector Program)

SYNOPSIS

v.mkgrid
v.mkgrid help
v.mkgrid map=*name* **grid**=*rows,columns* **coordinate**=*x,y* **box**=*length,width* [**angle**=*degree*]
[**type**=*const*|*rows*|*cols*] [**value**=*constant value*]

DESCRIPTION

v.mkgrid will create a binary format, vector map representation of a regular coordinate grid.

Flags:

−q
Quiet. Cut out the chatter.

Parameters:

map=*name*
Name to be assigned to binary vector map layer output.
grid=*rows,columns*
Number of ROWS and COLUMNS to appear in grid.
coordinate=*x,y*
Lower left EASTING and NORTHING coordinates of vector map layer output.
box=*length,width*
LENGTH and WIDTH of boxes in grid.
*[**angle**=degree]*
optional rotate grid counter−clockwise about the origin (**coordinate**).
*[**type**=const|rows|cols]*
Option to add attributes to the grid. Area points are in the centre of the respective cells. *const* creates attributes of the same value in each cell (the value is **1** by default, but may be chosen with the **value** option. The other options number the cells in sequence. *rows* numbers the rows in turn, *cols* numbers the columns in turn. If the grid is rotated, numbering is relative to the pre−rotated position.
*[**value**=constant value]*
This option determines the constant value of the cell attributes if the **type**=*const* option is chosen, otherwise it is ignored if supplied.

If the user simply types **v.mkgrid** on the command line without specifying parameter values, the program will prompt the user for inputs using the standard interface described in the manual entry for *parser*.

NOTES

The new binary vector map output is placed under the *dig* directory in the user's current mapset, and can be used like any vector map layer. Run *v.support* to build the topology information for the vector map before using *v.mkgrid* map layer outputs in the *v.digit* program.

It may be advisable to use a small snap distance with *v.support* to make sure the grid lines snap correctly along the edges.

Since the grid is computer−generated, the corner coordinates will be exact and can be used when patching together several *v.mkgrid* grids.

This is NOT to be used to generate a vector map of USGS quadrangles, because USGS quads are not exact rectangles. To generate a vector map of USGS quads, use the program *v.mkquads*.

The program ignores the current GRASS geographic region settings. It will create the complete grid even if part of this grid falls outside the current geographic region.

Rotating the grid should not usually be necessary under normal usage, but this option is available. This can also be used with attribute options.

SEE ALSO

v.digit, v.mkquads, v.patch, v.support and parser

AUTHOR

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

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

NAME

v.mkquads − Creates a GRASS vector map layer and/or sites list and/or geographic region definition file for a USGS 7.5−minute quadrangle.
(GRASS Vector Program)

SYNOPSIS

v.mkquads
v.mkquads help
v.mkquads [**−esrvx**] **map=***name* [**type=***quad size*]

DESCRIPTION

There are three types of output available from the GRASS program *v.mkquads*:

1. a vector map of all the full USGS quadrangles that will fit within the boundaries of the current geographic region.
2. a GRASS sites list containing the corner coordinates of each of these quads.
3. GRASS geographic region definition files associated with each of the quads created.

A *quad* is defined as the area covered by a USGS 7.5−minute (1:24,000) map. This program is useful for managing a GRASS data base LOCATION which contains a number of quads which are to be patched together.

Flags:

−e

Encompass current geographic region with quads (rather than only creating those quads that lie inside of the geographic region). Use of this option will affect all output options.

−s

Create a GRASS sites list file. The sites list will contain all the corner coordinates of all the full quads that can be built in the current geographic region. The sites list file can then be displayed using the *d.sites* program.

−r

Create region file(s): quad.1 quad.2 ... The program will generate a separate geographic region definition file for each quad; each of the geographic region files created will have the prefix *quad.* with some number attached to it. For example, if only one quad were created, the geographic region file *quad.1* would also be created in the *windows* directory under the user's current mapset. To make the program−generated geographic region definition file *quad.1* your *current* geographic region setting, run the GRASS *g.region* program.

−v

Create vector file (default). Only full quads will be created. The binary vector map layer output is placed under the user's *dig* directory and can be used like any other vector map layer. Run *v.support* to build the topology information for the vector map before using *v.mkquads* map layer outputs in the *v.digit* program. Since the quads are computer−generated, the corner coordinates will be exact. This simplifies digitizing if one or more quad sheets will have to be brought together for a data base, because all of the quad corner points to be joined will be guaranteed to match.

−x

Create a GRASS registration (*reg*) file.

Parameter:

map*=name*

The name of a file to contain program output.

type*=quad size*

options: full,third,quarter,1x1,1x2
default: full

If the user runs *v.mkquads* without including program parameter value and desired flags on the command line, the program will prompt the user for the above information using the standard GRASS interface described in the manual entry for *parser*.

NOTES

All output options can be used on the command line at the same time. A listing of all the quad points in latitude/longitude and projected coordinates will be displayed each time the program is executed.

BUGS

There are no guarantees that *v.mkquads* will function properly if a quadrangle crosses UTM zones. This program has not been tested outside the northwest UTM quadrant.

SEE ALSO

d.sites, *g.region*, *v.digit*, *v.mkgrid*, *v.support* and *parser*

AUTHORS

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

Last changed: $Date: 2003/08/20 08:10:13 $

NAME

v.out.arc − Exports GRASS vector files to ARC/INFO's "ungenerate" file format.
(GRASS Vector Data Export Program)

SYNOPSIS

v.out.arc
v.out.arc help
v.out.arc type=*name* **vect**=*name* **arc_prefix**=*name*

DESCRIPTION

v.out.arc is a GRASS data export program that converts files in GRASS vector format to ARC/INFO's "Generate" file format. The companion program *v.in.arc* imports data in ARC/INFO's "Generate" format and converts them to GRASS vector format.

This program can be run either non−interactively or interactively. The program will be run non−interactively if the user specifies parameter values on the command line using the following format:

v.out.arc type=*name* **vect**=*name* **arc_prefix**=*name*

Alternately, the user can simply type:

v.out.arc

on the command line; in this case, the program will prompt the user for parameter values.

Parameters:

type=*name*
Coverage (feature) type.
Options: polygon, line

vect=*name*
The name of a GRASS vector file to be converted to ARC/INFO format.

arc_prefix=*name*
A prefix to be assigned to the ARC/INFO−format files output by *v.out.arc*.

INTERACTIVE MODE: USER PROMPTS

v.out.arc will prompt the user to enter the name of a GRASS vector file to be exported to ARC/INFO and for a filename prefix to be used in naming the files created by the program.

A GRASS vector file to be exported to ARC/INFO must either contain only linear features (i.e., have only line coverage) or contain only area edge features (i.e., have only polygon coverage). *v.out.arc* will begin by asking the user which type of coverage (line or polygon) is to be imported:

```
COVERAGE TYPE
Enter "polygon" or "line"
Hit RETURN to cancel request
>
```

The program then prompts the user for the name of the GRASS vector file to be converted to ARC/INFO format:

```
VECTOR (DIGIT) FILENAME
Enter 'list' for a list of existing binary vector files
Hit RETURN to cancel request
>
```

Next, the user is asked for a file−name prefix to be used in naming the output ARC/INFO Generate format files:

```
ARC/INFO (GENERATE) FILENAME PREFIX
Hit RETURN to cancel request
>
```

The filename prefix will be used to name the various files that will be created for export to ARC/INFO. When labeled polygon coverage data are exported, three such files will be created: a *lines file* with the suffix .lin, a *label−points file* with the suffix .lab, and a *label−text file* with the suffix .txt. When line coverage data are exported, two such files will be created: a *lines file* with the suffix .lin, and a *label−text file* with the suffix .txt. Export of unlabeled polygon or line coverage data will result in creation of a *lines file* (.lin suffix) only. All exported files are stored in the current directory.
See the DATA FILE FORMATS section for more information on these files.

EXAMPLE

Linear features and polygon data are made up of the series of arcs (aka, vectors) outlining their perimeters. The *ARC/INFO Users' Guide*, in its discussion of the *Ungenerate* command, explains how line and polygon coverage data can be created from files (like *prefix.lin* and *prefix.pol*) containing the geographic coordinates of these arcs, and from files (like *prefix.lab*) containing the geographic coordinates of label−points. Below is an example which illustrates the creation, within ARC/INFO, of a polygon coverage data file (named *soils*) from the files *soils.pol* and *soils.lab*.

```
Arc: GENERATE SOILS
Generate: INPUT soils.pol
Generate: LINES
Generating lines ...
Generate: INPUT soils.lab
Generate: POINTS
Generating points ...
Generate: QUIT
Arc: _
```

The above example would create a polygon coverage data file named *soils* with label−points. The label−points would have ID numbers that correspond to the GRASS category values for the polygons in the coverage. The INFO portion of ARC/INFO can be used to associate these label−point ID numbers with

descriptive text from the *soils.txt* file.

DATA FILE FORMATS

LINES FILE, also know as *prefix.lin* or *prefix.pol* file:
This text file is a "Generate" format lines file. The *lines* option of the ARC/INFO *Generate* command can be used to read this file into ARC/INFO. Each line in the file has a unique line−ID number.

```
101
223343.62 218923.15
223343.62 222271.06
259565.31 222271.06
259565.31 195577.37
END
102
237862.53 203392.37
244970.75 203744.28
253137.66 195577.37
259565.31 195577.37
END
103
237862.53 203392.37
237862.53 203744.28
223343.62 218392.37
END
104
239072.44 186200.56
237862.53 187410.50
237862.53 203392.37
END
END
```

LABEL−POINTS FILE, also known as *prefix.lab* file:
This text file will be created by *v.out.arc* if the vector file being exported represents a polygon coverage. *prefix.lab* consists of a list of label−point (x,y) coordinates, each with a unique label−point ID number.

```
1 242777.81 211533.09
2 243458.37 199282.28
3 243458.37 195199.28
```

LABEL−TEXT FILE, also known as *prefix.txt* file:
In the case of polygon coverage data, this file associates an integer category value and a category label ("attribute") text string (containing no spaces) with each label−point ID number. In the case of line coverage data, this file associates an integer category value and an attribute text string with each line−ID number.

The first column is the row number (which is arbitrary), the second column contains the category value, the third column holds the line or label−point ID number, and the fourth column contains the attribute text string.

```
1 4 1 Coniferous
2 5 2 Deciduous
3 2 3 Rangeland
```

SEE ALSO

v.in.arc
v.support

AUTHOR

Dave Johnson
DBA Systems, Inc.
10560 Arrowhead Drive
Fairfax, Virginia 22030

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

NAME

v.out.ascii − Converts a binary GRASS vector map layer into an ASCII GRASS vector map layer.
(GRASS Vector Data Export Program)

SYNOPSIS

v.out.ascii
v.out.ascii help
v.out.ascii input=*name* **output**=*name*

DESCRIPTION

v.out.ascii converts a GRASS vector map in binary format to a GRASS vector map in ASCII format.

The program can be run non−interactively if the user specifies all needed program arguments on the command line, in the form

v.out.ascii input=*name* **output**=*name*

Parameters:

input=*name*
Name of the binary GRASS vector input file to be converted to ASCII format.
output=*name*
Name of the ASCII GRASS vector output file.

If the user runs *v.out.ascii* without giving the names of an input and output file on the command line, the program will prompt the user for these names.

NOTES

The GRASS program *v.in.ascii* performs the function of *v.out.ascii* in reverse; i.e., it converts vector files in ASCII format to their binary format. These two companion programs are useful both for importing and exporting vector files between GRASS and other software, and for transferring data between machines.

The output from *v.out.ascii* will be placed in the user's current mapset under the *$LOCATION/dig_ascii* directory.

v.out.ascii does not copy the *dig_cats* file associated with the binary vector *input* map to the new *output* file name. The user must copy the *dig_cats* file to the new *output* name if this is desired (e.g., using the UNIX *cp* command).

SEE ALSO

v.digit, v.import, v.in.ascii, v.support

AUTHORS

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

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

NAME

v.out.dgn − GRASS vector format to Microstation DGN format conversion program.
(GRASS Vector Data Export Program)

SYNOPSIS

v.out.dgn
v.out.dgn help
v.out.dgn [**−a**] input=*name* **dgn=***name* [**type=***name[name,...]*] [**catas=***name[name,...]*] [**entity=***name*] [**level=***value*] [**color=***value*] [**cat=***value*]

DESCRIPTION

The GRASS *v.out.dgn* conversion program generates an Micostation DGN file. Lines with more than 101 points are written as complex chains. Areas are written as filled shapes and areas with islands as complex shapes with holes.

Categories may be written as texts, levels or mslinks. Mslink is Microstation link to external databe.

COMMAND LINE OPTIONS

Flags:

−a
Append to existing DGN file.

Parameters:

input=*name*
The name of an existing GRASS vector.

dgn=*name*
Full path to DGN output file.
NOTE: DGN files output by Microstation have the suffix **.dgn**

type=*name*
The type of grass objects which should be exported:
point,line,area,boundary or centroid.
Area is exported as shape and boundary as line.

catas=*name*

Specifies how categories should be exported. Options are: **nothing,mslink,text,level,color**.

entity=*name*
Entity number used for mslinks if catas=mslink is used.

level=*value*
The level the elements will be written on.

color=*value*
Color number used for elements.

cat=*value*
Output only GRASS objects of specified category.

SEE ALSO

v.in.dgn

AUTHOR

Radim Blazek

NAME

v.out.dlg − Converts binary GRASS vector data to DLG−3 Optional vector data format.
(GRASS Vector Data Export Program)

SYNOPSIS

v.out.dlg
v.out.dlg help
v.out.dlg input=*name* **output=***name*

DESCRIPTION

The GRASS program *v.out.dlg* allows the user to convert GRASS vector data to DLG−3 Optional format, for export to other systems. The user can run the program non−interactively by specifying all program arguments on the command line, in the form:

v.out.dlg input=*name* **output=***name*

Parameters:

input=*name*
Name of the binary GRASS vector data file to be converted to DLG−3 format.
output=*name*
Name to be assigned to the DLG−3 Optional format output file created.

If the user does not give the names of an input and output file on the command line, the program will prompt the user to enter these names.

NOTES

The *v.out.dlg* program requires that the *input* vector map layer have full topological information associated with it. This means that the GRASS program *v.support* should have been the last program to have effected any changes upon the vector map layer before it is run through *v.out.dlg*. If this is not the case, *v.out.dlg* will terminate with a message that *v.support* needs to be run. The output from *v.out.dlg* will be placed in *$LOCATION/dlg*.

SEE ALSO

v.import, *v.in.ascii*, *v.in.dlg*, *v.support*

AUTHOR

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

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

NAME

v.out.dlg.scs − Converts binary GRASS vector data to binary DLG−3 Optional vector data format.
(SCS GRASS Vector Data Export Program)

SYNOPSIS

v.out.dlg.scs
v.out.dlg.scs help
v.out.dlg.scs input=*name* **output=***name*

DESCRIPTION

The GRASS program *v.out.dlg.scs* allows the user to convert GRASS vector data to DLG−3 Optional format, for export to other systems, just as *v.out.dlg* does. However, a flat ASCII file of labels and their coresponding dlg record numbers is created. This flat file is used to provide the label information to other systems, since most do NOT support text attributes in a DLG import.

The user can run the program non−interactively by specifying all program arguments on the command line, in the form:

v.out.dlg.scs input=*name* **output=***name*

Parameters:

input=*name*
Name of the binary GRASS vector data file to be converted to DLG−3 format.
output=*name*
Name to be assigned to the DLG−3 Optional format output file created.

If the user does not give the names of an input and output file on the command line, the program will prompt the user to enter these names.

NOTES

The *v.out.dlg.scs* program requires that the *input* vector map layer have full topological information associated with it. This means that the GRASS program *v.support* should have been the last program to have effected any changes upon the vector map layer before it is run through *v.out.dlg.scs*. If this is not the case, *v.out.dlg.scs* will terminate with a message that *v.support* needs to be run.

The output from *v.out.dlg.scs* will be placed in *$LOCATION/dlg*. The output flat file from *v.out.dlg.scs* will also be placed in *$LOCATION/dlg*, with the extension ".att" attached to the same output name.

SEE ALSO

v.import, v.in.ascii, v.in.dlg, v.in.dlg.scs, v.support

AUTHOR

R.L. Glenn, USDA, SCS, NHQ−CGIS

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

NAME

v.out.dxf − Exports GRASS vector files to DXF file format.
(GRASS Vector Data Export Program)

SYNOPSIS

v.out.dxf
v.out.dxf help
v.out.dxf input=*name* **output**=*name*

DESCRIPTION

The GRASS program *v.out.dxf* conversion program generates an ASCII DXF (AutoCad) file from a GRASS vector ASCII file. The output file is placed in the user's current working directory unless the user specifies a full pathname for the *output*.

Parameters:

input=*name*
The name of an existing GRASS vector ASCII file.
output=*name*
Name to be assigned to the DXF output file.
NOTE: DXF files output by AutoCad have the suffix **`.dxf`**

NOTES

This program does not currently read in binary files.

SEE ALSO

v.cadlabel, *v.digit*, *v.in.ascii*, *v.in.dxf*, *v.support*

AUTHOR

Charles Ehlschlaeger, U.S. Army Construction Engineering Research Laboratory,
wrote original *v.out.dxf* program in 4/89.

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

NAME

v.out.e00 − Write an Arc−Info line/polygon coverage in e00 format

SYNOPSIS

v.out.e00
v.out.e00 help
v.out.e00 input=*name* [**mapset**=*name*] [**output**=*name*]

DESCRIPTION

The *v.out.e00* program is designed to create an ESRI Arc/Info e00 ascii file from a Grass vector file.

v.out.e00 will be run non−interactively if the user specifies program arguments on the command line, using the form:

v.out.e00 input=*name* [**output**=*name*]

Alternately, the user can simply type:

v.out.e00

on the command line without program arguments. In this case, the user will be prompted for parameter values using the standard GRASS user interface described in the manual entry for *parser*.

FEATURES

v.out.e00 attempts to create an Arc/Info export file (.e00) for a line/polygon coverage. The attribut of each Grass line or area will be the Arc/Info coverage−ID. If there is no attribute, the coverage−ID will be the same as the coverage−# (numbered from 1 to the number of lines or areas).

OPTIONS

v.out.e00 requires the user to enter the following information:

Parameters:

mapset=*name*
Mapset holding vector map to be exported (Default = current)
output=*name*

Name of output file. If no name is given, *v.out.e00* will write on the standard output.

BUGS AND CAVEAT

Since e00 is NOT a public format, this program is mainly based on analysis of existing files.

NOTES

SEE ALSO

m.in.e00, s.out.e00, v.in.shape.

AUTHOR

Michel J. Wurtz, CEREG,

Ecole Nationale du Genie de l'Eau et de l'Environnement.

mw@engees.u−strasbg.fr

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

NAME

v.out.idrisi − export routine from GRASS to IDRISI
(GRASS Vector Data Export Program)

SYNOPSIS

v.out.idrisi
v.out.idrisi help
v.out.idrisi [−o] **type=**name **input=**name **output=**name

FLAG

−o IDRISI version 3 export

DESCRIPTION

v.out.idrisi This is a quick and dirty export routine from GRASS to IDRISI. The module exports binary GRASS vector files to ASCII IDRISI format. It should run fine in most situations. Any comments or suggestions are welcome.

PARAMETERS

type maptype
options: polygon,line,point
default: polygon

input input vector file

output prefix for resulting .vec and .dvc files

AUTHOR

Philip Verhagen
S t i c h t i n g R A A P
Regionaal Archeologisch Archiverings Projekt
adress: Plantage Muidergracht 14
1018 TV Amsterdam
THE NETHERLANDS
e−mail: motte@xs4all.nl

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

NAME

v.out.moss − Converts GRASS site, line, or area data into MOSS import format.
(GRASS Vector Data Export Program)

SYNOPSIS

v.out.moss

DESCRIPTION

This program produces a MOSS vector file containing GRASS site, line, or area features that have been converted into MOSS point, line, or polygon features, respectively. Only one type of data (site, line, or area) can be converted at a time. Site data can be extracted from GRASS *site lists* files or *vector* files, at the user's discretion. The resultant MOSS files will be created in the *moss* subdirectory of the current mapset.

The user will be prompted for the GRASS data type, the name of the *site lists* or *vector* file to be converted, and the name of the the MOSS vector file to be created. If line data is being converted, the user will be asked whether area edges should be processed as lines. If so, the line features produced from the area edges will not have attributes associated with them.

Upon successful completion of the data export, the number of items converted will be printed, and the user will be given the option of processing another GRASS file.

NOTES

Vector data cannot be exported until the necessary GRASS support files have been built.

This program is interactive and requires no command line arguments.

SEE ALSO

v.support

AUTHOR

Chris Emmerich, Autometric, Inc.

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

NAME

v.out.scsgef − Converts binary GRASS vector data to ASCII SCS−GEF vector data format.
"(SCS GRASS Vector Data Export Program)

SYNOPSIS

v.out.scsgef
v.out.scsgef help
v.out.scsgef input=*name* **output=***name*

DESCRIPTION

The GRASS program *v.out.scsgef* allows the user to convert GRASS vector data to SCS−GEF format, for export to other systems.

v.out.scsgef creates the SCS−GEF header, lines, text, and feature files. All files are created and placed in a *$LOCATION/gef* directory as a single UNIX file under the output name.

The following is the SCS−GEF file structure:

```
header record 1
   |     |
header record n
−head
line record 1
   |     |
line record n
−line
text record 1
   |     |
text record n
−text
feature record 1
   |     |
feature record n
−feat
```

The user will be required to use standard UNIX commands to separate this file into individual files as required by SCS−GEF specifications.

The user can run the program non−interactively by specifying all program arguments on the command line, in the form:

v.out.scsgef input=*name* **output=***name*

Parameters:

input*=*name
Name of the binary GRASS vector data file to be converted to SCS−GEF format.
output*=*name
Name to be assigned to the SCS−GEF output file created.

If the user does not give the names of an input and output file on the command line, the program will prompt the user to enter these names.

NOTES

The *v.out.scsgef* program requires that the *input* vector map layer have full topological information associated with it. This means that the GRASS program *v.support* should have been the last program to have effected any changes upon the vector map layer before it is run through *v.out.scsgef*. If this is not the case, *v.out.scsgef* will terminate with a message that *v.support* needs to be run.

SEE ALSO

v.export, *v.import*, *v.out.ascii*, *v.out.dlg*, *v.out.dlg.scs*, *v.support*

AUTHOR

R.L.Glenn, USDA, SCS, NHQ−CGIS

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

NAME

v.out.sdts − Creates an SDTS dataset conforming to the Topological Vector Profile from a GRASS vector map layer.
(GRASS Vector Data Export/Processing Program)

SYNOPSIS

v.out.sdts
v.out.sdts help
v.out.sdts [**−am**] **input**=*name* [**path**=*name*] **output**=*name*

DESCRIPTION

v.out.sdts creates an SDTS dataset that conforms to the requirements of both the federal Spatial Data Transfer Standard and the SDTS Topological Vector Profile. It creates the dataset from the files associated with a vector map layer which is specified by the user. The files that make up the SDTS dataset are output to the current directory, unless otherwise specified by the user (with the path parameter). The output dataset is in the mandatory ISO 8211 (FIPS 123) format; the ISO 8211/SDTS output files can be inspected with *m.sdts.read*.

COMMAND LINE OPTIONS

Flags:

−a
transfer Lines of "AREA type" only; omit Lines of "LINE type" from output SDTS dataset.

−m
access user−defined metadata file. This file is typically, but not necessarily, created with *v.sdts.meta*. See the discussion of SDTS data quality files and metadata below.

Parameters:

input=*name*
name of vector map layer from which the SDTS dataset wil be created.

path=*name*
full path to location for placement of output SDTS dataset. Path must exist and be writable by the user. If path is not specified, dataset will be output to the current directory.

output=*name*
four−character string to be used as prefix for each of the output SDTS files. Can be any combination of letters and digits, although letters must be upper−case.

LINE TYPES IN GRASS AND SDTS

GRASS makes a distinction between types of lines, between those that are edges of areas or polygons and those that are not. GRASS handles these types different topologically: AREA type lines carry pointers to left and right polygons, but LINE type lines carry no such pointers.

The SDTS Topological Vector Profile, however, does not distinguish line types, and does require that ALL lines carry left and right polygon references. This has meant that during the export process topology−building algorithms are applied to construct the missing topology that SDTS requires. A potential problem arises, however, with the transfer of object attributes in certain circumstances. E.g. if a polygon in GRASS is bisected by a line of type LINE, the resulting SDTS dataset will contain two polygons where only one existed in GRASS: should both these polygons be assigned the attribute of the original, now non−existent polygon? The "−a" option, which transfers only AREA type lines, works around this problem.

SDTS REQUIREMENTS: DATA QUALITY REPORTS

SDTS datasets are required to contain 5 different data quality report modules, for Lineage, Positional Accuracy, Attribute Accuracy, Logical Consistency, and Completeness. When *v.out.sdts* is run, it searches in the user's mapset's *dig_misc* directory for appropriate files, one for each module, containing narrative text in ASCII format. If found, they are converted to SDTS/ISO 8211 format and added to the export dataset; warning messages are displayed if any data quality modules are missing.

Data quality reports can be created, and installed in the proper location under *dig_misc*, with *v.sdts.meta*.

OTHER METADATA

When *v.out.sdts* is run, if the "−m" flag is set, the program searches in the *dig_misc* directory for a supplementary metadata file for the map layer being transferred. If found, its contents are incorporated in the SDTS dataset. This file can be created and installed with *v.sdts.meta*; for details see the *man* page for this program.

SDTS REQUIREMENTS: THE 'README' FILE

In addition to the files created by v.out.sdts, every SDTS transfer must contain a README file. This file is not generated by *v.out.sdts*, and must be created by hand. It should contain:

"volume name [if appropriate], date, a list of SDTS transfers (if more than one), and then for each SDTS transfer: a list of subdirectories and non−SDTS files, if approriate, the file name of the Catalog/Directory module, where it can be found, and an explanation that this file and all other SDTS files are in ISO 8211 format, and that the Catalog/Directgory module carries a complete directory to all other SDTS ISO 8211 files comprising the SDTS transfer, notes about any non−SDTS adjunct/auxiliary files, a brief explanation of the spatial domain, the purpose, authority (FIPS 173), source (e.g. agency name) and contacts within the source organization...." (*SDTS, IV: Topological Vector Profile, 6.10*).

GRASS ATTRIBUTES IN THE SDTS DATASET

The SDTS dataset produced by *v.out.sdts* contains two attribute module files. One, containing attribute module "AP00", stores global attributes, i.e., metadata items applicable to the entire transfer (most are derived from the *dig* file header). The second holds attribute module "AP01", and contains records with two fields:

ATTR_NUM contains *dig_att* integer values; and ATTR_LABEL contains the corresponding labels or descriptions from the *dig_cats* file.

RESTRICTIONS

Currently, the user can only create an SDTS dataset from a single vector map layer in his or her mapset at a time.

SEE ALSO

m.sdts.read, v.in.sdts, v.out.sdts, v.sdts.dp.cp, v.sdts.meta.cp, v.sdts.meta

AUTHORS

David Stigberg, U.S.Army Construction Engineering Research Laboratory
Tin Qian, University of Illinois

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

NAME

v.out.shape − export GRASS vector files to ESRI Shapefile

SYNOPSIS

v.out.shape [−valA] map=*name* type=*area,line,point* [mapset=*name*] [cats=*[none],string,integer,float*] [prefix=*name*] [log=*filename*] [table=*name*] [key=*name*]

DESCRIPTION

v.out.shape will export GRASS vectors files to ESRI Shapefile format.

OPTIONS

Flags:

−*v* Verbose messages

−*a* Restrict lines exported to arcs of type area−edge

−*l* Restrict lines exported to arcs of type line

−*A* Output all area rings (include unlabelled areas)

Parameters:

map=*name*
: name of input vector map. Must be level 2.(Run v.support option=build)

type=*option*
: coverage type.
Options: area, line, point

mapset=*name*
: name of mapset containing vector file (default current).

cats=*option*
: If category support is available, a third field is generated containing data of the selected type. Note `float' uses a double precision number. Otherwise ignored.
Options: none(default),string,integer,float.

prefix=*name*
: prefix for SHAPE output filenames. Default is name of vector map.

log=*filename*
: choose a logfile. If the file cannot be opened, standard error stream is used.

***table**=name*
name of database table used as attributes source.
***key**=name*
key column in table used for linking rows to shapes.

NOTES

The vector file export program, v.out.shape, will prompt you to enter the name of a GRASS vector file to be exported to SHAPE and for various options, see above.

A GRASS vector file to be exported to SHAPE may contain POINT, LINE and POLYGON coverage. If you select more types for conversion, more layers will be produced with the appended suffixes **_line**, **_point** and **_area**. Lines used as area edges however are not duplicated in the lines layer. If one layer is empty the unused files are deleted.

If *table=* option is specified, dbf table structure and data is used for output dbf table. *key=* option is name of column used as link between table rows and GRASS categories. Database connection must be established by db.connect first. Supported column types are numbers and strings (int, double, char, varchar,...). Time and other types are not supported.

When exporting area features, the *−A* flag specifies that all areas, whether labelled or not, should be exported. This may result in exporting area features that really represent "holes" in other area features.

BUGS

If data are read from table, problem with char and varchar column types may occur. For example for MySQL column width is set according to first record instead of column size. PostgreSQL tables are OK. Problem is in ODBC driver and database specific drivers.

SEE ALSO

v.in.shape, m.in.e00, v.out.e00, v.in.arc, v.out.arc, v.out.dlg, db.connect

AUTHOR

David D. Gray
<ddgray@armadce.demon.co.uk>

Depends on shapelib−1.2.8 or later by Frank Warmerdam.

DBMI support by Radim Blazek

NAME

v.patch − Creates a new binary vector map layer by combining other binary vector map layers.
(GRASS Vector Program)

SYNOPSIS

v.patch
v.patch help
v.patch input=*name*[,*name*,...] **output**=*name*

DESCRIPTION

v.patch allows the user to combine any number of vector map layers together to create one composite vector map layer.

OPTIONS

Parameters:

input=*name,name, ...*
Name(s) of input vector map(s) to be patched together.
output=*name*
Name assigned to composite "patched" vector output map.

The program will be run non−interactively if the user specifies the names of the vector map(s) to be patched and the name of an output file to store the resulting composite patched vector map on the command line, in the form:

v.patch input=*name*[,*name*,...] **output**=*name*

Alternately, if the user runs *v.patch* without specifying input and output file names on the command line (by typing simply **v.patch**), the program will prompt the user for inputs using the standard GRASS interface described in the manual entry for *parser*.

NOTES

The vector map layers to be patched together must exist in the user's current mapset search path, and the composite vector map layer name given must not already exist in the user's current mapset.

After running *v.patch*, the header file will contain only information taken from the first *input* file name given

in the string **input**=*name,name, ...*, with the exception of the geographic region's edge information, and the scale and threshold information. (The user's current geographic region settings are ignored; this information is instead extracted from the vector file headers.) In the new composite vector map layer, the boundaries of the geographic region will be expanded to encompass all of the geographic area included in the map layers being patched, and the scale will be set equal to the smallest (i.e., most gross) scale used by any of the patched map layers (this will affect default node−snapping thresholds). The map threshold is calculated automatically from the map scales given in the file headers, and (currently) is not used directly. The composite vector map layer's header will probably need to be edited; this can be done from within the GRASS program *v.digit*.

The GRASS programs *v.mkgrid* and *v.mkquads* can be used to ensure that the borders of the maps to be patched together align neatly.

Any vectors that are duplicated among the maps being patched together (e.g., border lines) will have to be edited or removed after *v.patch* is run. Such editing can be done using *v.digit*.

After running *v.patch* the user must run *v.support* on the composite vector map layer in order to create a **dig_plus** (topology) file for it. At this time, you can request that a *very* small snapping threshold be used, to cause the nodes that match up across vector map layers to snap together without affecting the integrity of the remainder of the vector map layer.

BUGS

The **dig_cats** and **reg** file information for the maps being patched together is not copied to the composite, patched map layer. The user should therefore run *v.support* on the output file produced by this program.

SEE ALSO

v.digit, *v.in.ascii*, *v.mkgrid*, *v.mkquads*, *v.support*, and *parser*

AUTHOR

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

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

NAME

v.plant − module to insert intermediate points on GRASS vector lines.
(GRASS Vector Program)

SYNOPSIS

v.plant
v.plant help
v.plant map=*name*

v.plant is an interactive program.

DESCRIPTION

This report describes the use and operation of the program *v.plant*, used with the GRASS GIS system. The report contains (i) a discussion of the problem which motivated the development of the program, (ii) a description of the method used, (iii) the manual page describing the usage of the program, (iv) a listing of the source−code.

Introduction

Map projections

Projections are representations of information defined on curved surfaces (usually spheroids) in two−dimensional space. This allows the description of the location information in cartesian or other plane coordinate systems. Particular projections are defined by precise mathematical relations, which allows conversion of maps from between projections to be achieved using straightforward mathematical operations [1].

Projecting line segments

When converted from one map projection to another, straight−lines become curves. Thus, a line segment which can be described by four parameters only (eg the coordinates of the end points) in one projection becomes an arc which needs more parameters in other projections. In GIS systems which describe arcs as a sequence of line segments, this means that intermediate points between the ends of the segment are used. The easiest method of projecting a line accurately is to include intermediate points in the description in the source projection, so that all these points are included in the target map.

Context within the GRASS GIS

Vector maps (lines and arcs) in the GRASS GIS are represented by a relatively straightforward format. This is

exemplified in the ascii version of vector files, as produced by *v.out.ascii* [2] and described in the programmer's manual [3]. Individual vector segments are described by a sequence of coordinate pairs, with the number of coordinated pairs for the segment recorded in a brief header for each segment. The program *v.prune* [4] is provided to remove points which are considered redundant through being too close together within a segment. *v.plant* has been designed as the complement of this, to insert extra points at a specified spacing along straight−line segments.

Design of v.plant

v.plant has been implemented as a Bourne shell script. The basic procedure is as follows:

1. Write out the existing map in the *dig_ascii* format;

2. Use an included *awk* script to process each vector segment in turn, moving from point to point along the segment. If a span between two adjacent points is greater than the specified threshold (in map units) extra points are inserted collinearly.

3. The modified *dig_ascii* file is re−imported, overwriting the original file.

Because the original *dig* file is overwritten, all the support files are preserved. Since the vector segments remain in the same sequence, and all the original points remain in the modified map, including the end−points of each segment, all topology, attributes, etc are maintained.

The usage is exemplified in the attached manual page. The source code is also attached, which may be examined for a detailed understanding of the method.

TODO

A command−line version of *v.plant* would be highly desirable.

SEE ALSO

v.prune v.support

References

[1] Evenden, G.I. (1990) Cartographic projection procedures for the UNIX environment − a user's manual. USGS Open−File Report 90−284 (Also see Interim Report and 2nd Interim Report on Release 4, Evenden 1994).

[2] Higgins, M. & Westervelt, J. v.out.ascii − Converts a binary GRASS vector map layer into an ASCII GRASS vector map layer. GRASS 4.1 documentation (main section).

[3] Shapiro, M., Westervelt, J, Gerdes, D., Larson, M. & Brownfeld, K.R. (1993) GRASS 4.1 programmer's manual. US Army Construction Engineering Research Laboratory.

[4] Gerdes D., v.prune − Prunes points from binary GRASS vector data files. GRASS 4.1 documentation (main section).

Copyright © 1997 AGCRC & CSIRO

CSIRO Exploration & Mining Report 239F, March 1996.

S.J.D. Cox

AGCRC, CSIRO Exploration & Mining, Nedlands, WA

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

NAME

v.proj − Allows projection conversion of vector files.

SYNOPSIS

v.proj
v.proj help
v.proj [**−sl**] **input**=*name* ***location***=*name [**dbase**=name] [**mapset**=name] [***out**=name*]

DESCRIPTION

allows a user to convert a vector map in a specified mapset of a specified location (different from current) with projection of input location to the vector map in a current mapset of current location with projection of current location (both projections are defined by corresponding PROJ_INFO files).

v.proj will create *dig*, *dig_att*, and *dig_cats* directories in the output mapset, if they do NOT exist. Map files for *dig*, *dig_att*, and *dig_cats* are also created for the new map layer.

Flags:

−s
: Automatically run "v.support" on newly created vector file.

−l
: List vector files in input location and exit

Parameters:

***input**=name*
: Name of the input vector map.

***location**=name*
: Name of the location containing input vector map

***dbase**=name*
: path to GRASS database of input location

***mapset**=name*
: Name of the mapset containing input vector map

***output**=name*
: Name of the output vector map.

If the user simply types **v.proj** without specifying parameter values on the command line, the program will prompt the user to enter these.

NOTES

If **out** is not specified it is set to be the same as input map name.
If **dbase** is not specified it is assumed to be the current database.
If **set** is not specified, its name is assumed to be the same as the current mapset's name.

v.proj supports transformations between nad27 and nad83. It performs the transformation automatically based on the datum specified in the mapset region. See BUGS section.

BUGS

Currently v.proj does not support general datum conversions. It only supports conversion between nad27 and nad83, and only within the CONUS conversion grid; 20 degrees to 50 degrees north latitude, 63 degrees to 131 degrees west longitude. That covers all of the conterminous USA plus Mexico north of Mexico City and most of Canada farther south than Winnipeg, Manitoba.

SEE ALSO

m.proj, *r.proj*, *s.proj*, *g.setproj*, *i.rectify*, *i.rectify3*, *r.support*, *r.stats*, *s.sample*, *s.surf.idw*, *s.surf.rst*

AUTHORS

Irina Kosinovsky, US ARMY CERL
M.L. Holko, USDA, SCS, NHQ−CGIS
R.L. Glenn, USDA, SCS, NHQ−CGIS

Last changed: $Date: 2002/04/22 16:08:29 $

NAME

v.prune − Prunes points from binary GRASS vector data files.
(GRASS Vector Program)

SYNOPSIS

v.prune
v.prune help
v.prune [**−i**] **input**=*name* **output**=*name* **thresh**=*value*

DESCRIPTION

The GRASS program *v.prune* allows the user to remove extra points from a vector file. This allows users to reduce disk space required by a vector file and still have data accuracy within a given tolerance.

OPTIONS

Flag:

−i
The pruning threshold value is specified in map inches, rather than in data base units on the ground.

Parameters:

input=*name*
Name of binary GRASS vector file containing data to be pruned.
output=*name*
Name to be assigned to new, pruned vector output file.
thresh=*value*
Threshold value used for pruning.

The program will be run non−interactively if the user specifies all parameters and (optionally) the *−i* flag on the command line, using the form:

v.prune [**−i**] **input**=*name* **output**=*name* **thresh**=*value*

If the user simply types **v.prune** without specifying program arguments on the command line, the program will prompt the user to enter parameter values.

NOTES

The threshold value is the same as the *v.digit* pruning threshold. This is specified in data base units on the ground (e.g., in ground meters for UTM data bases). The threshold can also be specified in inches on the map, and the program will convert these to data base ground units using the *scale* in the vector file. If you specify the scale in map inches rather than in ground units, you must specify that inches are used by setting the $-i$ flag. The input vector data layer will be read and the resultant pruned vector data layer will be placed into a newly created output file whose name is specified by the user, leaving the original vector map unchanged.

The pruning algorithm throws away redundant points within the specified threshold. It works on each vector separately, working from node to node. It does not change the position or number of nodes.

SEE ALSO

v.in.ascii, *v.clean*, *v.digit*, *v.prune*, *v.rmdup*, *v.spag*, *v.support*

AUTHOR

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

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

NAME

v.psu − Program to build PSU polygon and PSU sites from single labeled line segment.
(SCS GRASS Vector Program)

SYNOPSIS

v.psu
v.psu help
v.psu input=*name* **output**=*name* **psu**=*name* **subj**=*name* [**xdist**=*name*] [**ydist**=*name*]

DESCRIPTION

This program builds a complete PSU polygon and PSU points from a single labeled line segment identifying the west edge of the PSU. The resultant polygon and points will be automatically labeled based on the original line segment label.

Parameters:

input=*name*
Vector input file name.
output=*name*
Vector output file name.
psu=*name*
psu_data file name.
subj=*name*
Subject file name.
xdist=*value*
x distance.
ydist=*value*
y distance.

EXAMPLE

psu.vect in=*spri* **out**=*spri.psu* **psu**=*SD009.PNT* **subj**=*bonhomme.psu*

NOTES

Refer to SCS PSU Digitizing Manual for examples and further instruction.

AUTHOR

Paul H. Fukuhara, USDA SCS National Cartographic Center

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

NAME

v.psu.subj − Converts SCS ISU PSU offset file to GRASS subject file.
(SCS GRASS Vector Program)

SYNOPSIS

v.psu.subj
v.psu.subj help
v.psu.subj input=*name*

DESCRIPTION

This program converts a PSU offset file as distributed by SCS into a GRASS subject file to be used with the SCS PSU digitizing package.

Parameters:

input=*name*
PSU offset file name.

SEE ALSO

Refer to the SCS PSU Digitizing Manual for examples and further instruction.

AUTHOR

Paul H. Fukuhara, USDA SCS National Cartographic Center

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

NAME

v.random − Creates a GRASS site_lists file of randomly placed symbols (sites) within a GRASS vector area. *(SCS GRASS Vector Program)*

SYNOPSIS

v.random
v.random help
v.random [**−nsv**] **map=***name* [**site=***name*] **dot=***name*

DESCRIPTION

allows a user to create a GRASS site_lists file containing sites randomly placed within an area. This program is designed for demographic map areas, and may NOT perform well for resource maps (very irregularly shaped polygons). The user provides the program with: a file (dot=) containing area category names [default] (the −n option allows the use of the area category number) and a count of dots for that name; input vector file name (map=), and a site_lists file name (site=). All sites in the site_lists file will have the same category code (1). *v.random* is made to work with the *mapgen* mapping package to create special symbols at the site locations.

Flags:

−n
: Use category numbers NOT names.

−s
: Determine optimum dot size.

−v
: Verbose mode.

Parameters:

map=*name*
: Input vector file name.

site=*name*
: Output site_lists file name.

dot=*name*
: File name containing labels and dot counts.

FORMATS

The dotfile file format is:

```
−Using Names−    −Using Numbers−
area name_1:3    category_num_1:3
area name_2:15   category_num_1:15
area name_3:5    category_num_1:5
area name_n:54   category_num_1:54
```

SEE ALSO

mapgen

AUTHOR

R.L. Glenn, USDA, SCS, NHQ−CGIS
M.L. Holko, USDA, SCS, NHQ−CGIS

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

NAME

v.reclass − Creates a new map layer whose category values are based upon the user's reclassification of categories in an existing vector map layer. (SCS GRASS Vector Program)

SYNOPSIS

v.reclass
v.reclass help
v.reclass [−d] input=*name* **output**=*name* **type**=*area, line, or site* [**title**=*name*]

DESCRIPTION

v.reclass creates an output map layer based on an input vector map layer. The output map layer will be a reclassification of the input map layer based on reclass rules input to *v.reclass*. A title for the output map layer may be (optionally) specified by the user.

The reclass rules are read from standard input (i.e., from the keyboard, redirected from a file, or piped through another program).

OPTIONS

The user can run this program either non−interactively or interactively. The program will be run non−interactively if the user specifies program arguments and flag settings on the command line using the form:

v.reclass [−d] input=name output=name type=area, line, or site [title=name]

After the user types in the above information on the command line, the program will (silently) prompt the user for reclass rules to be applied to the input map layer categories. The form of these rules is described in further detail in the sections on non−interactive program use reclass rules and examples, below.

Flag:

−d
Dissolve common boundaries between areas

Parameters:

input=name
Vector input map name.
output=name

Vector output map name.

type=name

Select type of vector input file.
Options: area, line, site

Alternately, the user can simply type *v.reclass* on the command line, without program arguments. In this case, the user will be prompted for all needed inputs. Before using *v.reclass* one must know the following:

1 The new categories desired; and, which old categories fit into which new categories.

2 The names of the new categories.

INTERACTIVE PROGRAM USE: EXAMPLE

Suppose we want to reclassify the vector map layer roads, consisting of five categories, into the three new categories: paved roads, unpaved roads, and railroad tracks. The user is asked whether the reclass table is to be established with each category value initially set to 0, or with each category value initially set to its own value. A screen like that shown below then appears, listing the categories of the roads vector map layer to be reclassified and prompting the user for the new category values to be assigned them.

```
  ENTER NEW CATEGORY NUMBERS FOR THESE CATEGORIES

OLD CATEGORY NAME              OLD             NEW
                               NUM             NUM
no data                        0               0__
Hard Surface, 2 lanes          1               0__
Loose Surface, 1 lane          2               0__
Improved Dirt                  3               0__
Unimproved Dirt Trail          4               0__
Railroad, single track         5               0__

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

In the following screen the new category values have been entered beside the appropriate old category names. Cells assigned category values 2, 3, and 4 in the old vector map layer are now assigned the new category value 2 in the reclassed map; cell data formerly assigned to category value 5 in the old vector map map are now assigned the new category value 3 in the reclassed map.

```
  ENTER NEW CATEGORY NUMBERS FOR THESE CATEGORIES

OLD CATEGORY NAME              OLD             NEW
                               NUM             NUM
no data                        0               0__
Hard Surface, 2 lanes          1               1__
Loose Surface, 1 lane          2               2__
Improved Dirt                  3               2__
Unimproved Dirt Trail          4               2__
Railroad, single track         5               3__

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

Hitting the escape key <ESC> will bring up the following screen, which prompts the user to enter a new title and category label for the newly reclassed categories.

```
        ENTER NEW CATEGORY NAMES FOR THESE CATEGORIES

TITLE:  Roads Reclassified
     CAT         NEW CATEGORY NAME
     NUM
      0          no data
      1          Paved Roads
      2          Unpaved Roads
      3          Railroad, single track

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

Based upon the information supplied by the user in the above sample screens, the new output map and supporting category files are created.

NON−INTERACTIVE PROGRAM USE: RECLASS RULES

In non−interactive program use, the names of an input map, output map, type of input map, and output map title are given on the command line. However, the reclass rules are still read from standard input (i.e., from the keyboard, redirected from a file, or piped through another program).

Once the user has specified an input vector map layer, output map layer name, type of input map layer, and (optionally) output map layer title by typing:

v.reclass [−d] input=name output=name type=area, line, or site [title=name]
Each line of input must have the following format:

input_categories=output_category [label]
where the input lines specify the category values in the input vector map layer to be reclassified to the new output_category category value. Specification of a label to be associated with the new output map layer category is optional. If specified, it is recorded as the category label for the new category value. The equal sign = is required. The input_category(ies) may consist of single category values or a range of such values in the format "low thru high." The word "thru" must be present.

A line containing only the word end terminates the input.

NON−INTERACTIVE PROGRAM USE: EXAMPLES

The following examples may help clarify the reclass rules.

1 This example reclassifies categories 1, 3 and 5 in the input vector map layer to category 1 with category label "poor quality" in the output map layer, and reclassifies input vector map layer categories 2, 4, and 6 to category 2 with the label "good quality" in the output map layer.

```
1 3 5   =   1   poor quality
2 4 6   =   2   good quality
```

2 This example reclassifies input vector map layer categories 1 thru 10 to output map layer category 1, input map layer categories 11 thru 20 to output map layer category 2, and input map layer categories 21 thru 30 to output map layer category 3, all without labels.

```
 1 thru 10   =   1
11 thru 20   =   2
21 thru 30   =   3
```

3 Subsequent rules override previous rules. Therefore, the below example reclassifies input vector map layer categories 1 thru 19 and 51 thru 100 to category 1 in the output map layer, input vector map layer categories 20 thru 24 and 26 thru 50 to the output map layer category 2, and input vector map layer category 25 to the output category 3.

```
1 thru 100    =   1   poor quality
20 thru 50    =   2   medium quality
    25        =   3   good quality
```

4 The previous example could also have been entered as:

```
 1 thru 19  51 thru 100    =   1   poor quality
20 thru 24  26 thru 50     =   2   medium quality
           25              =   3   good quality
```

or as:

```
 1 thru 19     =   1   poor quality
51 thru 100    =   1
20 thru 24     =   2
26 thru 50     =   2   medium quality
    25         =   3   good quality
```

The final example was given to show how the labels are handled. If a new category value appears in more than one rule (as is the case with new category values 1 and 2), the last label which was specified becomes the label for that category. In this case the labels are assigned exactly as in the two previous examples.

NOTES

The *v.reclass* program generates a new vector map layer. The *v.support* program will have to be run on the newly created vector map layer to build topology. The dissolve *[−d]* option will work on only those areas which are of the same conversion category value. If a map area is inside (island) a converted area and is NOT converted to the same value, its boundaries are output to the resultant map.

WARNING

Category values which are not explicitly reclassified to a new value by the user will be reclassified to 0.

SEE ALSO

v.support

AUTHORS

Paul H. Fukuhara, USDA SCS, NCG−GSS
R. L. Glenn, USDA SCS, NHQ−CGIS

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

NAME

v.reclass.inf − Generate new vector map layer derived from attribute data in the currently selected database. *(GRASS−RDBMS Vector Interface Program)*

SYNOPSIS

v.reclass.inf
v.reclass.inf help
v.reclass.inf [−d] sql=*name* **key**=*name* **type**=*name* **input**=*name* **[output**=*name]*

DESCRIPTION

v.reclass.inf generates a new vector map layer based on the results of one or more queries to the currently selected database. The user constructs a series of mutually exclusive SQL select statments designed to return groups of records from the database. Each group of records should be internally consistent in terms of attribute characteristics specified by the user in the SELECT clause. These groups should also be mutually exclusive, thereby insuring that a row returned by one SELECT clause is not also returned by a subsequent SELECT clause. Each group of records therefore forms the basis for a single category in the resulting GRASS vector map. v.reclass.inf processes each SELECT statement in order returning groups of records which will form a single category in the resulting map. As each SELECT statement is processed the group of records returned receives a common category value. The category value is incremented by one for each subsequent SELECT statment which is processed. The resulting reclass map will have one category for each of the original SELECT statements.

For example, the rows associated with the first SELECT statement will be assigned to category 1, those associated with the second SELECT statement will be assigned to category 2 and so on. The output map will contain only those line segments associated with database rows returned by the SELECT statement(s).

COMMAND LINE OPTIONS

Flags:

−d
Disolve common boundaries between reclassed areas.
Parameters:

sql=*filename*
Name of file containing SQL query statements.
key=*database_column_name*
Key column in db.
type=*area/line.*

Key column in db.

***input**=map*

Name of existing vector file to be reclassed using query output.

***output**=map*

Name of new raster (reclass), file.

EXAMPLE: produces vector map of primary and secondary roads.

v.reclass.inf sql=*vect.sql* **key**=*tlid* **input**=*t.roads.inf* **output**=*t.roads.12*

vect.sql:

```
SELECT UNIQUE tlid,cfcc FROM type1
WHERE cfcc MATCHES "A1*"
ORDER BY tlid;
SELECT UNIQUE tlid,cfcc FROM type1
WHERE cfcc MATCHES "A2*"
ORDER BY tlid
```

BUGS

None known.

NOTE

This program requires the Informix database software.

SEE ALSO

g.column.inf, g.select.inf, g.stats.inf, g.table.inf, d.rast.inf, d.site.inf, d.vect.inf, d.what.r.inf, d.what.s.inf, d.what.v.inf, r.rescale.inf

AUTHOR

James A. Farley, Wang Song and W. Fredrick Limp University of Arkansas, CAST

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

NAME

v.reclass.pg − Generate new vector map layer derived from attribute data in the currently selected database. (GRASS−RDBMS Vector Interface Program)

SYNOPSIS

v.reclass.pg
v.reclass.pg help
v.reclass.pg [**−d**] **input**=name **key**=name **tab**=name **col**=name [**where**=name] [**output**=name] **type**=name
v.reclass.pg [−sd] sql=file **input**=name [**output**=name] **type**=name

DESCRIPTION

v.reclass.pg reclasses vector maps according to the SELECT query. The input map may be of vector area, line or vector site type, resulting in the reclass map of the same type.
The sql file if chosen to input contains a single line like: 'select key_col, reclass_col from info_tab [where *clause*]'.
The users needs to either input two names of the columns interactively or feed them from the input sql file. The first name is the old map key column, the second one is the key for the new map. Additionally, a clause to choose only some vectors to be reclassed from the input map can be imposed by user, either interactively or from the input file. Both columns used for creating the reclass rules must be of numeric (integer) type.
If the user omitted the output map name, the reclass map would only be displayed on monitor and not created.

COMMAND LINE OPTIONS

Flag:

−d
: Dissolve common boundaries between reclassed areas

Parameters:

sql=filename
: Name of file containing SQL query statements

key=databasecolumnname
: Key column in db

tab=name
: Table containing [col]

col=name
: Column to base reclass on

where=name

Where clause for query (ie. where col='paved')

type=area/line/site

Select area, line or site as type of the input/output vector map

input=map

Name of existing vector file to be reclassed using query output.

output=map

Name of new raster (reclass) file

EXAMPLE:

1. Reclass to vector map of quartiles from forest stands map (*kuruma_id*).

v.reclass.pg −s −d sql=reclass.sql input=kuruma_id output=kuruma_quart type=area

and reclass.sql is:

select rec_id, quartnum from info_kuruma

2. Reclass to vector map of forest types (*kuruma_oak*) from map of forest plots (*kuruma_id*) taking only oak (types 32−37).

v.reclass.pg −d kuruma_id key=rec_id col=type_id tab=info_kuruma where='type_id > 31 and type_id < 38' output=kuruma_oak type=area

BUGS

None known.

NOTE

This program requires the PostgreSQL database software. It uses other GRASS modules ***v.reclass*** and ***d.vect*** launched from inside.

SEE ALSO

g.column.pg, *g.select.pg*, *g.stats.pg*, *g.table.pg*, *d.rast.pg*, *d.site.pg*, *d.vect.pg*, *d.what.r.pg*, *d.what.s.pg*, *d.what.v.pg*, *r.reclass.pg*, *r.rescale.pg*

AUTHOR

Original Informix SQL−tools: James A. Farley, Wang Song, and W. Fredrick Limp University of Arkansas, CAST

Postgres modifications: Janne Soimasuo, Faculty of Forestry, University of Joensuu, Finland.

Updated to GRASS 5 by Alex Shevlakov (sixote@yahoo.com)

NAME

v.reclass − Changes vector category values for an existing vector map.
(SCS GRASS Vector Program)

SYNOPSIS

v.reclass
v.reclass help
v.reclass [**−d**] **type**=*name* **input**=*name* **output**=*name* **file**=*name*

DESCRIPTION

v.reclass allows a user to create a new vector map based on the reclassification of an existing vector map. The user provides the program with a category conversion file, input vector map name, an output vector map name, and the type of input map. There is an option (d) to dissolve common boundaries between adjoining map areas of the same re−classed category value.

Note: The dissolve option will work on only those areas which are of the same conversion category value. If a map area is inside (island) a converted area and is NOT converted to the same value, its boundaries are output to the resultant map.

COMMAND LINE OPTIONS

Flags:

−d
Dissolve common boundaries (default is no) .

Parameters:

type=*name*
Select area, line, or site.
Options: area, line, site
input=*name*
Vector input map name.
output=*name*
Vector output map name.
file=*name*
Text file name for category conversion.

EXAMPLE

v.reclass −d input=*soils* **output**=*soil_groupa* **type**=*area*
file=*convert1,convert2,convert3*
the input map *soils* contains 15 map area categories,
the conversion files contain :

```
  convert1      convert2        convert3
   1:1             3:2             2:3
  10:1             4:2             7:3
  12:1             5:2             8:3
  15:1             6:2             9:3
                  11:2            13:3
                  14:3
```

Produces a new vector area file *soil_groupa* containing 'area' boundaries from *soilss* with area category values of 1,10,12,15 changed to category 1; values of 3−6,11 changed to 2; and values 2,7−9,13−14 changed to 3. Any common boundaries are dissolved.

NOTE:
The format for "category label" is:

```
if NO SPACES in the labels      if SPACES in the labels
         Abc                       area name 1:
         Def1                      area name 2:
         12A                       .    .   .
         WWd                     area name n:
```

The format for "category value" is:
1
10
12
15

INTERACTIVE MODE

v.reclass
o The first question asked is the map type:
Enter the type of map (area, line, site) [area] :
The default is for areas.
o The next question is if common boundaries are dissolved :
Do you want common boundaries dissolved ?(y/n) [n]

The default is no, meaning all exiting boundaries will be retained.
o The next question is an option for using category labels :
Do you want to use category names ?(y/n) [n]

The default is no, meaning the user will be using category values.
o The next question asks for the name of the input map :
Enter vector map
Enter 'list' for a list of exiting vector files
Hit RETURN to cancel request
>

Any map is the user's search list is available.
o The next question asks for the name of the output map :
Enter name for resultant vector map
Enter 'list' for a list of exiting vector files
Hit RETURN to cancel request
>

If the name is for an existing map, the user will be asked if the map can be over−written.
o The next question asks if a file of labels/categories is to be used :

If names was selected previously:
Do you want to use a file of labels?(y/n) [n]
If names was NOT selected previously:
Do you want to use a file of categories?(y/n) [n]
o At this time the user will be asked for category 1 information:
1. be asked to enter a file name − if file input was selected, or
2. be asked to enter the information manually.
Then the user will be asked for category 2 information:
Then the user will be asked for category 3 information:
.
. Then the user will be asked for category n information:

o When no entry is provided the program will begin.

NOTES

This manual−page differs from the v.reclass documentation available via *g.manual* from the GRASS prompt (Comment WW).

SEE ALSO

v.extract

AUTHOR

R.L. Glenn, USDA, SCS, NHQ−CGIS

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

NAME

v.report − Generates statistics for vector files.
(GRASS Vector Program)

SYNOPSIS

v.report
v.report help
v.report [**−ehfq**] **map**=*name* **type**=*name [***units**=*name*[,*name*,...]] [**pl**=*value*] [**pw**=*value*]

DESCRIPTION

generates a table showing the area present in each of the categories of a user−selected data layer. Area is given in hectares, square meters, and square kilometers. If the units option is used, area is given in acres, square feet, and square miles. *v.report* works on the map data; therefore, the current region and mask are ignored.

Flags:

−e
: Use scientific format for the numbers that are too long to fit in the tab table field if their decimal form is used.

−h
: Suppress page headers.

−f
: Use formfeeds between pages.

−q
: Run quietly.

Parameters:

map=*name*
: vector map to report on.

type=*name*
: Type of vector map.
Options: area, line, site

units=*name*,*name*,...
: mi(les), f(eet), me(ters), k(ilometers), a(cres), h(ectacres), c(ounts).

pl=*value*
: Page length, in text lines.
Default: 0

pw=*value*

Page width, in characters.
Default: 79

EXAMPLE

Following is a sample table generated by *v.report* where type=area.
v.report −hfq map=*soils* **type**=*area* **units**=*c,me*

```
+----------------------------------------------------------------------+
|                     VECTOR MAP CATEGORY REPORT                       |
|LOCATION: spearfish                          Wed Apr 24 15:59:22 1991|
|----------------------------------------------------------------------|
|          north: 4928000    east: 609000                              |
|WINDOW    south: 4914000    west: 590000                              |
|          res:       100    res:     100                              |
|----------------------------------------------------------------------|
|MAP:   soils in grass                                                 |
|----------------------------------------------------------------------|
|             Category Information              |     |        square|
| #|description                                  |count|        meters|
|----------------------------------------------------------------------|
| 0|no data. . . . . . . . . . . . . . . . . . . |    0|0.00000000e+00|
| 1|AaB. . . . . . . . . . . . . . . . . . . . . |    1|1.71169450e+05|
| 2|Ba . . . . . . . . . . . . . . . . . . . . . |    2|5.85232360e+05|
| 3|Bb . . . . . . . . . . . . . . . . . . . . . |    5|6.65371740e+05|
| 4|BcB. . . . . . . . . . . . . . . . . . . . . |    7|9.18686470e+05|
| 5|BcC. . . . . . . . . . . . . . . . . . . . . |   13|1.23052087e+06|
| 6|BeE. . . . . . . . . . . . . . . . . . . . . |   15|7.16046145e+06|
| 7|BhE. . . . . . . . . . . . . . . . . . . . . |   23|2.30631653e+06|
| 8|BkD. . . . . . . . . . . . . . . . . . . . . |    9|1.26339976e+06|
| 9|CBE. . . . . . . . . . . . . . . . . . . . . |   18|3.53705437e+07|
|10|CaD. . . . . . . . . . . . . . . . . . . . . |   15|2.95884910e+06|
|11|CaE. . . . . . . . . . . . . . . . . . . . . |   18|3.50254750e+06|
|12|Cc . . . . . . . . . . . . . . . . . . . . . |    2|1.11726840e+05|
|13|GBE. . . . . . . . . . . . . . . . . . . . . |    8|4.34185839e+07|
|14|GaD. . . . . . . . . . . . . . . . . . . . . |    3|1.12212602e+06|
|15|GcD. . . . . . . . . . . . . . . . . . . . . |    2|2.17705920e+05|
|16|GdE. . . . . . . . . . . . . . . . . . . . . |    2|1.81687130e+05|
:     :                                         :                     :
:     :                                         :                     :
|54|Wb . . . . . . . . . . . . . . . . . . . . . |    5|3.81216040e+06|
|55| . . . . . . . . . . . . . . . . . . . . . . |    1|1.08310000e+02|
|----------------------------------------------------------------------|
|TOTAL                                          |  735|2.82182547e+08|
+----------------------------------------------------------------------+
```

Following is a sample table generated by v.report type=line.
v.report −hfq map=*roads* **type**=*line* **units**=*c,me,f*

```
+----------------------------------------------------------------------+
|                     VECTOR MAP CATEGORY REPORT                       |
|LOCATION: spearfish                          Wed Apr 24 16:34:24 1991|
|----------------------------------------------------------------------|
|          north: 4928000    east: 609000                              |
|WINDOW    south: 4914000    west: 590000                              |
|          res:       100    res:     100                              |
|----------------------------------------------------------------------|
|MAP:   roads in grass                                                 |
|----------------------------------------------------------------------|
|      Category Information       |     |             |              |
```

```
|#|description                    |count|        meters|          feet|
|-------------------------------------------------------------------|
|0|no data. . . . . . . . . . . . |    5| 0.0000000e+00| 0.0000000e+00|
|1|interstate . . . . . . . . . . |   51| 7.5353920e+04| 2.4722114e+05|
|2|primary highway, hard surface. |   60| 4.4100270e+04| 1.4468417e+05|
|3|secondary highway, hard surface|   42| 3.4949180e+04| 1.1466127e+05|
|4|light−duty road, imprvd surface|  512| 1.8597865e+05| 6.1015875e+05|
|5|unimproved road. . . . . . . . |  153| 2.2649644e+05| 7.4308952e+05|
|-------------------------------------------------------------------|
|TOTAL                          |  823| 5.6687846e+05| 1.8598149e+06|
+-------------------------------------------------------------------+
```

Following is a sample table generated by *v.report* where type=sites.
v.report −hfq map=*bugsites* **type**=*sites* **units**=*c*
NOTE: count is the ONLY option available for site maps.

```
+-------------------------------------------------------------------+
|                    VECTOR MAP CATEGORY REPORT                     |
|LOCATION: spearfish                         Wed Apr 24 16:41:17 1991|
|-------------------------------------------------------------------|
|          north: 4928000    east: 609000                           |
|WINDOW    south: 4914000    west: 590000                           |
|          res:       100    res:     100                           |
|-------------------------------------------------------------------|
|MAP:   bugsites in grass                                           |
|-------------------------------------------------------------------|
|                    Category Information                     |     |
| #|description                                                |count|
|-------------------------------------------------------------------|
| 1| . . . . . . . . . . . . . . . . . . . . . . . . . . . . .|    1|
| 2| . . . . . . . . . . . . . . . . . . . . . . . . . . . . .|    1|
| 3| . . . . . . . . . . . . . . . . . . . . . . . . . . . . .|    1|
| 4| . . . . . . . . . . . . . . . . . . . . . . . . . . . . .|    1|
| 5| . . . . . . . . . . . . . . . . . . . . . . . . . . . . .|    1|
| 6| . . . . . . . . . . . . . . . . . . . . . . . . . . . . .|    1|
| 7| . . . . . . . . . . . . . . . . . . . . . . . . . . . . .|    1|
:     :                                    :                     :
:     :                                    :                     :
|90| . . . . . . . . . . . . . . . . . . . . . . . . . . . . .|    1|
|-------------------------------------------------------------------|
|TOTAL                                                      |   90|
+-------------------------------------------------------------------+
```

After the table, the user is given the options of printing out a copy of the report and of saving the report in a file.

AUTHOR

R.L. Glenn , USDA, SCS, NHQ−CGIS

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

NAME

v.rm.dangles − Remove lines of specified length from GRASS vector file. (GRASS Vector Program)

SYNOPSIS

v.rm.dangles
v.rm.dangles help
v.rm.dangles input=*name* **output**=*name* [**maxlength**=*value*]

DESCRIPTION

v.rm.dangles removes lines from a binary vector file (input) which have a length less than or equal to maxlength. The length is computed as sum(sqrt(x^2+y^2)) for the entire length of the line in the units of the map. Lines that have attached lines at BOTH ends are not removed, regardless of length. (Note that nodes that have more than 1 line attached show as red in v.digit. Nodes with a single line show as green.)

Specifying maxlength of less than or equal 0 (default is −1) disables length checking and only lines with attached lines at both ends are output. Note that the lines at the ends of a string of lines will be eliminated because they have 2 lines attached at one node (endpoint) and only 1 line attached at the other node. For example, 5 lines attached in a row would have the first and last line eliminated leaving 3 lines when using maxlength = −1. Running v.rm.dangles maxlength = −1 again would remove 2 more lines, leaving 1 line in this example.

See v.build.polylines for building single lines from multiple attached lines.

To remove small segments on the order of the map resolution (from a raster to vector conversion, for example) consider specifying maxlength to be slightly greater than sqrt(2) times the map resolution (assuming the Easting and Northing resolutions are the same).

The input file is NOT changed. The input file is coppied to the output file with appropriate lines removed.

This program is very simple. It copies header information to the new map file but does not copy catagory or attribute information.

Parameters:

input=*name*
Name of a binary vector file
output=*name*
Name given to binary vector output file.
maxlength=*value*
Threshold length for deleting lines. Default value is −1

SEE ALSO

v.build.polylines, *v.clean*, *v.digit*, *v.prune*, *v.spag*, *v.support*

AUTHOR

J.Soimasuo
Faculty of Forestry
University of Joensuu, Finland

Last changed: $Date: 2003/02/05 11:06:51 $

NAME

v.rmdup − Remove duplicate items GRASS vector file. (GRASS Vector Program)

SYNOPSIS

v.rmdup
v.rmdup help
v.rmdup input=*name* **output**=*name* [**threshold**=*value*]

DESCRIPTION

v.rmdup removes duplicate lines, areas, and points from a binary vector file (input). Duplicate may be defined as exact (less than 1.12e−16 for IEEE double precision), or defined with a threshold.

Parameters:

input=*name*
Name of a binary vector file
output=*name*
Name given to binary vector output file.
threshold=*value*
Threshold for defining duplicate lines, areas, and points.

The search technique used by *v.rmdup* is not extremely sophisticated and may take a long time for large files. Instead, only two sets of lines, areas, or points are active at any one time (so it is conservative with memory).

SEE ALSO

v.clean, *v.digit*, *v.prune*, *v.spag*, *v.support*

AUTHOR

James Darrell McCauley, Purdue University
GRASS 5 fixes by Roger S. Miller

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

NAME

v.rmedge − Selects edge vectors from an existing vector map, removes them, and creates a new vector map.
"(SCS GRASS Vector Program)

SYNOPSIS

v.rmedge
v.rmedge help
v.rmedge input=*name* **output=***name*

DESCRIPTION

allows a user to create a new vector map from an existing vector map, however *ALL OUTER* boundaries will be gone. Any outer edge that needs to be retained will require adding another outside boundary, *v.digit* can be used to provide this additional boundary.

Parameters:

input*=name*
Name of vector input file.
output*=name*
Name of vector output file.

NOTES

When using *v.digit* to add an additional boundary to a map, it may be necessary to break the boundary of an existing area. The user **** MUST REMEMBER **** that breaking the boundary of a named area **** REMOVES THE LABEL ****, and the area **** MUST BE RE−LABELED **** prior to leaving *v.digit*;
−OR−
**** v.rmedge will REMOVE that area boundary ALSO ****

This will result in MISSING data in a patching operation.

SEE ALSO

v.digit
v.merge
v.patch

AUTHOR

R.L. Glenn, USDA, SCS, NHQ−CGIS

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

NAME

v.scale.random − Creates a site_lists file of randomly placed symbols within a GRASS vector area.
(SCS GRASS Vector Program)

SYNOPSIS

v.scale.random
v.scale.random help
v.scale.random [**−n**] **map**=*name* **site**=*name* **dot**=*name* [**outdot**=*name*] **scale**=*value* [**cover**=*value*] [**size**=*value*]

DESCRIPTION

v.scale.random allows a user to create a GRASS site_lists file containing sites randomly placed within an area. This program is designed as an interface to *v.random* to aid the user in determining the number of dots to locate. This progam uses the same type of file listing the category and counts. This file is then modified to try to produce a pleasing dot count for a map of a specified scale.

COMMAND LINE OPTIONS

Flags:

−n
Use category values, NOT category names.

Parameters:

map=*name*
Input vector file name.
site=*name*
Output site_lists file name.
dot=*name*
Name of file containing labels and counts.
outdot=*name*
Name of new file to contain scaled counts.
scale=*value*
The denominator of the map scale.
Options: 1 − 999999999999
cover=*value*
The fraction of the most dense area to cover with dots.
Options: 0 − 1

size=*value*
The size of each dot, in centimeters.
Options: 0 − 100

SEE ALSO

s.menu
s.random
v.random

AUTHOR

M.L. Holko, USDA, SCS, NHQ−CGIS

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

NAME

v.sdts.dq.cp − installs SDTS data quality reports.
(GRASS Vector Data Export/Processing Program)

SYNOPSIS

v.sdts.dq.cp
v.sdts.dq.cp help
v.sdts.dq.cp [**−f**] **map**=*name* [**HL**=*name*] [**PA**=*name*] [**AA**=*name*] [**LC**=*name*] [**CG**=*name*]

DESCRIPTION

The program provides assistance for the preparation of the five data quality report modules (Lineage, Positional Accuracy, Attribute Accuracy, Logical Consistency, and Completeness) required in an SDTS transfer dataset. The program has one simple function: the user specifies a map layer in his current mapset, and then one or more files to be used for SDTS data quality reports for this map layer. The program copies the specified files to a standard location (in the user's current mapset, under the *dig_misc* directory). Later, when *v.out.sdts* is run for the same map layer, the data quality reports will be incorporated into an SDTS export dataset. .sp

COMMAND LINE OPTIONS

Flags:

−f
force overwriting of pre−exisiting data quality report(s).

Parameters:

map=*name*
name of vector map layer to which the specified data quality report files apply.
HL=*name*
name of file to be used for the SDTS Lineage (HL) data quality report.
PA=*name*
name of file to be used for the SDTS Positional Accuracy (PA) data quality report.
AA=*name*
name of file to be used for the SDTS Attribute Accuracy (AA) data quality report.
LC=*name*
name of file to be used for the SDTS Logical Consistency (LC) data quality report.
CG=*name*
name of file to be used for the SDTS Completeness (CG) data quality report.

NOTES

Data Quality report files should be simple narrative text files. After the files have been installed with *v.sdts.dq.cp*, *v.out.sdts* will convert the installed copy of each report to SDTS ISO 8211 format. Each paragraph in the original file will become a separate record in the SDTS data quality module.

Parameter names−−HL, PA, AA, LC, CG−−are SDTS codes for different data quality modules (HL=Lineage, PA=Positional Accuracy, etc.).

Data quality files to be installed can be created as well as installed with *v.sdts.meta*.

SEE ALSO

m.sdts.read, *v.in.sdts*, *v.out.sdts*, *v.sdts.dp.cp*, *v.sdts.meta.cp*, *v.sdts.meta*

AUTHORS

David Stigberg, U.S.Army Construction Engineering Research Laboratory
Tin Qian, University of Illinois

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

NAME

v.sdts.meta.cp − installs supplementary metadata file preparatory to creation of an SDTS export dataset. *(GRASS Vector Data Export/Processing Program)*

SYNOPSIS

v.sdts.meta.cp
v.sdts.meta.cp help
v.sdts.meta.cp [**−f**] **metafile**=*name* **map**=**name**

DESCRIPTION

The program provides assistance for the preparation of supplemental metadata for an SDTS export dataset. The user specifies a map layer in his current mapset, and then a file of metadata information pertaining to the named map. The program copies the specified metadata file to a standard location (in the user's mapset, under the *dig_misc* directory). Later, when *v.out.sdts* is run for the same map layer, the items in the metadata file will be incorporated in various places in the export dataset.

While the metadata file can be prepared by the user, it's format and contents are strictly defined. An alternative is to use the interface program, *v.sdts.meta*, with which the user can prepare and install a correctly formatted metadata file.

COMMAND LINE OPTIONS

Flags:

−f
force overwriting of pre−exisiting metadata file.

Parameters:

map=*name*
name of vector map layer to which the specified metadata file applies.
metafile=*name*
name of file to be installed as a metadata file for the specified map.

NOTES

The format of the metadata source file is rather highly specified. Following is a list of the items that can be included in the file. Note that each item is preceded by a particular code. The code, and the following ':' must

be entered intact for each included metadata item. The colon is then immediately followed by the user−supplied information:

IDEN_MPDT:(creation date of original source map; YYYY or YYMMDD format)

IDEN_TITL:(general TITLE for contents of transfer, for TITL field in IDEN module. If not specified, vector header "map_name" will be used)

IDEN_COMT:(general comment on transfer, for IDEN module's COMT field)

XREF_HDAT:(name of geodetic datum to which export data are referenced, for HDAT fieid in XREF module.)

DDSH_ENT_NAME:(for Dictionary/Schema module; name of kind of entity that *dig_att* and *dig_cats* values represent. If not specified, map name will be used.)

DDDF_GRASS_ENT:(definition for entity in "DDSH_ENT_NAME", for Dictionary/ Definition module. if not supplied, simple default is used.)

DDDF_ATTR_NUM:(definition for *dig_att* values, for Dictionary/Definition module; if not specified, simple default is used.)

DDDF_ATTR_LABEL:(definition for *dig_cats* values, for Dictionary/Definition module; if not specified, simple default is used.)

As noted above, the metadata file can be prepared, and installed, without the user having to worry about format details, with *v.sdts.meta*.

SEE ALSO

m.sdts.read, *v.in.sdts*, *v.out.sdts*, *v.sdts.dp.cp*, *v.sdts.meta.cp*, *v.sdts.meta*

AUTHORS

David Stigberg, U.S.Army Construction Engineering Research Laboratory
Tin Qian, University of Illinois

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

NAME

v.sdts.meta − Interactive menu−driven utility to create and install supplementary metadata and data quality reports for a vector map, preparatory to their incorporation in an SDTS transfer dataset.
(GRASS Vector Data Export/Processing Program)

SYNOPSIS

v.sdts.meta

DESCRIPTION

This menu−driven Tcl/Tk program enables the user to prepare and install supplementary metadata and data quality reports for a vector map preceding the creation of an SDTS transfer dataset. With v.*sdts.meta*, metadata and data quality files can be edited, saved, retrieved, and modified, and then installed in the GRASS database in association with a particular vector map layer. Subsequently, *v.out.sdts* will incorporate this metadata along with the associated vector map in an SDTS transfer dataset.

In addition to the five data quality modules, supplementary metadata items currently able to be edited and installed include

(1) creation date for the original source map;
(2) a general TITLE for the transfer;
(3) a general comment about the transfer;
(4) name of geographic datum;
(4) and (5), name and definition for the kind of entity contained in the GRASS vector layer being transferred;
(6) and (7) custom definitions for the *dig_att* and *dig_cats* values being transferred.

SEE ALSO

m.sdts.read, *v.in.sdts*, *v.out.sdts*, *v.sdts.dp.cp*, *v.sdts.meta.cp*, *v.sdts.meta*

AUTHORS

David Stigberg, U.S.Army Construction Engineering Research Laboratory
Tin Qian, University of Illinois

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

NAME

v.spag − Process spaghetti−digitized binary vector file.
(GRASS Vector Program)

SYNOPSIS

v.spag
v.spag help
v.spag [**−isp**] **map=***name* [**threshold=***value*]

DESCRIPTION

This program will fix vector data that were not digitized in correct GRASS vector format. It will create a node at every line crossing, and will delete all hanging lines of length less than the specified threshold.

OPTIONS

Flag:

−i
: Only do identical line removal pass.

−s
: Snap nodes.

−p
: Prompt user for threshold value.

Parameters:

map=*name*
: Name of the binary vector file to be fixed.

threshold=*value*
: Node−snapping threshold value.

NOTES

The user must run *v.support* after running *v.spag* to correct the topology (*dig_plus*) file.

v.spag generally deletes many lines from the input vector map layer. Because deleted lines are not eliminated from a vector data (*dig*) file, but are only marked as deleted, the user will probably wish to run *v.clean* on the vector file after running *v.spag* to eliminate any deleted lines from the *dig* file and decrease its size.

BUGS

This program contains several known bugs. It will sometimes get stuck in a loop in which it creates the same line over and over again. It will occasionally create an incorrect line. Users are strongly urged to back−up their vector data files before running this alpha version of *v.spag*.

Update Note:
1/93: For GRASS Release 4.1, the **−i** flag has been added and bugs fixed. It works better than the 4.0 release.
6/00: The **−s** and **−p** flags has been added. One reason for loop fixed, please report if you get into the loop after this date.

SEE ALSO

v.build.polylines, *v.clean*, *v.digit*, *v.prune*, *v.rmdup*, *v.support*

AUTHOR

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

Last changed: $Date: 2003/08/20 08:10:13 $

NAME

v.split − Split lines to line segments.
(GRASS Vector Program)

SYNOPSIS

v.split
v.split help
v.split input=*name* **output**=*name* [**type**=*type*]

DESCRIPTION

v.split splits lines to line segments. Attributes are not written for each new segment (dig_att file is only copy of the input dig_att file).

Parameters:

input=*name*
Name of an existing vector map.
output=*name*
Name of a new vector file.
type=*type*
Select line or area boundary.
Options: line, boundary (multiple allowed)
Default: line

NOTES

v.support must also be run on the resultant vector **output** file to build the needed topology information stored under the user's *dig_plus* directory.

SEE ALSO

v.support
parser

AUTHOR

Radim Blazek

NAME

v.stats − Prints information about a binary GRASS vector map layer.
(GRASS Vector Program)

SYNOPSIS

v.stats
v.stats help
v.stats [**−h**] **map**=*name*

DESCRIPTION

The program *v.stats* will print to standard output information about a user−specified binary GRASS vector map layer.

Flag:

−h
Display header information from the vector file.

Parameter:

map=*name*
Name of the binary GRASS vector file to be queried.

The program will be run non−interactively if the user specifies the parameter value and (optionally) sets the **−h** flag on the command line, using the form:

v.stats [**−h**] **map**=*name*

If the user instead simply types **v.stats** without specifying program arguments on the command line, the program will prompt the user to enter inputs through the standard user interface described in the manual entry for *parser*.

NOTES

Sample output follows:

Format: Version 4.0 (Level 2 access) (Portable)
Number of Lines: 3
Number of Nodes: 2

```
Number of Areas: 2 (complete)
Number of Isles: 1 (complete)
Number of Atts : 2
```

The GRASS version number is given. Parenthetical text following this describes the read access level available and notes whether or not the file is in GRASS version 4.0 portable data format. The access level indicates the types of data available for the named vector map layer. "Level 1" denotes a binary vector file (without any accompanying *dig_plus* file), while "Level 2" denotes the existence of a *dig_plus* (vector topology) file for the named map layer (generally created by *v.support*). If only Level 1 information is available for a vector map layer, only the number of lines and (optionally) the file header will be printed to standard output.

If all areas and islands (isles) in the vector file have been identified (usually by *v.support*), their counts will be followed by "*complete*." If not, they will be followed by "*incomplete*". The *dig_plus* file, which is created and updated by *v.support*, must exist for this information to be output.

SEE ALSO

v.digit, *v.import*, *v.support*, and *parser*

AUTHOR

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

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

NAME

v.support − Creates GRASS support files for (binary) GRASS vector data.
(GRASS Vector Program)

SYNOPSIS

v.support
v.support help
v.support [**−spr**] **map**=*name* [**option**=*name*] [**threshold**=*value*] [**err**=*name*]

DESCRIPTION

v.support builds GRASS support files for (binary) GRASS vector data files. These support files supply topology (**`dig_plus`**) and category (**`dig_att`**) information that are needed by other GRASS programs (e.g., by *v.digit*, *v.to.rast*, etc.).

The program gives the user the options of:

1. building topological information (the **`dig_plus`** file) needed by *v.digit* for the vector data file, and/or
2. editing the category label (**`dig_cats`**) file associated with the vector data file.

OPTIONS

Program parameters and flags have the following meanings.

Flags:

−s
: Snap nodes. Valid only with *build* option.

−p
: Prompt user for threshold value. Valid only with *build* option. This flag is designed to be used only by the *v.import* program, and can usually be ignore.

−r
: Reset corners of map region from range of data values.

Parameters:

map=*name*
: Name of binary GRASS vector data file for which support files are to be created or modified. This map layer must exist in the user's current GRASS mapset.

option=*name*
: Build topology information (**dig_plus** file), or edit categories (**dig_att** file) for *map*.
Options: *build*, *edit*
Default: *edit*

threshold=*value*
: Snapping threshold value to be used when building topology. Specified value is always saved in file header. (Note: snapping is done only with *build* option and **−s** flag).

err=*value*
: Name of new vector binary file, where are saved all area edges with topology problems like free ends or not closed areas.

The user can run this program non−interactively by specifying parameter values (and optionally, flag settings) on the command line.

Alternately, the user can simply type the command **v.support** on the command line. In this event, the program will prompt the user to enter parameter values and flag settings.

If the *build* option is chosen, the user may further specify the **−s** flag, to snap nodes in the vector file. If nodes are to be snapped, the user can either: (1) use the calculated default threshold (based on the scale of vector data); (2) enter the −p flag, causing the program to prompt the user for a snapping threshold value; or (3) enter a specific threshold value on the command line.

The spatial assignment of category values (found in the **dig_att** file) is also performed during building of file topology.

The *edit* option allows the user to edit the category labels to be associated with the category values attached to the vector data during topology building. These labels, if created, are then used for raster map layers derived from their vector counterparts. The labels are placed in the **dig_cats** directory.

NOTES

A **dig_plus** file must be created for each imported vector map before it can be used in post−GRASS 3.0 versions of *digit* (now referred to as *v.digit*).

Topological information for GRASS vector files can also be built using *option 4* of the *v.import* program.

This program will convert pre−4.0 version GRASS vector files to 4.0 format.

v.support creates support files only for binary vector files located in the user's current mapset.

SEE ALSO

v.build.polylines, *v.digit*, *v.import*, *v.in.ascii*, *v.prune*, *v.to.rast*

AUTHORS

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

Last changed: $Date: 2002/08/28 17:06:34 $

NAME

v.surf.rst − interpolation and topographic analysis from given contour data in vector format to GRASS floating point raster format using regularized spline with tension
(*GRASS Vector Program*)

SYNOPSIS

v.surf.rst
v.surf.rst help
v.surf.rst [−d] [−r] [−c] [−t] input = name **elev** = name [**slope** = name] [**aspect** = name] [**pcurv** = name] [**tcurv** = name] [**mcurv** = name] [**maskmap** = name] [**dmin** = val] [**dmax** = val] [**zmult =** val] [**tension** = val] [**smooth** = val] [**segmax =** val] [**npmin** = val] [**theta** = val] [**scalex** = val] [**devi** = name] [**treefile** = name] [= name]

DESCRIPTION

v.surf.rst
This program interpolates the z−values from vector data (e.g., contours, isolines) given in a vector file named *input* to grid cells in the output raster file *elev* representing a surface. As an option, simultaneously with interpolation, topographic parameters slope, aspect, profile curvature (measured in the direction of steepest slope), tangential curvature (measured in the direction of a tangent to contour line) or mean curvature are computed and saved as raster files specified by the options *slope, aspect, pcurv, tcurv, mcurv* respectively. If ***−d*** flag is set the program outputs partial derivatives f_x,f_y,f_{xx},f_{yy},f_{xy} instead of slope, aspect, profile, tangential and mean curvatures respectively. If the input data have time stamp, the program creates time stamp for all output files.

User can define a raster file named *maskmap*, which will be used as a mask. The interpolation is skipped for cells which have zero or NULL value in mask. NULL values will be assigned to these cells in all output raster files. Data points are checked for identical points and points that are closer to each other than the given *dmin* are removed. Additional points are used for interpolation between each 2 points on a line if the distance between them is greater than specified *dmax*. Parameter *zmult* allows user to rescale the z−values (useful e.g. for transformation of elevations given in feet to meters, so that the proper values of slopes and curvatures can be computed).

Regularized spline with tension is used for the interpolation. The *tension* parameter tunes the character of the resulting surface from thin plate to membrane. For noisy data, it is possible to define a smoothing parameter *smooth.* With the smoothing parameter set to zero (*smooth=0*) the resulting surface passes exactly through the data points. When smoothing parameter is used, it is possible to output site file *devi* containing deviations of the resulting surface from the given data.

If the number of given points is greater than *segmax*, segmented processing is used . The region is split into rectangular segments, each having less than *segmax* points and interpolation is performed on each segment of the region. To ensure the smooth connection of segments the interpolation function for each segment is computed using the points in given segment and the points in its neighborhood which are in the rectangular window surrounding the given segment. The number of points taken for interpolation is controlled by *npmin*, the value of which must be larger than *segmax* . User can choose to output vector files *treefile* and *overfile* which represent the quad tree used for segmentation and overlapping neighborhoods from which additional points for interpolation on each segment were taken. Anisotropic surfaces can be interpolated setting anisotropy angle *theta* and scaling factor *scalex*. The program writes several important values to history file of raster map ***elev***.

OPTIONS

The user can run this program either interactively or non−interactively. The program will be run non−interactively if the user specifies program arguments and flag settings on the command line using the form:

v.surf.rst [−d] [−r] [−c] [−t] input = name **elev** = name [**slope** = name] [**aspect** = name] [**pcurv** = name] [**tcurv** = name] [**mcurv** = name] [**maskmap** = name] [**dmin** = val] [**dmax** = val] [**zmult =** val] [**tension** = val] [**smooth** = val] [**segmax =** val] [**npmin** = val] [**theta** = val] [**scalex** = val] [**devi** = name] [**treefile** = name] [= name]

Alternately, the user can simply type v**.surf.rst** on the command line without program arguments. In this case, the user will be prompted for parameter values and flag settings using the standard GRASS parser interface described in the manual entry for *parser*.

Flags:

−d Output partial derivatives instead of aspect, slope and curvatures.

−t Use dnorm independent tension

−r Zero values in input file represent elevation.

−c Category data is used instead of attribute as an elevation

Parameters

input = *name*
Use the existing vector file name as input.

elev = *name*
Output elevation values to raster file *name*.

slope = *name*
Output slope or dx values to raster file *name*.

aspect = *name*
Output aspect or dy values to raster file *name*.

pcurv = *name*
Output profile curvature or dxx values to raster file *name*.

tcurv = *name*
Output tangential curvature or dyy values to raster file *name*.

mcurv = *name*
Output mean curvature or dxy values to raster file *name*.

maskmap = *name*
Use the existing raster file *name* as a mask.

dmin = *val*
Set min distance between points to *val*. Default value is set to 0.5 grid cell size.

dmax = *val*
Maximum distance between points. Default value is 5 * **dmin.**

zmult = *val*
Convert z−values using conversion factor *val*. Default value is 1.

tension =*val*
Set tension to *val*. Default value is 40, appropriate for smooth surfaces.

smooth = *val*
Set smoothing parameter to *val*. Default value is 0.1.

segmax = *val*
Set max number of points per segment to *val*. Default value is 40.

npmin = *val*
Set min number of points for interpolation to *val*. Default value is 200, for data with heterogeneous spatial distribution higher value is suggested (see notes).

theta = *val*
Set anisotropy angle in degrees (measured from East counterclockwise) to *val*.

scalex = *val*
Set anisotropy scaling factor to *val*. Values 0 and 1 give no anisotropy.

devi = *name*
Output deviations to a site file *name*.

treefile = *name*
Output quad tree used for segmentation to vector file *name*

overfile = *name*
Output overlapping neighborhoods used for segmentation to vector file *name*.

NOTES

v.surf.rst uses regularized spline with tensionfor interpolation from vector data. Additional points are used for interpolation between each 2 points on a line if the distance between them is greater than specified *dmax*. If *dmax* is small (less than cell size) the number of added data points can be vary large and slow down interpolation significantly. The implementation has a segmentation procedure based on quadtrees which enhances the efficiency for large data sets.Special color tables are created by the program for output raster files.

Topographic parameters are computed directly from the interpolation function so that the important relationships between these parameters are preserved. The equations for computation of these parameters and their interpretation is described in (Mitasova and Hofierka 1993). Slopes and aspect are computed in degrees (0−90 and 1−360 respectively). The aspect raster file has value 0 assigned to flat areas (with slope less than 0.1%) and to singular points with undefined aspect. Aspect points downslope and is 90 to the North, 180 to the West, 270 to the South and 360 to the East, the values increase counterclockwise. Curvatures are positive for convex and negative for concave areas. Singular points with undefined curvatures have assigned zero values.

Tension and *smooth*ing allow user to tune the surface character. For most landscape scale applications the default should work fine.The program gives warning when significant overshoots appear in the resulting surface and higher tension or smoothing should be used.
While it is possible to automatize the selection of suitable *tension* and *smooth*ing, it has not been done yet, so here are some hints which may help to choose the proper parameters if the results look "weird". It is useful to know that the method is scale dependent and the *tension* works as a rescaling parameter (high *tension* "increases the distances between the points" and reduces the range of impact of each point, low *tension* "decreases the distance" and the points influence each other over longer range). Surface with *tension* set too high behaves like a membrane (rubber sheet stretched over the data points) with peak or pit ("crater") in each given point and everywhere else the surface goes rapidly to trend. If digitized contours are used as input data, high tension can cause artificial waves along contours. Lower tension and higher smoothing is suggested for such a case.
Surface with *tension* set too low behaves like a stiff steel plate and overshoots can appear in areas with rapid change of gradient and segmentation can be visible. Increase tension should solve the problems.
There are two options how *tension* can be applied in relation to *dnorm* (dnorm rescales the coordinates depending on the average data density so that the size of segments with *segmax*=40 points is around 1 − this ensures the numerical stability of the computation):

1. Default (used also in s.surf.rst): the given *tension* is applied to normalized data (x/*dnorm*..), that means that the distances are multiplied (rescaled) by *tension/dnorm*. If density of points is changed, e.g., by using higher *dmin*, the *dnorm* changes and *tension* needs to be changed too to get the same result. Because the *tension* is applied to normalized data its suitable value is usually within the 10−100 range and does not depend on the actual scale (distances) of the original data (which can be km for regional applications or cm for field experiments).
2. Flag **−t** (experimental for s.surf.rst)**:** The given *tension* is applied to un−normalized data (rescaled tension = t*ension***dnorm*/1000 is applied to normalized data (x/*dnorm*) and therefore *dnorm* cancels out) so here *tension* truly works as a rescaling parameter. For regional applications with distances between points in km. the suitable tension can be 500 or higher, for detailed field scale analysis it can be 0.1. To help select how much the data need to be rescaled the program writes *dnorm* and rescaled tension=*tension***dnorm*/1000 at the beginning of the program run. This rescaled *tension* should be around 20−30. If it is lower or higher, the given *tension* parameter should be changed accordingly.

The default is a recommended choice, however for the applications where the user needs to change density of data and preserve the interpolation character the **−t** flag can be helpful.

Anisotropic data (e.g. geologic phenomena) can be interpolated using *theta* and *scalex* defining orientation and ratio of the perpendicular axes put on the longest/shortest side of the feature, respectively. *Theta* is measured in degrees from East, counterclockwise. *Scalex* is a ratio of axes sizes. Setting *scalex* in the range 0−1, you will get pattern prolonged in the direction defined by *theta*. *Scalex* value 0.5 means that your feature is approximatelly 2 times longer in the direction of *theta* than in the perpendicular direction. *Scalex* value 2 means that axes ratio is reverse and you will get pattern perpendicular to the previous example. Please note that anisotropy option has not been extensively tested and may include bugs − if there are problems, please report to GRASS bugtracker (accessible from http://grass.itc.it/).

The program gives warning when significant overshoots appear and higher tension should be used. However, with tension too high the resulting surface changes its behavior to membrane (rubber sheet stretched over the data points resulting in a peak or pit in each given point and everywhere else the surface goes rapidly to trend). Also smoothing can be used to reduce the overshoots.

For data with values changing over several magnitudes (sometimes the concentration or density data) it is suggested to interpolate the log of the values rather than the original ones.

The program checks the numerical stability of the algorithm by computating the values in given points, and prints the root mean square deviation (rms) found into the history file of raster map *elev*. For computation with smoothing set to 0. rms should be 0. Significant increase in *tension* is suggested if the rms is unexpectedly high for this case. With smoothing parameter greater than zero the surface will not pass exactly through the data points and the higher the parameter the closer the surface will be to the trend. The rms then represents a measure of smoothing effect on data. More detailed analysis of smoothing effects can be performed using the output deviations option.

The program writes the values of parameters used in computation into the comment part of history file *elev* as well as the following values which help to evaluate the results and choose the suitable parameters: minimum and maximum z values in the data file (zmin_data, zmax_data) and in the interpolated raster map (zmin_int, zmax_int), rescaling parameter used for normalization (dnorm), which influences the tension.

If visible connection of segments appears, the program should be rerun with higher *npmin* to get more points from the neighborhood of given segment and/or with higher tension.

When the number of points in a site file is not too large (less than 800), the user can skip segmentation by setting *segmax* to the number of data points or *segmax=700*.

The program gives warning when user wants to interpolate outside the rectangle given by minimum and maximum coordinates in the vector file, zoom into the area where the given data are is suggested in this case.

When a mask is used, the program takes all points in the given region for interpolation, including those in the area which is masked out, to ensure proper interpolation along the border of the mask. It therefore does not mask out the data points, if this is desirable, it must be done outside *v.surf.rst.*

For examples of applications see http://skagit.meas.ncsu.edu/~helena/gmslab/.

The user must run g.region before the program to set the region and resolution for interpolation.

SEE ALSO

s.surf.rst

AUTHORS

Original version of program (in FORTRAN) and GRASS enhancements:
Lubos Mitas, NCSA, University of Illinois at Urbana Champaign, Illinois, USA
Helena Mitasova, Department of Geography, University of Illinois at Urbana−Champaign, USA

Modified program (translated to C, adapted for GRASS, new segmentation procedure):
Irina Kosinovsky, US Army CERL
Dave Gerdes, US Army CERL

Modifications for new sites format and timestamping:
Darrel McCauley, Purdue University

REFERENCES

Hofierka J., Parajka J., Mitasova H., Mitas L., 2002, Multivariate Interpolation of Precipitation Using Regularized Spline with Tension. *Transactions in GIS* 6(2), pp. 135−150.

H. Mitasova, L. Mitas, B.M. Brown, D.P. Gerdes, I. Kosinovsky, 1995, Modeling spatially and temporally distributed phenomena: New methods and tools for GRASS GIS. International Journal of GIS, 9 (4), special issue on Integrating GIS and Environmental modeling, 433−446.

Mitasova and Mitas 1993: Interpolation by Regularized Spline with Tension: I. Theory and Implementation, Mathematical Geology ,25, 641−655.

Mitasova and Hofierka 1993: Interpolation by Regularized Spline with Tension: II. Application to Terrain Modeling and Surface Geometry Analysis, Mathematical Geology 25, 657−667.

Mitas, L., Mitasova H., 1988 : General variational approach to the interpolation problem, Computers and Mathematics with Applications, v.16, p. 983

Talmi, A. and Gilat, G., 1977 : Method for Smooth Approximation of Data, Journal of Computational Physics, 23, p.93−123.

Wahba, G., 1990, : Spline Models for Observational Data, CNMS−NSF Regional Conference series in applied mathematics, 59, SIAM, Philadelphia, Pennsylvania.

Last changed: $Date: 2003/02/05 12:46:04 $

NAME

v.timestamp print/add/remove a timestamp for a vector map
(GRASS Vector Module)

SYNOPSIS

v.timestamp
v.timestamp help
v.timestamp map=*name* [**date=***timestamp*[*/timestamp*]]

DESCRIPTION

This command has 2 modes of operation. If no date argument is supplied, then the current timestamp for the vector map is printed. If a date argument is specified, then the timestamp for the vector map is set to the specified date(s). See EXAMPLES below.

OPTIONS

Parameters:

map=*name*
Name of vector map.
date=*timestamp[/timestamp] or none*
String specifying timestamp, range, or *none*. See FORMAT below.

NOTES

Strings containing spaces should be quoted. For specifying a range of time, the two timestamps should be separated by a forward slash. To remove the timestamp from a vector map, use **date=***none*.

EXAMPLES

v.timestamp map=soils
Prints the timestamp for the "soils" vector map. If there is no timestamp for soils, nothing is printed. If there is a timestamp, one or two time strings are printed, depending on if the timestamp for the map consists of a single date or two dates (ie start and end dates).

v.timestamp map=soils date='15 sep 1987'
Sets the timestamp for "soils" to the single date
"15 sep 1987"

v.timestamp map=soils date='15 sep 1987/20 feb 1988'
Sets the timestamp for "soils" to have the start date
"15 sep 1987" and the end date "20 feb 1988"

v.timestamp map=soils date=none
Removes the timestamp for the "soils" vector map

TIMESTAMP FORMAT

The timestamp values must use the format as described in the GRASS datetime library. The source tree for this library should have a description of the format. For convience, the formats as of Feb, 1996 are reproduced here:

There are two types of datetime values: absolute and relative. Absolute values specify exact dates and/or times. Relative values specify a span of time. Some examples will help clarify:

Absolute

The general format for absolute values is:

```
day month year [bc] hour:minute:seconds timezone
```

day is 1−31
month is jan,feb,...,dec
year is 4 digit year
[bc] if present, indicates dates is BC
hour is 0−23 (24 hour clock)
mintue is 0−59
second is 0−59.9999 (fractions of second allowed)
timezone is +hhmm or −hhmm (eg, −0600)

parts can be missing

1994 [bc]
Jan 1994 [bc]
15 jan 1000 [bc]
15 jan 1994 [bc] 10 [+0000]
15 jan 1994 [bc] 10:00 [+0100]
15 jan 1994 [bc] 10:00:23.34 [−0500]

Relative

There are two types of relative datetime values, year− month and day−second. The formats are:

```
[−] # years # months
[−] # days # hours # minutes # seconds
```

The words years, months, days, hours, minutes, seconds are literal words, and the # are the numeric values. Examples:

```
2 years
5 months
2 years 5 months
100 days
15 hours 25 minutes 35.34 seconds
100 days 25 minutes
1000 hours 35.34 seconds
```

The following are *illegal* because it mixes year−month and day−second (because the number of days in a month or in a year vary):

```
3 months 15 days
3 years 10 days
```

SEE ALSO

r.timestamp

BUGS

Spaces in the timestamp value are required.

AUTHOR

Markus Neteler, Institute of Physical Geography and Landscape Ecology, University of Hannover
based on r.timestamp by Michael Shapiro,
U.S.Army Construction Engineering Research Laboratory

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

NAME

v.to.db − load values from vector to database
(GRASS Vector Program)

SYNOPSIS

v.to.db [−ps] map=*name* **type**=*name[,name,...]* **option**=*name[,name,...]* [**units**=*name*] **table**=*name* [**key**=*name*] [**col1**=*name*] [**col2**=*name*]

Flags:

−p
Print only

−s
Print only sql statements

Parameters:

map
Vector map

type
Type of elements
Options: area, line, site

option
What load
Options: cat, label, area, length, count, coor

units
Units data are in. Either: mi(les), f(eet), me(ters), k(ilometers), a(cres), h(ectacres)
Options: mi (miles), f (feet), me (meters), k (kilometers), a (acres), h (hectacres)

table
DB table

key
Key column

col1
Column 1

col2
Column 2

EXAMPLE

Uploading of the attributes of a vector map to the DBMS has to be done column−wise:

```
v.to.db map=soils type=area option=cat table=soils key=cat

v.to.db map=soils type=area option=label table=soils key=cat
col1=soiltype
```

AUTHOR

Radim Blazek

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

NAME

v.to.gnuplot − outputs an ASCII vector map in GNUPLOT format
(GRASS Shell Script)

SYNOPSIS

v.to.gnuplot help
v.to.gnuplot *name*

DESCRIPTION

v.to.gnuplot is an *awk* shell script that converts an ASCII vector map into a format suitable for plotting with *gnuplot* and writes the results to standard output.

OPTIONS

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

Parameter:

name
Full path of an ASCII vector map layer.

EXAMPLE

Typing the following at the command line:

v.to.gnuplot < LOCATION/dig_ascii/elevation > elev.dat

will write the raster data to *elev.dat*. After staring the GRASS graphics monitor, the following dialogue:

gnuplot
gnuplot> **plot 'elev.dat' noTITLE with lines**

will plot a map of *elevation*.

NOTES

Output may be saved as PostScript, FrameMaker, TeX, etc (approximately 2 dozen output formats).

v.clean and *v.out.ascii*, must be run prior to *v.to.gnuplot*.

FILES

$GISBASE/scripts/v.to.gnuplot

SEE ALSO

v.clean, *v.out.ascii*, *r.to.gnuplot gnuplot*

AUTHOR

James Darrell McCauley, Agricultural Engineering, Purdue University

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

NAME

v.to.pg − Export areas and lines from an existing vector map to Postgres/PostGIS table. (GRASS−RDBMS Interface Display Program)

SYNOPSIS

v.to.pg
v.to.pg help
v.to.pg [−fvtp] key=*name* **tab**=*name***type**=*name* [**where**=*name*] **map**=*name* [**color**=*name*]

ALTERNATE

v.to.pg −s help
v.to.pg]−sfvp] sql=*filename* **map**=*name* **type**=*name* [**color**=*name*]

DESCRIPTION

v.to.pg exports vectors from an existing vector map based on the unique values in a database column (or optionally all vectors without referencing to any existing table, see −t option). Each row returned by a user constructed database query will be associated with a vector feature which may be drawn on the graphics display if X−windows are there. The user can control the color of the vector draw by specifying a color on the command line.

As result, a new Postgres table *table_bnd* or *table_arc* (or *table_mpoly* and *table_mstring* for PostGIS) is created to hold areas as internal type "polygon" and lines as "open path", where *table* is the **tab** parameter. Besides these elements in fields called *boundary* or *segment*, the table would also have the category field (named after **key** parameter), case number this category occurred in map (as field called 'num'), and an extra boolean field 'ex' for 'true' if the polygon is external and 'f' if it is a hole for *table_bnd* only.

For PostGIS, the difference is that types of the imported entities are POLYGON (i.e., with "holes" all in one POLYGON) and LINESTRING, respectively. The fields are called *grass_poly* and *grass_line*. There is no need to define whether a polygon is internal ("hole") or external, in this case, therefore there is no field 'ex'.

COMMAND LINE OPTIONS

Flag:

−f
Fill polygons selected on the query criteria.

−t

Export all map vectors without reference to existing table ('where' clause ignored).

−v

Verbose mode with statistics on the completion of the insertions.

−p

Create and populate PostGIS GEOMETRY format table instead of normal Postgres geometry types.

Parameters:

key=databasecolumnname

Column in table "tab" of the currently selected database containing values corresponding to the vector maps category values. Table is designated on the command line by tab=tablename and vector is given on the command line by map=mapname.

tab=databasetablename

Table in the currently selected database containing a column which has values corresponding to vector category values in the map designated by map=map.

where=SQLwhereclause

SQL "where" clause which specifies the query criteria to be used in subsetting the database. The field names specified in the where option must indicate the column(s) to be used, the operators to be used in the evaluation and the values which the data in the column will be evaluated against.

If the database column used as the selection criteria is a character field then the associated value must be placed in quotes. To determine the data types associated with columns in the currently selected database use the *g.column.pg* command with the *−v* flag.

Queries which are more complex are best implemented using the *−s* flag and a prepared SQL file.

map=map

Name of an existing vector map with category values which correspond to values in a specified column in the currently selected database.

color=name

Color to draw vectors in

Colors: red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, white, black

type=area,line

Type of the coverage to export.

ALTERNATE COMMAND LINE USAGE

The alternate command line usage is provided to simplify the process of retrieving information from more than one table in the query criteria. The alternate command line structure is selected using the the *[−s]* flag on the command line. When using this flag the user must include the name of a text file on the command line as well. This file must include a complete, well formed SQL SELECT statement specifying the query criteria to retrieve values which correspond to category values in a GRASS data layer.

Flag:

−s

SQL select statements are input from a prepared file

−f

Fill polygons selected on the query criteria

−v

Verbose mode with statistics on the completion of the insertions.

−p

Create and populate PostGIS GEOMETRY format table instead of normal Postgres geometry types.

Parameters:

sql=filename

SQL statements specifying well formed selection criteria.

map=name

Name of an existing vector map

color=name

Color to draw vectors in

Colors: red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, white, black

type=area,line

Type of the coverage to export.

EXAMPLE:

1. *v.to.pg −f key=rec_id map=kuruma_id tab=info_kuruma type=area where='type_id >32 and type_id < 38' color=red*

Result: only polygons with type in 33−37 range would be inserted in table *info_kuruma_bnd* and displayed in red color.

2. *v.to.pg −v −s −f −p sql=oak.sql map=kuruma_id type=area*

oak.sql is:

select rec_id from info_kuruma where type_id > 32 and type_id <38;

Result: this command would do the same as in the first example, some information printed on screen. However, the result table would be in PostGIS format.

BUGS

none

NOTE

This program requires the Postgres database software.

The 'total' mode of import (i.e., without referencing to existing table *and* to categories count in map) leads to that field 'num' in result tables would be incoherent (simply counts vectors from beginning to end).

SEE ALSO

g.column.pg, *g.select.pg*, *g.stats.pg*, *g.table.pg*, *d.rast.pg*, *d.site.pg*, *d.what.r.pg*, *d.what.s.pg*, *d.what.v.pg*, *v.reclass.pg*

AUTHOR

Alex Shevlakov (sixote@yahoo.com)

NAME

v.to.rast − Converts a binary GRASS vector map layer into a GRASS raster map layer.
(GRASS Vector Program)

SYNOPSIS

v.to.rast
v.to.rast help
v.to.rast input=*name* **output=***name*

DESCRIPTION

v.to.rast transforms (binary) GRASS vector map layers into GRASS raster map layer format. Most GRASS analysis programs operate on raster data.

OPTIONS

Parameters:

input*=name*
Name of the binary vector map layer to be converted.
output*=name*
Name to be assigned to the raster map layer output.

The user can run the program non−interactively by specifying the names of a vector input file and raster output file on the command line, using the form:

v.to.rast input=*name* **output=***name*

If the user instead types simply **v.to.rast** on the command line, the program will prompt the user to enter these names.

NOTES

v.to.rast will only affect data in areas lying inside the boundaries of the current geographic region. Before running *v.to.rast*, the user should therefore ensure that the current geographic region is correctly set and that the region resolution is at the desired level; the program may otherwise create an empty raster map layer. An empty raster map layer will also be created if the vector map layer has not been assigned category/attribute labels (e.g., through use of the *v.digit* program).

The *v.to.rast* program creates two files: a raster map layer, and a history file. The GRASS program *r.support* must be run to create additional support files for the raster map.

Additional problems sometimes lead to the creation of empty raster map layers. Unfortunately, error messages explaining these phenomena do not yet exist.

SEE ALSO

g.region, *r.poly*, *r.line*, *r.support*, *v.digit*, *v.import*, *v.support*

AUTHORS

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

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

NAME

v.to.sites − Converts point data in a binary GRASS vector map layer into a GRASS *site_lists* file.
(GRASS Vector Program)

SYNOPSIS

v.to.sites
v.to.sites help
v.to.sites input=*name* **output**=*name*

DESCRIPTION

The *v.to.sites* program extracts site data from a GRASS vector map layer and stores output in a new GRASS **`site_lists`** file. The resulting sites map layer can be used with such programs as *d.sites*. Only site (point) features in the named vector map layer are extracted and placed into the resulting site list. Lines and areas in the vector file are ignored.

OPTIONS

The user can run the program non−interactively by specifying the names of an existing vector *input* map layer and a new site list file to be *output* on the command line. The program will be run interactively if the user types **v.to.sites** without arguments on the command line. In this case, the user will be prompted to enter parameter values through the standard user interface described in the manual entry for *parser*.

Parameters:

input=*name*
Name of an existing binary vector map layer from which site data are to be extracted.
output=*name*
Name to be assigned to the resultant *site_lists* file.

If any of the sites have been labeled in *v.digit* then the resultant site list will contain category information. If none of the sites are labeled, a binary (0/1) site list file will be produced.

SEE ALSO

d.sites, *s.db.rim*, *s.menu*, *v.db.rim*, *v.digit* and *parser*

AUTHOR

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

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

NAME

v.to.sites − Converts point data in a binary GRASS vector map layer into a GRASS *site_lists* file.
(GRASS Vector Program)

SYNOPSIS

v.to.sites
v.to.sites help
v.to.sites [−acCid] input=*name* **output=***name* [**dmax=***value*]

DESCRIPTION

The *v.to.sites* program extracts data from a GRASS vector map layer and stores output in a new GRASS *site_lists* file. If **−a** flag is selected, *v.to.sites* extracts all vertices from a vector file, if not selected, it extracts site (point) features only, ignoring lines and areas. If **−i** flag is selected then, for each line, if the distance between any two vertices on this line is greater that **dmax**, additional points are added to keep the distance withing **dmax** range. The resulting sites map layer can be used with such programs as *d.sites*.

The user can run the program non−interactively by specifying the names of an existing vector **input** map layer and a new site list file to be **output** on the command line. The program will be run interactively if the user types **v.to.sites** without arguments on the command line. In this case, the user will be prompted to enter parameter values through the standard user interface described in the manual entry for *parser*.

OPTIONS:

Flags:

−a
Outputs all vertices (instead of site data only) from vector file to site file

−c
The Category NUMERIC data is used instead of attribute as a site description

−C
The Category TEXT data is used instead of attribute as a site description

−i
Additional sites are added between each 2 points on a line if the distance between them is greater than specified **dmax**. (valid only when **−a** is used)

−d
Write attribute as double instead of cat.
If any of the sites have been labeled in *v.digit*, then the resultant site list will contain category information (or attribute in case **−a** was used but **−c** was not). If none of the sites are labeled, a binary (0/1) site list file will be produced.

Parameters:

input*=*name
Name of an existing binary vector map layer from which site data are to be extracted.
output*=*name
Name to be assigned to the resultant *site_lists* file.
dmax*=*value
Maximum distance between points (valid only when **−a** and **−i** are used)

NOTES

Vector lines must be labeled.

If the flag **−i** is selected, a spline function is used to generate additional points along the line. To avoid the generating of lots of points along contours, **dmax=** should be really large so that **v.to.sites** does not add any addtional sites to the data. On the other hand, if additional sites data are desired, **dmax=** should be smaller than the distances between existing data points.

SEE ALSO

d.sites, *v.digit* and *parser*

AUTHOR

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

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

NAME

v.transform − Transforms an binary vector map layer from one coordinate system into another coordinate system.
(GRASS Vector Program)

SYNOPSIS

v.transform
v.transform help
v.transform [**−y**] **input**=*name* **output**=*name* [**pointsfile**=*name*]

DESCRIPTION

This program has been used to import vector files that were in scanner or digitizer (x,y) coordinates and to transform these into UTM coordinates.

OPTIONS

Flag:

−y
Suppress the printing of residuals or other information to standard output.

Parameters:

input=*name*
Name of the binary vector map layer to be transformed.
output=*name*
Name to be assigned to the resultant (transformed) binary vector map layer.
pointsfile=*name*
Name of a file containing transformation points, whose format is given below. Give a full path name for this file or it will be assumed to be located in the user's current directory.

The user can run this program non−interactively by specifying parameter values (and optionally, the flag setting) on the command line.

If the user runs **v.transform** without specifying program arguments on the command line, the program will prompt the user for inputs. When the program prompts the user for two sets of transformation points, the first set of points entered by the user should be in the coordinate system of the input map, and the second set of points should represent the corresponding geographic points in the coordinate system into which the map will transformed. A user must enter 4 to 10 of each set of points for the transformation to work correctly.

After the user has entered both sets of points, the program will show the amount of error associated with the transformation of the given points as the *residual mean average* (RMS). (An acceptable RMS for a 1:24,000 UTM map would be 1.2 to 2.4 (meters).) It will then ask if the transformation RMS value is acceptable. After an RMS is accepted by the user, *v.transform* will transform the binary vector (*dig*) map and its associated attribute (*dig_att*) file into the requested coordinate system.

Remember to run *v.support* on the output map.

NOTES

When rectifying a map to another coordinate system using *v.transform*, the user should specify the coordinates of between 4 to 10 points, and state these both in the coordinate systems of the input and output maps. The two sets of coordinates can be input to *v.transform* interactively, or from a file specified on the command line with the *pointsfile* option. The *pointsfile* option is especially useful when several maps in the same geographic area require transformation, as it eliminates the necessity for the user to repeatedly type in the same transformation coordinates.

A *pointsfile* file will contain between 4 to 10 lines; each line will contain the set of coordinate transformation points for the input map and the corresponding set of coordinates for the output map. The minimum number of lines for the transformation to take place is four.

The format of the *pointsfile* file is shown below:

```
Input Map   Output Map

x       y   x        y
x       y   x        y
x       y   x        y
x       y   x        y
```

In the format shown above the *x*'s and *y*'s can be thought of as eastings and northings, depending on what coordinate systems you are transforming to and from.

An example of the *pointsfile* file:

```
1       1       589000   4913000
1       17000   589000   4930000
17000   17000   610000   4930000
17000   1       610000   4913000
```

Within the *pointsfile* file, numbers on a line must be spaced apart with tabs or blanks. The example shown above was used to convert a map in digitizer coordinates (range of 1−18000) to UTM coordinates within the UTM zone for the Spearfish sample data base location.

Because this *pointsfile* file is not your usual GRASS data file, the user will have to keep track of where it is on the system. When the *pointsfile* option is used on the command line to input the transformation points, the program does not ask whether or not everything is acceptable before converting the vector file and the attribute file.

The user is advised to run this program interactively with a specific set of transformation coordinates and to examine the resulting residuals, to determine how accurate the transformation will be (i.e., pick points with known values in both coordinate systems). After the residuals are acceptable, those transformation coordinates

can be used with the program run non−interactively to transform other maps in the same geographic area.

WARNING

This is a general purpose program and can be fooled into giving low residuals. It is strongly suggested that any transformed map be checked for accuracy. The program assumes that the coordinate systems will be planimetric and has never been tested with negative values.

If this program is being used to transform maps from State Plane to UTM coordinates, and vice versa, users should be aware of the following points. This program will work better with State Plane zones that use the Transverse Mercator projection. Those are states that have their state zones splitting the state vertically, like Illinois. This program will not work as well with states that use the Lambert Conformal Conic projection. Those are states that have their state zones splitting the state horizontally, like Wisconsin. It is also best to keep the area being transformed relatively small.

SEE ALSO

v.digit
v.support

AUTHOR

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

Last changed: $Date: 2002/04/03 09:19:40 $

NAME

v.trim − Trims small spurs, and removes excessive nodes from a binary GRASS vector (*dig*) file.
(GRASS Raster Program)

SYNOPSIS

v.trim
v.trim help
v.trim input=*name* **output**=*name* [**trim**=*value*]

DESCRIPTION

v.trim scans the user−specified GRASS vector **input** file and removes from it all lines having a length less than a user−specified trimming factor. Excess nodes (those unnecessary to a line's definition) between line junctions are also removed. The resulting vector output is sent to a user−named **output** file; the original vector **input** file is not modified by *v.trim*.

The trimming factor parameter (**trim**=*value*) gives the user control over the size of small spurs or "dangling lines" to be removed. The trimming factor is expressed in the same units (map coordinates) as the vector (*dig*) data within the user's current GRASS data base LOCATION (e.g.: in meters for UTM locations; in pixels or cells for locations in an x,y coordinate system; etc.).

OPTIONS

The user can run this program either non−interactively or interactively. The program will be run non−interactively if the user specifies program arguments on the command line, using the form:

> **v.trim input**=*name* **output**=*name* [**trim**=*value*]

If vector map **input** and **output** names are given on the command line, any other parameter values left unspecified on the command line will be set to their default values (see below). Alternately, the user can simply type **v.trim** on the command line, without program arguments. In this case, the user will be prompted for needed parameter values using the standard GRASS *parser* interface.

Parameters:

input=*name*
: Name of an existing vector map layer in the user's current mapset search path containing lines to be "trimmed".

output=*name*
: Name of a new vector file to contained the "trimmed" output.

***trim**=value*

A user−specified trimming factor, denoting the length of trimmed lines in map units. All lines having a length less than this trimming factor will be "trimmed" (i.e., removed) from the named vector input file.
Default: 10 (in units of meters or cells)

NOTES

v.support must be run on the vector **input** file prior to running *v.trim*.

v.support must also be run on the resultant vector **output** file to build the needed topology information stored under the user's *dig_plus* directory.

r.line maintains the same format (binary or ASCII) and attribute type (linear or area edge) as those of the original vector (*dig*) **input** file.

A trimming factor of zero (0) will not remove any small spurs, but will remove all excess nodes.

SEE ALSO

v.digit
v.import
v.support
parser

AUTHOR

Mike Baba
DBA Systems, Inc.

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

NAME

v.what − Query the category contents of a (binary) vector map layer at user−selected locations.
(GRASS Vector Program)

SYNOPSIS

v.what
v.what help
v.what [**−1i**] **map=***name* [**east_north=***east,north*[,*east,north*,...]

DESCRIPTION

v.what outputs the category value(s) associated with user−specified location(s) in a vector map layer.

OPTIONS

If the **−i** flag is specified, the program activates the mouse, and expects the user to indicate the location(s) to be queried by depressing a mouse button over desired location(s) within the current geographic region in the active display frame on the graphic monitor.

If the **−i** flag is not used, the program expects eastings and northings to be entered from standard input. In this case, input is terminated by typing Control−D.

Flags:

−1
: Identify and query just one point location.

−i
: Query interactively using mouse.

Parameter:

map=*name*
: Name of an existing binary vector map in the user's mapset search path.

east_north=
: One or multiple coordinate pairs for query

EXAMPLE

Two sample *v.what* sessions are given below.

Although it is not necessary that the user first display a vector map to be queried in the active display frame, it is helpful to have a map displayed there for reference for interactive queries.

v.what −i map=*roads.24000*

After typing this, the user moves the mouse to a desired location on the displayed *roads* map layer, and presses the left mouse button to query the category value of the *roads* vector map at this location. The program then outputs the category value of a line type corresponding to this user−selected map location, for the vector map queried by the user.

The query may be repeated as often as desired using the left mouse button. The right button on the mouse is used to quit the *v.what* session.

Users can also use this program inside of shell scripts. For example, if the file *coords* contains three UTM coordinates:

```
599817.37 4922332.96
593512.25 4917170.38
604979.96 4921655.90
```

cat coords | v.what map=*landcover*

will return information about these three locations and then exit.

NOTES

v.what output can be redirected into a file.

d.what.rast can be used to interactively query the map category contents of multiple raster map layers at user−selected locations.

v.what was created from *d.what.vect* so that non−interactively queries could be made. Specifying the **−i** flag makes *v.what* behave just as *d.what.vect* does.

SEE ALSO

d.vect, *d.what.rast*, *d.what.vect*, *g.region*, and *parser*,

AUTHORS

Jim Hinthorn, Central Washington University, was the original author.
Dennis Finch, National Park Service
James Darrell McCauley, Agricultural Engineering, Purdue University
added the non−interactive part and renamed to *v.what*

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