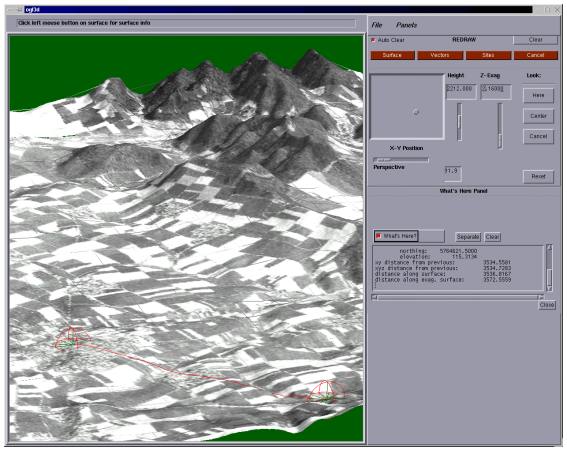
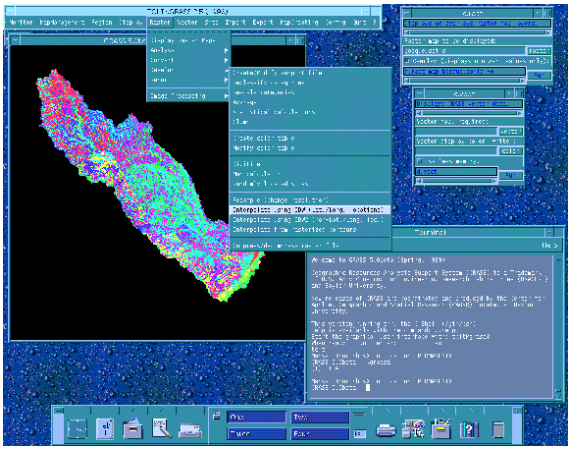
GRASS Reference Manual

Vector Commands





GRASS Development Team

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

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

grass@baylor.edu

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

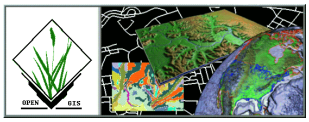


Table of Contents

Topic Page

GRASS Introduction

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

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

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

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

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

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

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

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

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

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

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

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

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

Look for our worldwide mirrors!

The GRASS Development Team is:

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

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

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

Note:

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

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

To reference GRASS:

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

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

v.alabel

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

GRASS VERSION
4.x, 5.x

SYNOPSIS
v.alabel
v.alabel help
v.alabel [-i] map=name [value=value]

DESCRIPTION
v.alabel allows the user to bulk-label (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) 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.

Flag:
-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 labeled. 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

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

v.apply.census

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

GRASS VERSION
4.x, 5.x

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]

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 Path/name of STF1 or PL94 input file

out Type of output: site | atts | table:Lxxx | - (stdout)
Default: -

Ef Path/name of text file with formula expression(s)
formula *map=expression*

name_field field name for parsing stdin lines (-a to ignore)
Default: -a

Zone UTM zone number; default is location zone
Options: 1-60
Default: 10

Spheroid 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.

DESCRIPTION
This program reads a previously selected subset of STF1 or PL94-171 U.S. Census Bureau demographic records (see *m.in.stf1.tpe* and *m.in.stf1.db3*), 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.

SOME PARAMETERS EXPLAINED BRIEFLY

out=
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 preceding the expression value instead of easting and northing. Lxxx is in same choice of representation as column designation in formula (see below).

formula=
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 is Keyword in stdin stream preceding 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.

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 preceded 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), or *m.in.stf1.db3* to print a simplified list of IDENTIFICATION SECTION field names. 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% population 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.db3* or *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 mandatory 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.

- *m.in.stf1.tpe* is used as a preprocessor to select subsets of STF1 or PL-171 tape format records for input to this program. *m.in.stf1.db3* is used as a preprocessor to select subsets of STF1 records from the CD-ROM in dBase 3 format. Unix utility programs such as 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.

v.area

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

GRASS VERSION
4.x, 5.x

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.

v.autocorr

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

GRASS VERSION
4.x, 5.x

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

v.cadlabel

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

GRASS VERSION
4.x, 5.x

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 labels 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 a 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

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

v.circle

NAME
v.circle - Create 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)

GRASS VERSION
4.x,5.x

SYNOPSIS
v.circle
v.circle help
v.circle -s[radius=value][radius_uom=name][area=value][area_uom=name]sitefile=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, i.e. (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, i.e. 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).

AUTHOR
Bruce Powell, National Park Service

v.clean

NAME
v.clean - Cleans out dead lines in GRASS vector files.
(GRASS Vector Program)

GRASS VERSION
4.x, 5.x

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.

SEE ALSO
v.digit, v.support

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

v.cutter

NAME
v.cutter - Polygon Cookie Cutter (Boolean AND Overlay)
(GRASS Vector Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
v.cutter
v.cutter help
v.cutter [-u] [-q] cutter=name data=name out=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. Both lines, sites, and polygons are clipped correctly.

OPTIONS
Flags:
-q Run quietly. Don't print percent information.

-u Intersect all areas even those that are unlabeled. Not generally useful and also not fully tested.

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

data=mapname Name of the binary vector file which is to be cut.

out=mapname Name of new vector file to be created.

NOTES
The user must run *v.spag -i* after running *v.cutter* to remove identical lines. The polygons of the cutter map must be labeled to be effective unless the *-u* flag is specified. The attributes of created polygons will be generated from the attributes of the data 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.

Currently the program generates many duplicate lines. So *v.spag -i* must be run on the new file to clean them up. The *-i* flag specifies to only do the identical line removal phase and is a new option to *v.spag* in release 4.1. Don't forget to use this flag, as it will run much faster, and will not make any other changes to your data.

Borders between areas with the same attributes are not dissolved.

There is no fuzzy data handling code, so expect to see slivers.

SEE ALSO
v.spag, v.digit, v.support, v.clean

AUTHOR

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

v.db.rim

NAME
v.db.rim - RIM database management/query interface for GRASS vector
(GRASS Vector Program)

GRASS VERSION
4.x

SYNOPSIS
v.db.rim
v.db.rim data_base

OVERVIEW
v.db.rim allows users 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 database through a command language defined in SECTION ONE or through 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 publicly by the University of Washington Academic Computing Services. Your system must have a FORTRAN 77 compiler to use this program.

DESCRIPTION
The vector databases are stored in a subdirectory named 'rim/vect' in the user's current mapset. Databases in other mapsets, selectable through the GRASS *g.mapsets* command, can be accessed for 'read-only' retrieval of records. Each mapset may have many databases. Each database 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 database needs to be opened.

Each vector database 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 database 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 database. 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.

SECTION ONE -- THE COMMAND VERSION
(Note: For each of the "dot commands", i.e., .make, described below there is a menu choice to select 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 database. 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 database. 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
 north4690673.30
 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 database 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* database. A message will be output if there is any problem writing the.backup file. Backup can only be done on databases 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 database. 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 database 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 database (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 database 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 database; each of these field types may only occur once in a database 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 database 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 database.

.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 database. 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 database 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 databases where retrieval 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 database.

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 labeled 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 database. 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 database. 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 database 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 database. 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
SYNOPSIS
v.db.rim

DESCRIPTION
When run interactively, *v.db.rim* allows you to create, manage and query information about vectors across the landscape on a data base through the 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 databases. For each mapset in the current GRASS mapset search path, the names of the existing vector databases are listed.

3. "Find" records in the database 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 raster, vector, and/or site maps; and/or display the selected vectors from the database; 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 raster and/or vector map names, if desired

 _______________  Raster 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 database 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 raster maps can be produced from a *v.db.rim* database 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 raster 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 database. (A new record is one whose number does not currently exist in the open database.) 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 discarded--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 MASKx
    (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 hitting 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

 __ 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 databases. Like 2 on main menu.

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

4. Choice 6 allows backup of the definition and data parts of a database to a transportable text file. To rebuild (or build for the first time) a *v.db.rim* database 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 database is thus created in the current mapset. Several *v.db.rim* commands should be run to verify the integrity of the newly created database.

5. This merely shows the screen layout of the currently open database. 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. Databases may also be backed up by copying the three binary files, which comprise the database 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 database 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 database from an existing GRASS vector map. This procedure will prompt for the vector map name and then add a record to the currently open database 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 database 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

v.db.rim interfaces to the RIM program. Both *rim* and *v.db.rim* contain FORTRAN code. The user must have access to a FORTRAN compiler in order to compile and use *v.db.rim*. See the FILES section, below, for location of source code for RIM and *b.rim*.

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.

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

SEE ALSO

RIM User's Manual by Jim Fox, Academic Computing Services Univ. of Washington
See especially Appendix B on redistribution of RIM.
GRASS 4.0 Installation Guide, by Jim Westervelt and Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

Gen.Maps, Gen.tractmap, g.mapsets, m.tiger.region, s.db.rim, s.menu, v.in.tiger

AUTHORS

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

v.digit

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)

GRASS VERSION
4.x, 5.x

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.
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
The GRASS 4.1 version of *v.digit* was upgraded so that it now no longer requires drivers to be written for each digitizer supported. Instead we are 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.

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

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

SEE ALSO
GRASS Tutorial - digit: Its Use and Its Features
v.import, v.in.ascii, v.out.ascii, v.out.rast, v.support

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

v.digit2

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)

GRASS VERSION
4.x

SYNOPSIS
v.digit2

DESCRIPTION
Note: This module is considered obsolete and has been modified into *v.digit*.

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

SEE ALSO
GRASS Tutorial digit Its Use and Its Features
v.import, v.in.ascii, v.out.ascii, v.out.rast, v.support, v.digit

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

v.export

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

GRASS VERSION
4.x,5.x

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 in 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 labeled 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.digitv.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

v.extract

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

GRASS VERSION
4.x, 5.x

SYNOPSIS
v.extract
v.extract help
v.extractFR [-dn] 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 or option (n) names, input vector file name, an output vector name, and the type of input map. 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 values or names.

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.

OPTIONS
Flags:
-d Dissolve common boundaries
Default: no

-n Use category names NOT category values.

Parameters:
type=name Select area, line, or site.
Options: area, line, site

input=name Input vector map name.

output=name Output vector map name.

new=value Enter 0 or a desired NEW category value.

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
$GISBASE/etc/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 values of 1 thru 4; any common boundaries are dissolved, and all areas of the new map will be assigned category value 1.

```
$GISBASE/etc/v.extract -dn list=Abc,Def1,12A,WWd &\
input=soils output=soil_groupa type=area new=0
```

Produces a new vector area file soil_groupa containing 'area' boundaries from soils with area category names of Abc,Def1, 12A,WWd; these names correspond to values 1 thru 4 of soils. Any common boundaries are dissolved, all areas of the new map will be retain their original category values 1 thru 4, in this case, since new was set to 0.

```
$GISBASE/etc/v.extract -n 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 value 1. The "names" (-n option) can be found in the file sample of the current directory.

The format for "sample" is:
Abc
Def1
12A
WWd

AUTHOR
R.L. Glenn, USDA, SCS, NHQ-CGIS

v.geom

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.

GRASS VERSION
4.x,5.x

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 minimizes (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 approximation 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: 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.geo*m 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 occurs 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

v.import

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

GRASS VERSION
4.x, 5.x

SYNOPSIS
v.import
v.import help

DESCRIPTION
This program performs all of the processes that are needed to convert ASCII DLG files, binary DLG files, ASCII SCS-GEF files, ASCII ARC Ungenerate files, ASCII DXF 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, SCS-GEF, or ARC files). 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 or SCS-GEF text data. This dig_att file is created for vector files by the labeling function of the GRASS *v.digit* program.

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

Import to GRASS Vector Format and Creates Needed Support Files
1 - ASCII DLG file to GRASS Vector Format
2 - Binary DLG file to GRASS Vector Format
3 - ASCII DIGIT file to GRASS Vector Format
4 - Binary DIGIT file to GRASS Vector Format
5 - ASCII SCS-GEF file to GRASS Vector Format
6 - ASCII ARC/INFO file to GRASS Vector Format
7 - ASCII DXF file to GRASS Vector Format
8 - ASCII TIGER file to GRASS Vector Format

If numbers 1-3 or 5-8 are chosen, *v.import* will respond with the current database units (in feet or meters), and ask if the new vector file is in the correct units for the database location. If the new vector file is not in the correct units, *v.import* will not allow it to be placed in the current database location. For each database location, all data layer should have the same units. If, for some reason, a data layer has different units than the rest of the data layers in the same database, a new database location will have to be created for it.

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.

Binary DLG File to GRASS Vector
Converts binary DLG files (which should be in the bdlg directory) to a vector file and creates the dig_plus and dig_att support files. The user is asked whether precedence should be given to Areas or Lines and if nodes should be snapped to other nodes within a calculated threshold.

This process is done in two phases:

1. The binary DLG (bdlg) file is converted to a binary vector file, and the dig_att support file containing attribute information is created.

2. 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:

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

Ascii SCS-GEF File to GRASS Vector
Creates the dig_plus, dig_att, and dig_cats support files. Creates a registration coordinates file.

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 $LOCATION/gef directory.
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 consistency across several mapsets (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 SCS-GEF 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).
Coord. System Desired (utm, stplane, ll, albers).

The program will then actively read the SCS-GEF data file and process it.

Note: 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.import*.

ASCII ARC/INFO Ungenerate Format Files to GRASS Vector Creates the dig_plus, dig_att, and dig_cats support files. The program will prompt you to enter the names of ARC/INFO files to be imported to GRASS. ARC/INFO vector files to be imported into GRASS must be exported from ARC/INFO using the ARC/INFO Ungenerate command. ARC/INFO vector files which are to are to be imported to GRASS, must be either line or polygon coverages. They must also be placed in a $LOCATION/arc directory. The section of the ARC/INFO manual that cover the Ungenerate command describes how to export line and polygon coverages.

A polygon coverage is represented by three files:

1) a lines file, which contains coordinates of all the area edge lines;
2) label-point file, which contains coordinates of label-points each with a unique label-point ID number. There is one label-point for each polygon defined in the lines file; and
3) a label-text file, which associates label-point ID number with a category value and attribute text.

A line coverage is represented by two files:

1) a lines file, which contains coordinates of the lines, each with a line-ID number; and
2) a label-text file, which associates each line-ID number with a category value and attribute text.

The program will start out by asking you which type of coverage is to be imported, as follows:

```
IMPORTING A POLYGON COVERAGE
The prompts that will be presented for coverage type "polygon."
   COVERAGE TYPE
   Enter "polygon" or "line"
   Hit RETURN to cancel request
   >
   Answer "polygon"

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

If you answer yes then vectors representing a box around the data will be inserted into the resulting GRASS vector file, otherwise no neatline will be created.

Next the program will prompt for the name of the lines-file containing the arc coordinates of the polygons. The lines- file is created with the Ungenerate LINES option and is the same format as the map_name.pol file created by the program. The following is the 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 the label-points file. The label-points files is created with the Ungenerate POINTS option and is the same format as the mapname.lab file created by the export.vect ARC program. The following is the 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
>
```

The last prompt for coverage type "polygon" asks for the name of the label-text file. This file associates each label-point ID number with a text string and is the same format as the mapname.txt file created by the export.vect ARC program. The following is the prompt:

```
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
>
```

The program will then scan the label-text file to determine how many columns are in the file and to determine which column should be used as the label-point ID number column. The program will then tell you how many lines and columns are in the label-text file. Next you will be prompted to enter the number of the column to be used for GRASS category values. The category number column MUST contain only integers. Enter the number of the column that is to be used for GRASS category values and the number the column to be used for GRASS attribute text. The attribute text column can contain a floating point number, an integer, or a word. Enter the number of the column that should be used for GRASS attribute text. Once you enter the category and attribute column numbers, the program will begin conversion of the ARC/INFO Ungenerate files into GRASS vector format.

IMPORTING A LINE COVERAGE

First, you are prompted for the name of the lines-file containing the arc coordinates of the lines. The lines-file is created with the Ungenerate LINES option and is the same format as the mapname.lin file created by the export.vect ARC program.

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

The last prompt for coverage type "line" asks for the name of the label-text file. This file associates each line-ID number with a text string and is the same format as the mapname.txt file created by the export.vect ARC program.

```
LABEL-TEXT FILENAMES
    Enter name of file associating line ID numbers numbers with label text.
    Hit RETURN if there is no such file
    >
```

The program will scan the label-text file to determine how many columns are in the file and will then tell you how many columns are in the label-text. Next you will be prompted to enter the number of the column to be used for line-ID numbers. Enter the number of the column that is to be used for line-ID numbers.

Next you will be prompted to enter the number of the column to be used for GRASS category values. The category number column MUST contain only integers. Enter the number of the column that is to be used for GRASS category values and the number of the column to be used for GRASS attribute text. The attribute text column can contain a floating point number, an integer, or a word. Enter the number of the

column that should be used as for GRASS attribute text. Once you enter the column numbers the program will begin conversion of the ARC/INFO Ungenerate files into GRASS vector format.

DATA FILE FORMATS
Following are examples of the data files discussed in the section above.

LINES FILE, also known as xxx.lin or xxx.pol file. This type of file can be created in ARC/INFO by using the lines subcommand of the Ungenerate command. Each line, or 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 only important for line coverages. 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 xxx.lab file. This type of file can be 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   19,399.8481 0 00
281,079.875    1,678.8262 1153
3    955,952.500   10,229.6373 2198
441,530.875 926.8874 3173
587,900.188    1,900.9095 4133
6    166,125.125    3,512.9506 5153
729,460.563 824.9687 6173
8    1022769.875    9,105.7078 7209
951,385.500    1,075.6389 8173
10   376,834.875    4,470.027    10 9 92
```

```
11    65,802.688    1,575.088    1110163
```

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 labeled).

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 documentation 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.scs

AUTHOR
R.L. Glenn, USDA, SCS, NHQ-CGIS

v.in.arc

NAME
v.in.arc - Converts data in ARC/INFO format to GRASS's vector format, and stores output in the user's current GRASS mapset.
(GRASS Vector Data Import Program)

GRASS VERSION
4.x, 5.x

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: 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.

Once converted with the ARC/INFO Ungenerate command, the files to be imported into GRASS must be placed in a directory named arc in the user's current mapset. If the arc directory does not exist, it must be created (e.g., with the command mkdir $LOCATION/arc) before copying the ARC-INFO files to be converted into it.

(*v.in.arc* can be used to convert ARC-INFO data from other mapsets as well, since the program searches for the specified input file names in the arc directories, if any exist, in the mapsets in the user's current mapset search path.)

OPTIONS
Program parameters and the flag have the following meanings.

Flag:
-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   19,399.8481 0 0 0 0
  281,079.8751,678.8262 115 3 3
  3    955,952.500   10,229.6373 219 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 (a.k.a., 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 (a.k.a., 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.

```
1711539.875000   4651743.000000
2711429.000000   4650632.000000
3711027.625000   4651736.000000
4711022.625000   4651519.000000
5710482.750000   4651494.000000
6710474.500000   4651667.000000
7709269.750000   4651018.000000
8709726.500000   4651604.000000
9708926.375000   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  19,399.848   1   0   0  0
        2 81,079.875    1,678.826    2    1   15   3
        3955,952.500   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
        6166,125.125    3,512.950    6    5   15   3
        7 29,460.563 824.968    7    6   17   3
        81022769.875    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
        1165,802.688    1,575.088   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.

SEE ALSO

v.out.arc, v.support

AUTHOR

Dave Johnson, DBA Systems, Inc.

v.in.ascii

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

GRASS VERSION
4.x, 5.x

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. 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.

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 output from *v.in.ascii* will be placed into $LOCATION/dig.

SEE ALSO
v.digit, v.import, v.out.ascii, v.support

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

v.in.dlg

NAME
v.in.dlg - Converts an ASCII USGS DLG-3 Optional file to a binary GRASS vector (dig) file.

GRASS VERSION
4.x, 5.x

SYNOPSIS
v.in.dlg
v.in.dlg help
v.in.dlg [-l] inputname outputname [mattname] [baseval]

This program converts an ASCII USGS DLG-3 (dlg)file into a binary GRASS vector (dig) file.
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 corresponding multiple attributes. In this case matt file contains identifiers starting with 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*.

Give precedence to line information (default is area).

Parameters: Name of USGS DLG-3 Optional format input file. Name to be assigned to the binary GRASS vector files created. Name of file with multiple attributes (optional). 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. 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.

AUTHORS

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

v.in.dlg2

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)

GRASS VERSION
4.x, 5.x

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*.

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

v.in.dlg.scs

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)

GRASS VERSION
4.x,5.x

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; i.e. 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

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

v.in.dxf

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

GRASS VERSION
4.x, 5.x

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.

Flag:
-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=in_name:out_name[,in_name:out_name,...] lines=name[,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=in_name:out_name[,in_name:out_name,...] labels=name[,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 15, the GRASS vector map layer output would be named streams.15.

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

v.in.dxf3d

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

GRASS VERSION
4.x

SYNOPSIS
v.in.dxf3d
v.in.dxf3d help
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 automatically importing, to GRASS vector file, layers with Z values (isolines and level contours) from DXF format files.

First the *v.in.dxf* program should be run to 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 automatically, for a maximum 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.

OPTIONS
Parameters:
dxf Name of the DXF input design file from where will be extracted the Z values.

line 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

v.in.poly

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

GRASS VERSION
4.x,5.x

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:
sites Name of input file (omit or use - for input from stdin)
Default: - (stdin)

vect Name of new vector map to create

radius Radius of polygon's circumscribed circle default: none

segments 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 misshapen 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.

v.in.scsgef

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

GRASS VERSION
4.x, 5.x

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 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.

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 the 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*.

OPTIONS
Flag:

-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

v.in.sdts

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)

GRASS VERSION
4.x

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 appropriate 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 mismatch 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 thereof. 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.

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 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

v.in.shape

NAME
v.in.shape - Read an ESRI ArcView file
(GRASS Raster Data Import Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
v.in.shape
v.in.shape help
v.in.shape [-s] input=name [mapset=name] [verbose=debug level] [logfile=name] [attribute=a_name]

DESCRIPTION
The *v.in.shape* program is designed to import ESRI ArcView Shapefiles.

v.in.shape will be run non-interactively if the user specifies program arguments on the command line, using the form:

v.in.shape [-s] input=name [mapset=name] [verbose=debug level] [logfile=name] [attribute=a_name]

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
Grass files created have the name extracted from the basename of the shapefile.

OPTIONS
v.in.shape requires the user to enter the following information:

Parameters:
mapset=name For creating a new mapset for the data imported. This may be useful, since we cannot create a projection info file and a default window in an existing mapset. When a file is imported in the current mapset, you should take care that they are in the same projection. Unfortunately, you may run in trouble after that when using *g.region* (Cf BUGS).

verbose=debug level Number between 0 (no trace of what's happening) and 9 (very verbose log).

logfile=name Name of file where log info will be written. By default log info are directed to stderr.

attribute=a_name Name of the attribute to use as the category number in dig_att. Give a value of list to see a list of available attributes.

BUGS AND CAVEAT
Polygons with internal holes are not treated properly. The holes are just treated as a continuation of the polygon.

There is no support for projection.

Label for polygons are located on the edge, rather than interior to the polygon. This is ambiguous in cases of polygons that share boundaries with other polygons and will cause conflicts when running *v.support*.

New mapset are always created with proj=0 zone=99. If the default proj and zone are not the same, *g.region* complaints and *d.vect* or *d.rast* refuse to display your data. The only thing to do is to import in the current mapset.

SEE ALSO
m.in.e00, g.mapsets, g.region, g.setproj, v.support v.to.rast

AUTHOR
Frank Warmerdam (warmerda@home.com)

Based on Shapelib (http://gdal.velocet.ca/projects/shapelib/)

v.in.tig.basic

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

GRASS VERSION
4.x, 5.x

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]

OPTIONS
Parameters:
t1 TIGER Type 1 path/file name

t2 TIGER Type 2 path/file name

out Name of vector map to create

zone UTM zone number; default is location zone
Options: 1-60
Default: (current zone)

spheroid Spheroid for LL to UTM conversion; see *m.gc.ll*
Default: clark66

tlid Path/file for list of TLID numbers to process from TIGER.1 File
Default: none

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)

DESCRIPTION
This program creates a GRASS vector map in the current mapset (UTM or Lat-Long locations only) with labeled 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.

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.stfl.tape* and/or unix text tools can be used to extract/concatenate useful subsets of TIGER type records prior to running *v.in.tiger*.basic.

AUTHOR

Dr. James Hinthorne, GIS Laboratory, Central Washington University

v.in.tig.lndmk

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

GRASS VERSION
4.x, 5.x

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]

PARAMETERS
t1 Path/name of Type 1 Tiger File
t2 Path/name of Type 2 Tiger File
t7 Path/name of Type 7 Tiger File
t8 Path/name of Type 8 Tiger File
tI Path/name of Type I Tiger File
tP Path/name of Type P Tiger File

input file path/name for input commands
vect Name of vector map to create
site Name of site map to create
zone UTM zone number; default is location zone
options: 1-60
default: 12

spheroid 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.

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 labeled 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 labeled 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 labeled.

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.

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 specified 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 it 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.

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

D00Landmark Feature, Classification Unknown or Not Elsewhere Classified

D10Military installation

D20Multihousehold and transient quarters
D21Apartment building or complex
D22Rooming or boarding house
D23Trailer court or mobile home park
D24Marina
D25Crew of vessel
D26Housing facility for workers
D27Hotel, motel, resort, spa, YMCA, or YWCA
D28Campground
D29Shelter or mission

D30Custodial facility
D31Hospital
D32Halfway house

D33Nursing home, retirement home, or home for the aged
D34County home or poor farm
D35Orphanage
D36Jail or detention center
D37Federal penitentiary, state prison, or prison farm

D40Educational or religious institution
D41Sorority or fraternity
D42Convent or monastery
D43Educational institution
D44Religious institution

D50Transportation terminal
D51Airport or airfield
D52Train station
D53Bus terminal
D54Marine terminal
D55Seaplane anchorage

D60Employment center
D61Shopping center or major retail center
D62Industrial building or industrial park
D63Office building or office park
D64Amusement center
D65Government center
D66Other employment center

D70Tower
D71Lookout tower

D80Open space
D81Golf course
D82Cemetery
D83National park or forest
D84Other federal land
D85State or local park or forest

D90Special purpose landmark
D91Post office box ZIP code

v.in.tig.rim

NAME
v.in.tig.rim - Imports Census Bureau line data (TIGER files) to GRASS vector format.
(GRASS Vector Data Import Program)

GRASS VERSION
4.x, 5.x

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, GIS Lab, Central Washington University, Ellensburg, WA.

v.in.tiger.scs

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)

GRASS VERSION
4.x, 5.x

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 documentation 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

NOTICE
This program is part of the contrib section of the GRASS distribution. As such, it is externally contributed code that has not been examined or tested by the Office of GRASS Integration.

v.in.transects

NAME
v.in.transects - import transect data to a GRASS vector map
(GRASS Vector Program)

GRASS VERSION
4.x, 5.x

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 .

COMMAND LINE OPTIONS
Flag:
-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

v.llabel

NAME
v.llabel - Bulk-labels unlabeled lines in a binary GRASS vector file.
(GRASS Vector Program)

GRASS VERSION
4.x

SYNOPSIS
v.llabel
v.llabel help
v.llabel [-in] map=name [value=value]

DESCRIPTION
v.llabel allows the user to bulk-label currently unlabeled lines (not 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.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 labeling 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) 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 lines incrementally. For each unique, unlabeled line 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 labeled. This map layer must exist in the user's current GRASS mapset.

value=value The category value to be assigned to all unlabeled 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 polygon in the vector map. Default: 1 .

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, v.alabel

AUTHORS
James Darrell McCauley, Agricultural Engineering, Purdue University
Dave Gerdes, U.S. Army Construction Engineering Research Laboratory

v.make.subj

NAME
v.make.subj - Create a "subject" file from all category labels found in a set of listed vector maps.
(SCS GRASS Vector Program)

GRASS VERSION
4.x,5.x

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

v.merge

NAME
v.merge - Merges vector map files.
(SCS GRASS Vector Program)

GRASS VERSION
4.x, 5.x

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

v.mkgrid

NAME
v.mkgrid - Creates a (binary) GRASS vector map of a user-defined grid.
(GRASS Vector Program)

GRASS VERSION
4.x, 5.x

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.

Flag:
-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, parser

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

v.mkquads

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)

GRASS VERSION
4.x, 5.x

SYNOPSIS
v.mkquads
v.mkquads help
v.mkquads [-esrvx] map=name

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.

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 UTM coordinates will be displayed each time the program is executed. The spheroid being used for the lat/lon to UTM conversions is clark66.

BUGS
Currently, this program only works for GRASS locations in a Universal Transverse Mercator (UTM) coordinate system (in meters). 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, parser

AUTHORS
Michael Higgins, U.S. Army Construction Engineering Research Laboratory
Marilyn Ruiz, U.S. Army Construction Engineering Research Laboratory

v.out.arc

NAME
v.out.arc - Converts GRASS vector files to ARC/INFO's "Generate" file format.
(GRASS Vector Data Export Program)

GRASS VERSION
4.x, 5.x

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. 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 (a.k.a., 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.

v.out.ascii

NAME
v.out.ascii - Converts a binary GRASS vector map layer into an ASCII GRASS vector map layer. (GRASS Vector Data Export Program)

GRASS VERSION
4.x, 5.x

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

v.out.dlg

NAME
v.out.dlg - Converts binary GRASS vector data to DLG-3 Optional vector data format.
(GRASS Vector Data Export Program)

GRASS VERSION
4.x, 5.x

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

v.out.dlg.scs

NAME
v.out.dlg.scs - Converts binary GRASS vector data to binary DLG-3 Optional vector data format. (SCS GRASS Vector Data Export Program)

GRASS VERSION
4.x, 5.x

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 corresponding 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

v.out.dxf

NAME
v.out.dxf - GRASS vector format to DXF format conversion program.
(GRASS Vector Data Export Program)

GRASS VERSION
4.x, 5.x

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
Chuck Ehlschlaeger, U.S. Army Construction Engineering Research Laboratory

v.out.idrisi

NAME
v.out.idrisi - exports binary GRASS vector files to ASCII IDRISI format.
(GRASS Vector Program)

GRASS VERSION
4.x

SYNOPSIS
v.out.idrisi

DESCRIPTION
This is a quick and dirty export routine from GRASS to IDRISI. The program exports binary GRASS vector files to ASCII IDRISI format.

The user will be prompted for the name of the file to be exported and the new filename.

AUTHOR
Philip Verhagen

v.out.moss

NAME
v.out.moss - Converts GRASS site, line, or area data into MOSS import format.
(GRASS Vector Data Export Program)

SYNOPSIS
v.out.moss

GRASS VERSION
4.x, 5.x

DESCRIPTION
This program produces a MOSS import 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 MOSS import 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.

v.out.scsgef

NAME
v.out.scsgef - Converts binary GRASS vector data to ASCII SCS-GEF vector data format.
(SCS GRASS Vector Data Export Program)

GRASS VERSION
4.x, 5.x

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

v.out.sdts

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)

GRASS VERSION
4.x

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 will 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 appropriate, 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/Directory 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

v.patch

NAME
v.patch - Creates a new binary vector map layer by combining other binary vector map layers.
(GRASS Vector Program)

GRASS VERSION
4.x, 5.x

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.

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, parser

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

v.plant

NAME
v.plant - module to insert intermediate points on GRASS vector lines.
(GRASS Vector Program)

GRASS VERSION
4.3, 5.x

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 that 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 (e.g. the coordinates of the end points) in one projection becomes an arc that needs more parameters in other projections. In GIS systems that 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.

TO DO
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).

AUTHOR

CSIRO Exploration & Mining Report 239F, March 1996.
S.J.D. Cox, AGCRC, CSIRO Exploration & Mining, Nedlands, WA

v.proj

NAME
v.proj - Allows projection conversion of vector files.

GRASS VERSION
4.x, 5.x

SYNOPSIS
v.proj
v.proj help
v.proj map=name [out=name] inloc=name [dbase=name] [set=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.

Parameters:
map=name Name of the input vector file.

out=name Name of the output vector file.

inloc=name Name of the location containing input vector file

dbase=name Name of the database containing input location

set=name Name of the mapset containing input vector file

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.

SEE ALSO
g.setproj

AUTHOR
Irina Kosinovsky, US ARMY CERL
M.L. Holko, USDA, SCS, NHQ-CGIS
R.L. Glenn, USDA, SCS, NHQ-CGIS

v.proj (old)

Note: use *v.proj* above

NAME
v.proj - Allows projection conversion of vector files.
(SCS GRASS Vector Program)

GRASS VERSION
4.x

SYNOPSIS
v.proj
v.proj help

DESCRIPTION
allows a user to create a new vector map in another mapset with a different projection, from an existing vector map in the current or some other user-specified mapset.

Note: If the "outset" mapset is not available, *v.proj* will create it in the current location. If the "outset" mapset exists, *v.proj* reads its projection information, and uses it to convert to the output projection. If the "outset" projection information is not available, *v.proj* runs the program *m.setproj* to create it.

EXAMPLE
The current mapset (Map_utm14) has a utm projection, the coordinates are for zone14 meters, and contains the map sample. The user wants to create a new map of sample in the mapset lambert (which is a Lambert Conformal Conic projection mapset).

The command:

v.proj map=sample outset=lambert

will create a new map sample in the mapset lambert, changing the utm coordinates of sample into lambert coordinates. If the mapset lambert did not exist, *v.proj* would have created it. *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.

SEE ALSO
m.setproj

AUTHOR
M.L. Holko, USDA, SCS, NHQ-CGIS
R.L. Glenn, USDA, SCS, NHQ-CGIS

v.prune

NAME
v.prune - Prunes points from binary GRASS vector data files.
(GRASS Vector Program)

GRASS VERSION
4.x, 5.x

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.

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 database units on the ground (e.g., in ground meters for UTM databases). 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.digit, v.in.ascii

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

v.psu

NAME
v.psu - Program to build PSU polygon and PSU sites from single labeled line segment.
(SCS GRASS Vector Program)

GRASS VERSION
4.x, 5.x

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

v.psu.subj

NAME
v.psu.subj - Converts SCS ISU PSU offset file to GRASS subject file.
(SCS GRASS Vector Program)

GRASS VERSION
4.x, 5.x

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.

Parameter:
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

v.random

NAME
v.random - Creates a GRASS site_lists file of randomly placed symbols (sites) within a GRASS vector area.
(SCS GRASS Vector Program)

GRASS VERSION
4.x, 5.x

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

v.reclass

NAME
v.reclass - Changes vector category values for an existing vector map.
(SCS GRASS Vector Program)

GRASS VERSION
4.x, 5.x

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.

OPTIONS
Flag:
-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
$GISBASE/etc/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 soils 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:
Def1area name 2:
12A . . .
WWd area name n:

The format for "category value" is:

1

10

12

15

INTERACTIVE MODE

v.reclass
The first question asked is the map type:

```
Enter the type of map (area, line, site) [area] :
```

The default is for areas.

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.

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.

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.

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.

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]
```

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:

When no entry is provided the program will begin.

SEE ALSO
v.extract

AUTHOR
R.L. Glenn, USDA, SCS, NHQ-CGIS

v.reclass.inf

NAME
v.reclass.inf - Generate new vector map layer derived from attribute data in the currently selected database.
(GRASS-RDBMS Vector Interface Program)

GRASS VERSION
4.x, 5.x

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 statements 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 statement, 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 Dissolve 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
```

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, v.reclass.inf

AUTHOR
James A. Farley, Wang Song and W. Fredrick Limp University of Arkansas, CAST

v.report

NAME
v.report - Generates statistics for vector files.
(SCS GRASS Vector Program)

GRASS VERSION
4.x,5.x

SYNOPSIS
v.report
v.report help
v.report [-hfq] 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:
-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

v.rmedge

NAME
v.rmedge - Selects edge vectors from an existing vector map, removes them, and creates a new vector map.
(SCS GRASS Vector Program)

GRASS VERSION
4.x,5.x

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 RELABELED 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

v.scale.random

NAME
v.scale.random - Creates a site_lists file of randomly placed symbols within a GRASS vector area. (SCS GRASS Vector Program)

GRASS VERSION
4.x, 5.x

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 program 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.

OPTIONS
Flag:
-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, v.random

AUTHOR
M.L. Holko, USDA, SCS, NHQ-CGIS

v.sdts.dq.cp

NAME
v.sdts.dq.cp - installs SDTS data quality reports.
(GRASS Vector Data Export/Processing Program)

GRASS VERSION
4.x

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-existing 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

v.sdts.meta

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)

GRASS VERSION
4.x

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

v.sdts.meta.cp

NAME
v.sdts.meta.cp - installs supplementary metadata file preparatory to creation of an SDTS export dataset. (GRASS Vector Data Export/Processing Program)

GRASS VERSION
4.x

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-existing 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 field 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

v.spag

NAME
v.spag - Process spaghetti-digitized binary vector file.
(GRASS Vector Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
v.spag
v.spag help
v.spag [-i] 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.

COMMAND LINE OPTIONS
Flag:
-i Only do Identical line removal pass.

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.

SEE ALSO
v.clean, v.digit, v.support

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

v.stats

NAME
v.stats - Prints information about a binary GRASS vector map layer.
(GRASS Vector Program)

GRASS VERSION
4.x, 5.x

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, parser

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

v.support

NAME
v.support - Creates GRASS support files for (binary) GRASS vector data.
(GRASS Vector Program)

GRASS VERSION
4.x, 5.x

SYNOPSIS
v.support
v.support help
v.support [-spr] map=name [option=name] [threshold=value]

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. Valid only with build option and *-s* flag.

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

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

v.surf.rst

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*)

GRASS VERSION
4.x, 5.x

SYNOPSIS
v.surf.rst
v.surf.rst help
v.surf.rst [-drct] 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] [devi = name] [treefile = name] [overfile = name]

DESCRIPTION
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 fx, fy, fxx, fyy, fxy 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 that 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. 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] [devi = name] [treefile = name] [overfile = 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).

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 tension for 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 are 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 smoothing 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 smoothing, 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.tps*): the given tension is applied to normalized data (x/dnorm..), which 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 = tension*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.

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 computing 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 that 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://www.cecer.army.mil/grass/viz/VIZ.html and http://www2.gis.uiuc.edu:2280/modviz/.

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
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.

v.surf.spline

NAME
v.surf.spline - GRASS module to interpolate vector contour data by fitting a cubic spline function to profiles in four directions. It has two advantages over other interpolation techniques: (1) The RMS Error can be calculated for EACH CELL in the DEM as well as an overall RMSE. Spatial variation in error can examined; (2) A cubic spline has the property of 2nd derivative being continuous therefore having implications for slope measurements. Terracing effects should be minimized.

GRASS VERSION
4.x

SYNOPSIS
v.surf.spline[-rs] in=name out=name [interval=value]

OPTIONS
Flags:
-r Rasterize contours using rooks case adjacency

-s Constrain interpolation using simple truncation

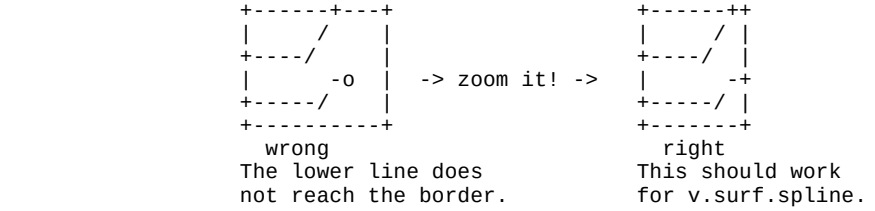
Parameters:
in Vector contour map to be interpolated.

out Resultant Digital Elevation Model.

interval Contour interval (or 0 for no interval constraint).
Default: 0

USAGE RECOMMENDATION
```
r.contour input=elevation.dem output=contours.test step=20
v.support option=build map=contours.test
g.region vect=contours.test
v.surf.spline in=contours.test out=dem.test interval=20 -s > /dev/null
```

SPECIAL NOTE
If you want to interpolate from self-digitized, avoid an "STEP 2-ERROR" in this way:
The region settings must satisfy the rule that at least one vector line has to hit the border. Totally at least four vector lines have to hit each of the four regional borders. Check this by displaying your vector contour file and use the *d.vect.zoom* command eventually. Or you digitize more lines.



NOTE
Method suggested by Yoeli(1986) and discussed in Wood and Fisher (1993).

SEE ALSO
r.surf.gauss, r.surf.random, r.surf.idw, r.surf.idw2, r.surf.contour

REFERENCES

Yoeli, P. (1986) Computer executed production of regular grid of height points from digital contours, The American Cartographer, 13(3), pp.219-229.

Wood, J and Fisher, P (1993) Assessing interpolation accuracy in Elevation Models, IEEE Computer Graphics and Applications, 13(2), pp.48-56.

v.to.gnuplot

NAME
v.to.gnuplot - outputs an ASCII vector map in GNUPLOT format
(GRASS Shell Script)

GRASS VERSION
4.x

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 *g.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:

g.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, g.gnuplot

AUTHOR
James Darrell McCauley, Agricultural Engineering, Purdue University

v.to.rast

NAME
v.to.rast - Converts a binary GRASS vector map layer into a GRASS raster map layer.
(GRASS Vector Program)

GRASS VERSION
4.x, 5.x

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.

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

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

v.to.sites

NAME
v.to.sites - Converts point data in a binary GRASS vector map layer into a GRASS site_lists file. (GRASS Vector Program)

GRASS VERSION
4.x, 5.x

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 within 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 data is used instead of attribute as a site description (valid only when -a is used).

-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)

SEE ALSO
d.sites, s.db.rim, s.menu, v.db.rim, v.digit, parser

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

v.transform

NAME
v.transform - Transforms an ASCII vector map layer from one coordinate system into another coordinate system.
(GRASS Vector Program)

GRASS VERSION
4.x, 5.x

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.

Flag:
-y Suppress the printing of residuals or other information to standard output.

Parameters:
input=name Name of the ASCII vector map layer to be transformed.

output=name Name to be assigned to the resultant (transformed) ASCII 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 ASCII (dig_ascii) vector map and its associated attribute (dig_att) file into the requested coordinate system.

Remember to run *v.support* or *v.import* 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.import, v.support

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

v.trim

NAME
v.trim - Trims small spurs, and removes excessive nodes from a binary GRASS vector (dig) file.
(GRASS Raster Program)

GRASS VERSION
4.x, 5.x

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 user interface described in the manual entry for *parser*.

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
r.line, r.thin, v.digit, v.import, v.support, parser

AUTHOR
Mike Baba, DBA Systems, Inc.

v.what

NAME
v.what - Query the category contents of a (binary) vector map layer at user-selected locations.
(GRASS Vector Program)

GRASS VERSION
4.x

SYNOPSIS
v.what
v.what help
v.what [-1i] map=name

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:
-l 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.

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, 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*