John Lloyd
MEA582 Fall 2011
Final Project: GRASS and the python script v.transects.py
November 28, 2011

Table of Contents

Introduction

In this project I setup two GRASS environments and tested a user developed python script in both environments comparing the results.

I setup both Microsoft Windows and Linux environments for GRASS development and test. I built these environments from GRASS source code. By building from source code I was able to use the latest versions of the GRASS application.

I then used my environments to test and debug the python script. A researcher in shore dynamics provided the python script. The script partitions the shore by creating line segments perpendicular to the shoreline called transects.

Approach

The approach I took in this project was to setup my development and testing environments, run a generic python scripts in the environments, and then test the provided script.

Environment setup

GRASS on Linux built from source

I built my GRASS Linux environment on CentOS Linux 5.7. CentOS Linux is a free distribution of an enterprise class operating system based on sources available from a "prominent North American Enterprise Linux vendor" (http://www.centos.org/). It is thought to be equivalent to Red Hat Enterprise Linux, although this is not explicitly stated.

I downloaded the additional packages specified on the GRASS Wiki and then the GRASS source code. I chose the source for GRASS version 6.4.2RC. After I all downloads completed, I compiled and installed GRASS using the steps on the GRASS Wiki.

GRASS on Microsoft Windows built from source

In addition to building a GRASS environment on Linux, I built a GRASS environment on Microsoft Windows7 Professional. I downloaded the source for GRASS 6.4.2RC and followed the instructions on the GRASS Wiki to compile the source in a Minimalist GNU for Windows (MinGW) build environment. I downloaded the required packages using the OSGeo4W installer. OSGeo4W installs open source geospatial software on Windows.

Running python scripts

After I setup the Linux and Windows environments, I ran python scripts via GRASS in text mode.

Python script vtransects.py

After setting up the environment and verifying that I could run python scripts, I tested the python script vtransects.py provided by Eric Hardin (http://www4.ncsu.edu/~ejhardi2/vTransect.html). The paper *Measuring short-term geomorphologic evolution in the Outer Banks of North Carolina* by Helena Mitasova and Eric Hardin contains the background for the script. In the paper they use the script to partition the shoreline along the Outer Banks. The script creates edges of rectangles perpendicular to the shoreline and extending for a distance given in the script's parameters and separated by a distance given in another script parameter. The user has the option of creating a line vector layer representing the edges of the rectangles perpendicular to the shoreline or creating area polygons by adding the edges roughly parallel to the shore.

However, researchers may use this script for purposes other than shoreline partitioning. The script accepts any line feature as input, so the script could partition for example a stream or road network into segments.

Testing python v.transects.py script

Test methodology

I tested the provided script in both Linux and Windows environments. I first executed my test cases in my Linux environment using the script as provided to me. My assumption was the provider of the script had already tested the script in a Mac OS environment and, since Mac OS and Linux are both UNIX like environments, I expected to find fewer issues in Linux than Windows. Any issues that I did find I recorded and when possible applied fixes.

Next, I executed the same test cases in my Windows environment. Any additional problems I found in this environment I fixed when possible. I then exported the transect line and area vector layers to ASCII format for both the Linux and Windows outputs and used a difference tool to verify both environments produced the same output layers.

I returned to the Linux environment for the final phase to verify that modifications made while testing in Windows did not introduce new defects.

Test cases

I created four groups of test cases; a parameter validation group, a Nags Head data group, additional line layers group, and a group in the WGS84 coordinate system. The parameter validation group simply verified that the script correctly handled invalid inputs. The Nags Head data group tested a range of parameter inputs using the shoreline data matching the scripts intended purpose. The additional line layers group tested the script with various shapes of polylines not representative of shorelines. The inputs in these test cases were all NC State Plane Meters. The WGS84 group tested a subset of the lines re-projected into WGS84.

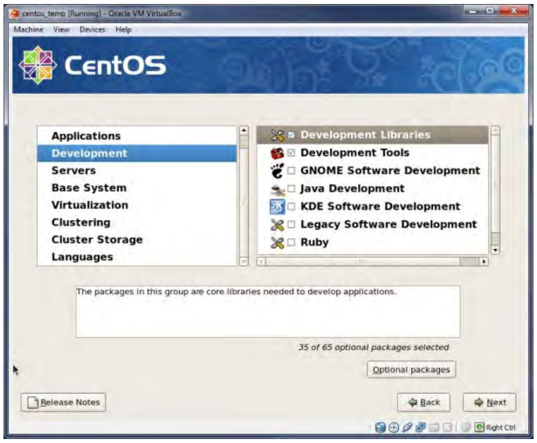
Results

Installing GRASS from Source

Linux Environment

CentOS Install

I built GRASS in my CentOS Linux 5.7 environment. Before I built GRASS I had to prepare the environment by installing various packages. When I did the CentOS install I selected the “Development Libraries”, “Development Tools” and “X Software Development” groups of packages (*Figure 1 CentOS Install Package Groups*). If these were not selected at install then I would have had to install them later using the “yum” tool (ex. Yum groupinstall “X Software Development”).



- ☑ Development Libraries
- ☑ Development Tools
- □ GNOME Software Development
- □ Java Development
- □ KDE Software Development
- □ Legacy Software Development
- □ Ruby

The packages in this group are core libraries needed to develop applications.

35 of 65 optional packages selected

Figure 1 CentOS Install Package Groups

GRASS Compile

I followed the steps outlined on the GRASS Wiki (http://grass.osgeo.org/wiki/Compile_and_Install) to download the prerequisite packages, download the GRASS source, and build and install the GRASS source.

I downloaded the source and built the libraries for the following packages in my environment. The standard procedure for building the libraries is to uncompress the downloaded file, run “./configure”, “make”, and “make install” as the “root” user.

- tiff-3.9.5.tar.gz – provides support for the TIFF image format and is available at http://www.remotesensing.org/libtiff/
- proj-4.7.0.tar.gz – a cartographic projections library available at http://trac.osgeo.org/proj/
- gdal-1.8.1.tar.gz - a translation library for raster geospatial formats available at http://www.gdal.org/

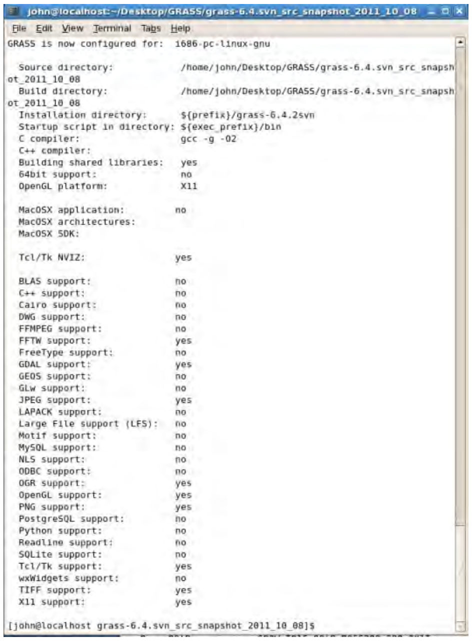
I installed additional prebuilt libraries using the standard Linux “yum install” method. These libraries are listed below.

- tcl-devel – development files and man pages for Tool Command Language (TCL)
- tk-devel – header files and documentation for writing Tk extensions in C, C++, etc.
- fftw-devel – a C subroutine library for Discrete Fourier Transforms
- python-ctypes – a python module to handle C data types
- numpy – a scientific computing package for python

The final step in setting up the Linux environment was to download the GRASS source and build it. Various version of the source are available on the GRASS Download site (http://grass.osgeo.org/download/software.php). I chose the GRASS 6.4.2RC source snapshot of 2011-10-08. Then, I followed the standard Linux procedure to build by executing "./configure", "make", and "make install" as the "root" user (*Figure 2 GRASS ./configure output* and *Figure 3 GRASS make output*).

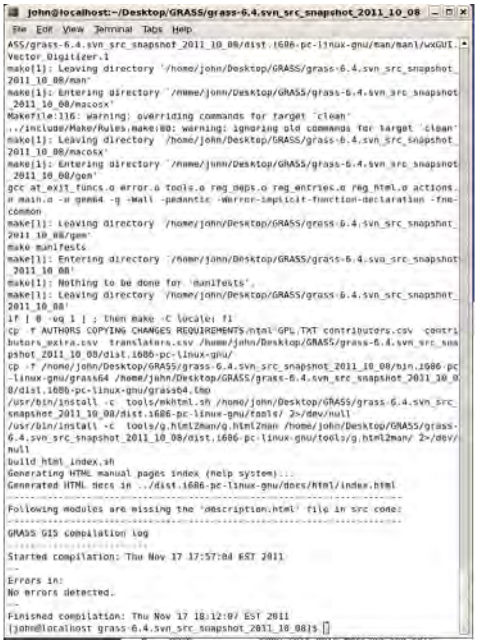
After building GRASS the Linux shell command below started the GRASS application.

- /usr/local/bin/grass64


john@localhost:~/Desktop/GRASS/grass-6.4.svn_src_snapshot_2011_10_08
File Edit View Terminal Tabs Help

```
GRASS is now configured for:  i686-pc-linux-gnu

  Source directory:           /home/john/Desktop/GRASS/grass-6.4.svn_src_snapsh
ot_2011_10_08
  Build directory:            /home/john/Desktop/GRASS/grass-6.4.svn_src_snapsh
ot_2011_10_08
  Installation directory:     ${prefix}/grass-6.4.2svn
  Startup script in directory: ${exec_prefix}/bin
  C compiler:                 gcc -g -O2
  C++ compiler:
  Building shared libraries:  yes
  64bit support:              no
  OpenGL platform:            X11

  MacOSX application:        no
  MacOSX architectures:
  MacOSX SDK:

  Tcl/Tk NVIZ:               yes

  BLAS support:              no
  C++ support:               no
  Cairo support:             no
  DWG support:               no
  FFMPEG support:            no
  FFTW support:              yes
  FreeType support:          no
  GDAL support:              yes
  GEOS support:              no
  GLw support:               no
  JPEG support:              yes
  LAPACK support:            no
  Large File support (LFS):  no
  Motif support:             no
  MySQL support:             no
  NLS support:               no
  ODBC support:              no
  OGR support:               yes
  OpenGL support:            yes
  PNG support:               yes
  PostgreSQL support:        no
  Python support:            no
  Readline support:          no
  SQLite support:            no
  Tcl/Tk support:            yes
  wxWidgets support:         no
  TIFF support:              yes
  X11 support:               yes

[john@localhost grass-6.4.svn_src_snapshot_2011_10_08]$
```

Figure 2 GRASS ./configure output


john@localhost:~/Desktop/GRASS/grass-6.4.svn_src_snapshot_2011_10_08
File Edit View Terminal Tabs Help

```
ASS/grass-6.4.svn_src_snapshot_2011_10_08/dist.i686-pc-linux-gnu/man/man1/wxGUI.
Vector_Digitizer.1
make[1]: Leaving directory `/home/john/Desktop/GRASS/grass-6.4.svn_src_snapshot_
2011_10_08/man'
make[1]: Entering directory `/home/john/Desktop/GRASS/grass-6.4.svn_src_snapshot
_2011_10_08/macosx'
Makefile:116: warning: overriding commands for target `clean'
../include/Make/Rules.make:80: warning: ignoring old commands for target `clean'
make[1]: Leaving directory `/home/john/Desktop/GRASS/grass-6.4.svn_src_snapshot_
2011_10_08/macosx'
make[1]: Entering directory `/home/john/Desktop/GRASS/grass-6.4.svn_src_snapshot
_2011_10_08/gem'
gcc at_exit_funcs.o error.o tools.o reg_deps.o reg_entries.o reg_html.o actions.
o main.o -o gem64 -g -Wall -pedantic -Werror-implicit-function-declaration -fno-
common
make[1]: Leaving directory `/home/john/Desktop/GRASS/grass-6.4.svn_src_snapshot_
2011_10_08/gem'
make manifests
make[1]: Entering directory `/home/john/Desktop/GRASS/grass-6.4.svn_src_snapshot
_2011_10_08'
make[1]: Nothing to be done for `manifests'.
make[1]: Leaving directory `/home/john/Desktop/GRASS/grass-6.4.svn_src_snapshot_
2011_10_08'
if [ 0 -eq 1 ] ; then make -C locale; fi
cp -f AUTHORS COPYING CHANGES REQUIREMENTS.html GPL.TXT contributors.csv  contri
butors_extra.csv  translators.csv /home/john/Desktop/GRASS/grass-6.4.svn_src_sna
pshot_2011_10_08/dist.i686-pc-linux-gnu/
cp -f /home/john/Desktop/GRASS/grass-6.4.svn_src_snapshot_2011_10_08/bin.i686-pc
-linux-gnu/grass64 /home/john/Desktop/GRASS/grass-6.4.svn_src_snapshot_2011_10_0
8/dist.i686-pc-linux-gnu/grass64.tmp
/usr/bin/install -c  tools/mkhtml.sh /home/john/Desktop/GRASS/grass-6.4.svn_src_
snapshot_2011_10_08/dist.i686-pc-linux-gnu/tools/ 2>/dev/null
/usr/bin/install -c  tools/g.html2man/g.html2man /home/john/Desktop/GRASS/grass-
6.4.svn_src_snapshot_2011_10_08/dist.i686-pc-linux-gnu/tools/g.html2man/ 2>/dev/
null
build_html_index.sh
Generating HTML manual pages index (help system)...
Generated HTML docs in ../dist.i686-pc-linux-gnu/docs/html/index.html
----------------------------------------------------------------------
Following modules are missing the 'description.html' file in src code:
----------------------------------------------------------------------
GRASS GIS compilation log
-------------------------
Started compilation: Thu Nov 17 17:57:04 EST 2011
--
Errors in:
No errors detected.
--
Finished compilation: Thu Nov 17 18:12:07 EST 2011
[john@localhost grass-6.4.svn_src_snapshot_2011_10_08]$
```

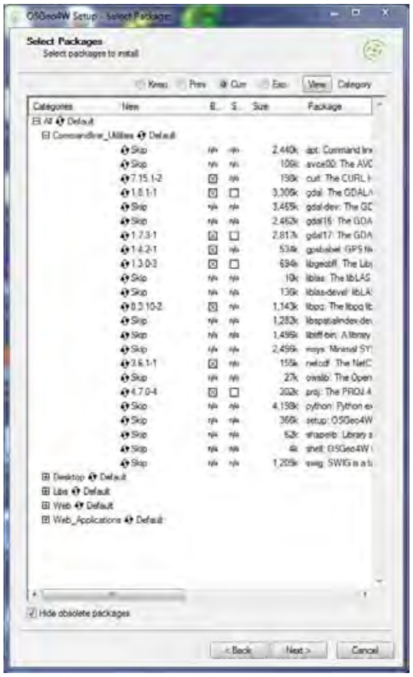
Figure 3 GRASS make output

Microsoft Windows Environment

Just as I built GRASS from source in my Linux environment, I built GRASS from source in my Windows7 environment. I followed the steps outlined on the GRASS Wiki (http://trac.osgeo.org/grass/wiki/CompileOnWindows) using the OSGeo4W installer (http://trac.osgeo.org/osgeo4w/). OSGeo4W distributes binaries of open source geospatial software for Windows. The OSGeo4W installer will create a "MSYS" command interpreter (http://www.mingw.org/wiki/MSYS). MSYS facilitates building applications that rely on GNU utilities. After setting up the "MSYS" environment I installed additional tools from the "Minimalist GNU for Windows" (MinGW) site (http://www.mingw.org/).

OSGeo4W

I chose the packages in *Table 1 OSGeo4W Packages* on the OSGeo4W Select Package screen (*Figure 4 OSGeo4W Package Selection*).


OSGeo4W Setup - Select Packages
Select Packages
Select packages to install
○ Keep ○ Prev ● Curr ○ Exp View Category
Hide obsolete packages
< Back
Next >
Cancel

```
Categories            New          B...  S...  Size    Package
⊟ All ♻ Default
  ⊟ Commandline_Utilities ♻ Default
                      ♻ Skip        n/a   n/a   2,440k  apt: Command line
                      ♻ Skip        n/a   n/a     109k  avce00: The AVC
                      ♻ 7.15.1-2    ☒     n/a     198k  curl: The CURL l
                      ♻ 1.8.1-1     ☒     □     3,306k  gdal: The GDAL/
                      ♻ Skip        n/a   n/a   3,465k  gdal-dev: The GD
                      ♻ Skip        n/a   n/a   2,462k  gdal16: The GDA
                      ♻ 1.7.3-1     ☒     □     2,817k  gdal17: The GDA
                      ♻ 1.4.2-1     ☒     n/a     534k  gpsbabel: GPS fil
                      ♻ 1.3.0-3     ☒     □       694k  libgeotiff: The Lib
                      ♻ Skip        n/a   n/a      10k  liblas: The libLAS
                      ♻ Skip        n/a   n/a     136k  liblas-devel: libLA
                      ♻ 8.3.10-2    ☒     n/a   1,143k  libpq: The libpq lib
                      ♻ Skip        n/a   n/a   1,282k  libspatialindex-dev
                      ♻ Skip        n/a   n/a   1,495k  libtiff-bin: A library
                      ♻ Skip        n/a   n/a   2,495k  msys: Minimal SY
                      ♻ 3.6.1-1     ☒     n/a     158k  netcdf: The NetC
                      ♻ Skip        n/a   n/a      27k  owslib: The Open
                      ♻ 4.7.0-4     ☒     □       202k  proj: The PROJ.4
                      ♻ Skip        n/a   n/a   4,198k  python: Python ex
                      ♻ Skip        n/a   n/a     366k  setup: OSGeo4W
                      ♻ Skip        n/a   n/a      52k  shapelib: Library a
                      ♻ Skip        n/a   n/a       4k  shell: OSGeo4W
                      ♻ Skip        n/a   n/a   1,205k  swig: SWIG is a to
  ⊞ Desktop ♻ Default
  ⊞ Libs ♻ Default
  ⊞ Web ♻ Default
  ⊞ Web_Applications ♻ Default
☑ Hide obsolete packages
```

Figure 4 OSGeo4W Package Selection

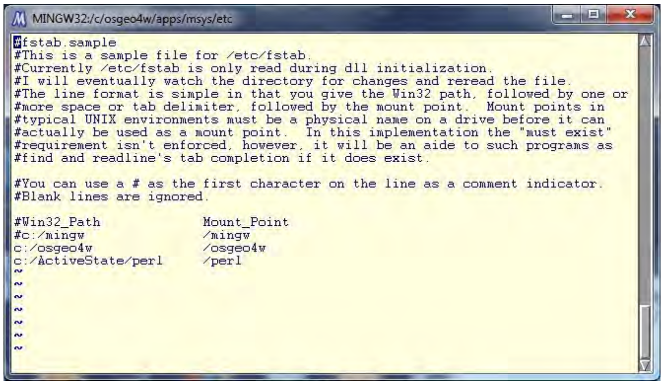
Table 1 OSGeo4W Packages

Desktop			
GRASS 6.4-dev			
Commandline_Utilities			
gpsbabel	gdal	proj	
Libs			
fftw-devel	freetype-devel	freetype-devel	gsl-libs
gsl-devel	libpng-devel	libpng-devel-mingw	libtiff-devel
libxdr	libxml2	pdcurses-devel	tcltk-devel
zlib-devel	gdal-ecw	gdal-mrsid	libtiff (4.0.0dev-90)

Minimalist GNU for Windows (MinGW)

The OSGeo4W installer created an icon for the MSYS command interpreter (*Figure 5 MSYS Desktop Icon*). In the MSYS command interpreter I installed additional packages for the MinGW environment. I first edited c:\osgeo4w\apps\msys\etc\fstab to make the Windows directory c:\osgeo4w accessible from the MSYS command interpreter (*Figure 6 MSYS Environment fstab File*). *Table 2 MinGW Packages* contains a list of the packages. The GRASS Compile on Windows Wiki page contains the download links. I uncompressed the packages as shown in *Figure 7 Uncompressing MinGW Packages* and *Figure 8 Uncompressing MinGW Make Package*.

Figure 5 MSYS Desktop Icon

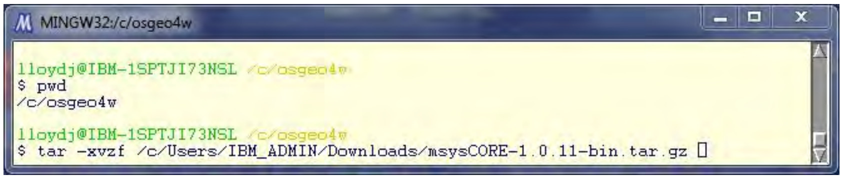


```
#fstab.sample
#This is a sample file for /etc/fstab.
#Currently /etc/fstab is only read during dll initialization.
#I will eventually watch the directory for changes and reread the file.
#The line format is simple in that you give the Win32 path, followed by one or
#more space or tab delimiter, followed by the mount point.  Mount points in
#typical UNIX environments must be a physical name on a drive before it can
#actually be used as a mount point.  In this implementation the "must exist"
#requirement isn't enforced, however, it will be an aide to such programs as
#find and readline's tab completion if it does exist.

#You can use a # as the first character on the line as a comment indicator.
#Blank lines are ignored.

#Win32_Path              Mount_Point
#c:/mingw                /mingw
c:/osgeo4w               /osgeo4w
c:/ActiveState/perl      /perl
~
~
~
~
~
~
~
```

Figure 6 MSYS Environment fstab File

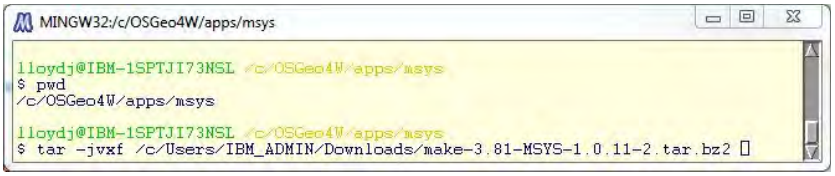
Table 2 MinGW Packages

msysCORE-1.0.11-bin	bin-utils	gcc-core
gcc-g++	mingw32-make	mingw-runtime
mingw-utils	w32api	make



```
lloydj@IBM-1SPTJI73NSL /c/osgeo4w
$ pwd
/c/osgeo4w

lloydj@IBM-1SPTJI73NSL /c/osgeo4w
$ tar -xvzf /c/Users/IBM_ADMIN/Downloads/msysCORE-1.0.11-bin.tar.gz
```

Figure 7 Uncompressing MinGW Packages



```
lloydj@IBM-1SPTJI73NSL /c/OSGeo4W/apps/msys
$ pwd
/c/OSGeo4W/apps/msys

lloydj@IBM-1SPTJI73NSL /c/OSGeo4W/apps/msys
$ tar -jvxf /c/Users/IBM_ADMIN/Downloads/make-3.81-MSYS-1.0.11-2.tar.bz2
```

Figure 8 Uncompressing MinGW Make Package

Pre-Built Binaries

After restarting MSYS I downloaded and uncompressed the pre-built binaries in *Table 3 Pre-Built Binaries* to c:\osgeo4w. The GRASS Wiki Compile on Windows page contains the download links.

Table 3 Pre-Built Binaries

bison	bison-deps	flex	gettext-bin
gettext-lib	libiconv-lib	libintl-bin	libintl-lib
regex	bc	readline	

Compiling GRASS

I downloaded the same GRASS source that I used in my Linux environment, GRASS 6.4.2RC source snapshot of 2011-10-08. I uncompressed it into the directory c:\OSGeo4W\usr\src. Running the command below from the MSYS command interpreter in the source directory will build GRASS.

- ./mswindows/osgeo4w/package.sh

Running GRASS

After compiling I could run GRASS from the MSYS command interpreter using the command below.

- /osgeo4w/bin/grass64

I could also run GRASS from a DOS command prompt using the following command. I modified this command to set environment variables needed later. This is the method I used to start GRASS through the remainder of the project.

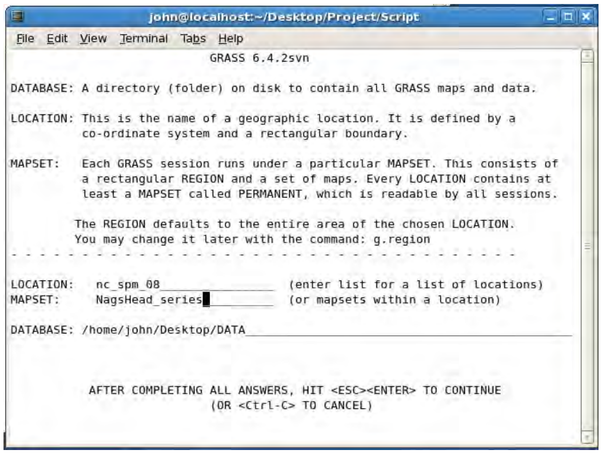
- c:\OSGeo4W\bin\grass64.bat

Running Python Scripts

Linux Environment

I ran the python script in my CentOS Linux 5.7 environment by starting a Linux shell and entering the commands below.

- "grass64 –text" (set the location, mapset and database as in *Figure 9 Setting Location, Mapset, and Database*)
- cd <directory containing python script>
- python <script name>.py <parameters>


john@localhost:~/Desktop/Project/Script
File Edit View Terminal Tabs Help

```
                              GRASS 6.4.2svn

DATABASE: A directory (folder) on disk to contain all GRASS maps and data.

LOCATION: This is the name of a geographic location. It is defined by a
          co-ordinate system and a rectangular boundary.

MAPSET:   Each GRASS session runs under a particular MAPSET. This consists of
          a rectangular REGION and a set of maps. Every LOCATION contains at
          least a MAPSET called PERMANENT, which is readable by all sessions.

         The REGION defaults to the entire area of the chosen LOCATION.
         You may change it later with the command: g.region
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

LOCATION:   nc_spm_08________________   (enter list for a list of locations)
MAPSET:     NagsHead_series__________   (or mapsets within a location)

DATABASE: /home/john/Desktop/DATA_____________________________________________


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

Figure 9 Setting Location, Mapset, and Database

Microsoft Windows Environment

From a command prompt in Windows I ran a python script by following the steps below.

- "\OSGeo4W\bin\grass64.bat –text" (set the location, mapset, and database)
- cd <directory containing python script>
- python <script name>.py <parameters>

v.transects Test Cases

This section lists the test cases run in the two environments. There are four sets of test cases. The first set verifies that the script handles incorrect parameters. The second set verifies that the script works on shoreline data. The third set verifies that the script handles polylines of various shapes. The datasets in the first three test case sets are all in NC State Plane Meters. The last set verifies that the script works with data not in State Plane but in WGS84.

The tables in each section contain a column describing the purpose of the test, the values of the five parameters to the script, and a column indicating the final result after the test had been executed in all environments. I executed the tests following the process detailed in the Test Execution section below.

Parameter Verification Test Set

This group of tests verifies that the script detects and correctly handles incorrect parameter inputs.

The parameters to the script are

- input - the name of a line vector layer
- output - the name a line or polygon vector layer
- transect_spacing - a floating point number in the units of the projection representing the distance between transects
- dleft - a floating point number in the units of the projection representing the distance the transect extends on the left side of the line
- dright - a floating point number in the units of the projection representing the distance the transect extends on the right side of the line
- type - line or area specifying the type of layer created

Table 4 Parameter Tests

		Parameters						
#	Description	Input	Output	Transect _spacing	Dleft	Dright	Type	Pass/fail
1	Input parameter required		NH_2008_ transcect	50				Pass - displays error message
2	Output parameter required	NH_2008_1m		50				Pass - displays error message
3	Transect_spacing parameter required	NH_2008_1m	NH_2008_ transect					Pass - displays error message
4	Transect parameter number	NH_2008_1m	NH_2008_ transect	Xxx				Fail – pass after update to display error message
5	Dleft parameter number	NH_2008_1m	NH_2008_ transect	50	Xxx			Fail – pass after update to display error message
6	Dright parameter number	NH_2008_1m	NH_2008_ transect	50		Xxx		Fail – pass after update to display error message
7	Input does not exist	Xxx	NH_2008_ transect	50				Pass - displays error message
8	Output exists	NH_2008_1m	NH_2008_ transect	50				Pass - displays error message
9	Invalid Option -							Pass - displays error message
10	Invalid Type Option	NH_2008_1m	NH_2008_ transect	50			Xxx	Pass - displays error message

Shoreline Dataset Verification Test Set

This group of test verifies that the script performs its designed purpose of creating transects along lines representing the shoreline. The dataset is in NC State Plane Meters.

Table 5 Shoreline Dataset Tests

		Parameters						
#	Description	Input	Output	Transect _spacing	Dleft	Dright	Type	Pass/fail
1	Parameters used in paper	NH_2008_1m	NH_2008_ transcect1	50				Pass
2	Large transect spacing	NH_2008_1m	NH_2008_ transcect2	500				Pass
3	Zero transect spacing	NH_2008_1m	NH_2008_ transcect3	0				Fail – pass after update to display error message
4	One meter transect spacing	NH_2008_1m	NH_2008_ transcect4	1				Pass
5	Left/right transect spacing different	NH_2008_1m	NH_2008_ transcect5	50	25	100		Pass
6	Zero transect spacing on left	NH_2008_1m	NH_2008_ transcect6	50	0			Pass
7	Zero transect spacing on right	NH_2008_1m	NH_2008_ transcect7	50		0		Pass
8	Parameters used in paper with area	NH_2008_1m	NH_2008_ transcect8	50			area	Pass
9	Parameters used in paper with line	NH_2008_1m	NH_2008_ transcect9	50			line	Pass

Verify Polylines of Various Shapes Test Set

This group of test verifies that the script works correctly with polyline features in shapes unlike those of a shoreline. I created the layers used in this test set with the ArcMap Edit Tool as shapefiles and then imported them into GRASS. The datasets are all in NC State Plane Meters.

Table 6 Various Shapes Tests

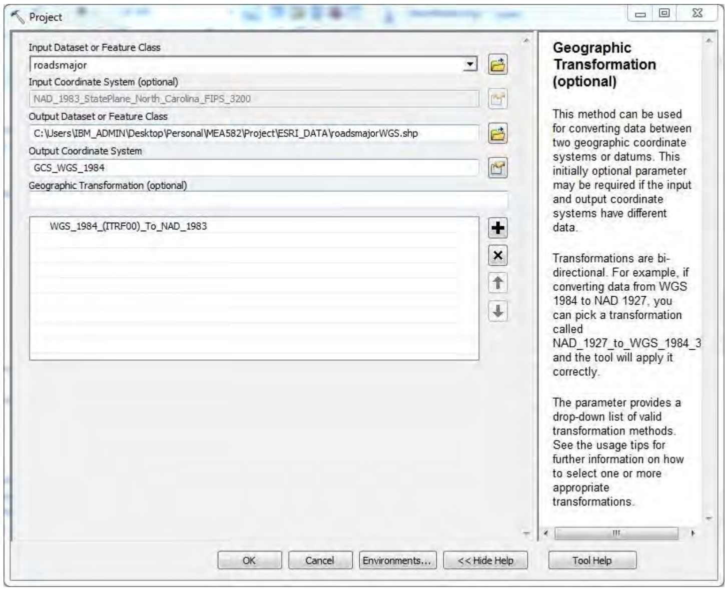
		Parameters						
#	Description	Input	Output	Transect _spacing	Dleft	Dright	Type	Pass/fail
1	No lines in input	noLines	noLines_tr ansects	50				Fail – pass after added error message
2	Multiple lines in input	multLines	multLines_ transects	50				Pass
3	Circle in input	circle	circle_trans ects	50				Pass
4	Right angle in input	rightAngle	rightAngle_	50				Pass

			transects					
5	Intersecting lines in input	interLines	interLines_transects	50				Pass
6	Network of Lines (streets)	streets_wake	Streets_wake_transects	50				Fail – but passed with updates in Linux only
7	Super long line	superLong	superLong_transects	50				Pass
8	Freehand	freeHand	freeHand_transects	50				Pass
9	Points	Points	Points_transects	50				Pass - displays error message
10	Raster	Raster	Raster_transects	50				Pass - display error message
11	Multiple lines in input with area	multLines	multLines_area	50			Area	Pass
12	Multiple lines in input with line	multLines	multLines_line	50			Line	Pass

WGS84 Test Set

The datasets in the previous tests all used the NC State Plane Meters projection. To determine whether the script worked with other projections, I re-projected some of the polyline shapefiles I created in ArcMap to WGS84 using the ArcMap tool (*Figure 10 Re-projecting Dataset to WGS84*). I then imported the re-projected datasets into GRASS after creating a “Location” using the WGS84 projection (*Figure 11 Creating WGS84 Location*).

There may not be a requirement for the script to handle datasets in the WGS84 projection. It is also unclear what the units for the transect_spacing, dleft, and dright parameters would be, although most likely they will be in degrees with the current implementation. However, specifying spacing in degrees is not convenient.



Input Dataset or Feature Class
roadsmajor
Input Coordinate System (optional)
NAD_1983_StatePlane_North_Carolina_FIPS_3200
Output Dataset or Feature Class
C:\Users\IBM_ADMIN\Desktop\Personal\MEA582\Project\ESRI_DATA\roadsmajorWGS.shp
Output Coordinate System
GCS_WGS_1984
Geographic Transformation (optional)
WGS_1984_(ITRF00)_To_NAD_1983

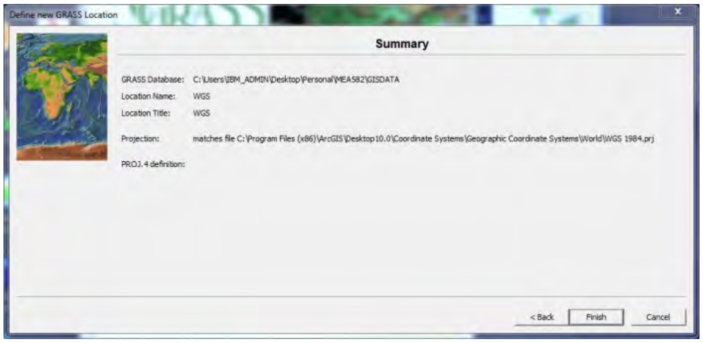
Geographic Transformation (optional)

This method can be used for converting data between two geographic coordinate systems or datums. This initially optional parameter may be required if the input and output coordinate systems have different data.

Transformations are bi-directional. For example, if converting data from WGS 1984 to NAD 1927, you can pick a transformation called NAD_1927_to_WGS_1984_3 and the tool will apply it correctly.

The parameter provides a drop-down list of valid transformation methods. See the usage tips for further information on how to select one or more appropriate transformations.

Figure 10 Re-projecting Dataset to WGS84



Summary

GRASS Database: C:\Users\IBM_ADMIN\Desktop\Personal\MEA582\GISDATA
Location Name: WGS
Location Title: WGS
Projection: matches file C:\Program Files (x86)\ArcGIS\Desktop10.0\Coordinate Systems\Geographic Coordinate Systems\World\WGS 1984.prj
PROJ.4 definition:

Figure 11 Creating WGS84 Location

Table 7 WGS84 Projection Tests

		Parameters						
#	Description	Input	Output	Transect _spacing	Dleft	Dright	Type	Pass/fail
1	Multiple lines in input	multLinesWGS	multLines WGS_transects	50				Fail
2	Freehand	freeHandWGS	freeHandWGS_transects	50				Fail
3	Multiple lines in input .01	multLinesWGS	multLines WGS01_transects	0.01				Pass

Test Execution

I tested the script in three phases. The first phase tested the script as provided to me in my CentOS Linux 5.7 environment. I expected the script to work in this environment as it had already been used in another UNIX based operating system, Mac OS. If I did encounter defects I fixed them when possible. I also saved all the transect layers created by exporting them to ASCII. Exporting to ASCII allowed me to use a difference tool in later phases to detect any variations in the outputs.

The second phase tested the script in my Windows7 environment. I ran the same test cases as in phase one and fixed any defects when possible. As in the first phase I saved the transect layers created by exporting them to ASCII. Then, at the conclusion of the phase I used the difference tool in the SlickEdit (http://www.slickedit.com) development product to compare the layers created. The tool allowed me to spot differences not visible in map documents.

The third phase repeated the tests again in the CentOS Linux environment to verify that changes made did not introduce defects. As in the other phases I used the difference tool to compare layers.

Phase 1- Run Original Script in Linux

In this phase I ran the script as provided against test cases in my CentOS Linux 5.7 environment. The test cases listed below failed.

Parameter Verification Failed Test Cases

Test case 4 – Verify Correct Handling of Invalid transect_spacing Parameter:

Attempting to do floating point operations on a string literal caused a run-time error. The fix to the script detected the string literal and exited gracefully.

Output Before fix applied	*Output After fix applied*

GRASS 6.4.2svn (nc_spm_08):~/Desktop/Project/Script >
python v.transects.py input=NH_2008_1m
output=NH_2008_transects transect_spacing=xxx
Traceback (most recent call last):
  File "v.transects.py", line 234, in ?
    main()
  File "v.transects.py", line 198, in main
    transect_spacing =
float(options['transect_spacing'])
ValueError: invalid literal for float(): xxx
	GRASS 6.4.2svn (nc_spm_08):~/Desktop/Project/Script >
	python v.transects.py input=NH_2008_1m
	output=NH_2008_transects transect_spacing=xxx
	ERROR: Invalid transect_spacing value.

Test case 5 – Verify Correct Handling of Invalid dleft Parameter:

This test case caused a failure similar to the previous test case and a similar fix was applied to the script.

Output Before fix applied	*Output After fix applied*

GRASS 6.4.2svn (nc_spm_08):~/Desktop/Project/Script >
python v.transects.py input=NH_2008_1m
output=NH_2008_transects transect_spacing=52
dleft=xxx
xxx 52.0
Traceback (most recent call last):
  File "v.transects.py", line 234, in ?
    main()
  File "v.transects.py", line 206, in main
    dleft = float(dleft)
ValueError: invalid literal for float(): xxx
	GRASS 6.4.2svn (nc_spm_08):~/Desktop/Project/Script >
	python v.transects.py input=NH_2008_1m
	output=NH_2008_transects transect_spacing=52
	dleft=xxx
	xxx 52.0
	ERROR: Invalid dleft value.

Test case 6 – Verify Correct Handling of dright Parameter:

This is another example of run-time error due to floating point operations on a string literal.

Output Before fix applied	*Output After fix applied*
GRASS 6.4.2svn (nc_spm_08):~/Desktop/Project/Script > python v.transects.py input=NH_2008_1m output=NH_2008_transects transect_spacing=52 dright=xxx 52.0 Traceback (most recent call last): File "v.transects.py", line 234, in ?	GRASS 6.4.2svn (nc_spm_08):~/Desktop/Project/Script > python v.transects.py input=NH_2008_1m output=NH_2008_transects transect_spacing=52 dright=xxx 52.0 ERROR: Invalid dright value.

`    main()`
`  File "v.transects.py", line 210, in main`
`    dright = float(dright)`
`ValueError: invalid literal for float(): xxx`	

Shoreline Dataset Failed Test Cases

Test Case 3 – Verify Correct Handling of zero transect_spacing

When I set the transect_spacing parameter to zero, I had to use a keyboard interrupt (CTRL-C) to stop the script. The transect spacing value of zero caused the script to enter a loop with no exit or infinite loop. I applied a fix to the script that made zero an invalid transect spacing value.

Output Before fix applied	*Output After fix applied*

`GRASS 6.4.2svn (nc_spm_08):~/Desktop/Project/Script/Orig > python v.transects.py input=NH_2008_1m output=NH_2008_transects3 transect_spacing=0`
` 0.0`
`Traceback (most recent call last):`
`  File "v.transects.py", line 220, in ?`
`    main()`
`  File "v.transects.py", line 212, in main`
`    transect_ends = getTransects( v, dleft, dright, transect_spacing )`
`  File "v.transects.py", line 117, in getTransects`
`    transect_locs[-1].append(  (r*array( line[j] ) + (1-r)*array( line[j-1] )).tolist()  )`
`KeyboardInterrupt`
	`GRASS 6.4.2svn (nc_spm_08):~/Desktop/Project/Script > python v.transects.py input=NH_2008_1m output=NH_2008_transects3 transect_spacing=0`
	`ERROR: Zero invalid transect_spacing value.`

Verify Polylines of Various Shapes Failed Test Cases

Test Case 6 – Verify Correct Handling of a Network of Lines

When I ran the script with the input layer set to a network of streets, the result was a run-time error writing the result.

Output Before fix applied	*Output After fix applied*

`GRASS 6.4.2svn (nc_spm_08):~/Desktop/Project/Script/Orig > python v.transects.py input=streets_wake output=streets_wake_transects transect_spacing=50`
` 50.0`
`Building topology for vector map <streets_wake_transects>...`
`Registering primitives...`
`197302 primitives registered`
`394604 vertices registered`
`Building areas...`
` 100%`
`0 areas built`
`0 isles built`
`Attaching islands...`
`Attaching centroids...`
` 100%`
`Number of nodes: 394604`
`Number of primitives: 197302`
`Number of points: 0`
`Number of lines: 197302`
`Number of boundaries: 0`
`Number of centroids: 0`
`Number of areas: 0`
`Number of isles: 0`
`WARNING: Unable to write lines to plus file`
`WARNING: Error writing out topo file`
`WARNING: Error writing out category index file`
	No fix applied

WGS Dataset Failed Test Cases

Test Case 1 – Verify Handling of Multiple Lines in WGS84

When using a layer in the WGS84 projection, the script exited but no features were created.

<table>
<tr><th><em>Output Before fix applied</em></th><th><em>Output After fix applied</em></th></tr>
<tr><td><pre>
GRASS 6.4.2svn (WGS):~/Desktop/Project/Script/Orig > python v.transects.py
input=multLinesWGS output=multLinesWGS_transects transect_spacing=50
 50.0
Building topology for vector map <multLinesWGS_transects>...
Registering primitives...
0 primitives registered
0 vertices registered
Building areas...
0 areas built
0 isles built
Attaching islands...
Attaching centroids...
Number of nodes: 0
Number of primitives: 0
Number of points: 0
Number of lines: 0
Number of boundaries: 0
Number of centroids: 0
Number of areas: 0
Number of isles: 0
v.in.ascii complete.
</pre></td><td>No fix applied</td></tr>
</table>

Test Case 2 – Verify Handling of Free Hand Line in WGS84

No features were created.

<table>
<tr><th><em>Output Before fix applied</em></th><th><em>Output After fix applied</em></th></tr>
<tr><td><pre>
GRASS 6.4.2svn (WGS):~/Desktop/Project/Script/Orig > python v.transects.py
input=freeHandWGS output=freeHandWGS_transects transect_spacing=50
 50.0
Building topology for vector map <freeHandWGS_transects>...
Registering primitives...
0 primitives registered
0 vertices registered
Building areas...
0 areas built
0 isles built
Attaching islands...
Attaching centroids...
Number of nodes: 0
Number of primitives: 0
Number of points: 0
Number of lines: 0
Number of boundaries: 0
Number of centroids: 0
Number of areas: 0
Number of isles: 0
v.in.ascii complete.
</pre></td><td>No fix applied</td></tr>
</table>

Phase 2 - Run Script with fixes applied in Windows

In the second phase of my testing I reran the test cases in my Microsoft Window7 environment using the script provided and any fixes applied in the first phase.

Before I could execute the test cases in my Windows environment, I had to fix two run-time errors. The first error was caused by parameters to the python function popen() that are not supported on

Windows (http://docs.python.org/library/subprocess.html). The second error was caused by re-opening a named temporary file, which is also not supported on Windows with the system call used in the original script (http://docs.python.org/library/tempfile.html).

The test cases listed below failed.

Error Messages Caused by Popen	Error Messages Caused by NamedTemporaryFile
GRASS 6.4.2svn (nc_spm_08)> python v.transects.py input=NH_2008_1m output=N H_2008_transects1 transect_spacing=50 50.0 Traceback (most recent call last): File "v.transects.orig.py", line 220, in <module> main() File "v.transects.orig.py", line 211, in main v = loadVector(vector) File "v.transects.orig.py", line 70, in loadVector p = Popen(expVecCmmd, shell=True, stdin=PIPE, stdout=PIPE, stderr=STDOUT, cl ose_fds=True) File "C:\OSGeo4W\apps\Python25\lib\subprocess.py", line 552, in __init__ raise ValueError("close_fds is not supported on Windows " ValueError: close_fds is not supported on Windows platforms	GRASS 6.4.2svn (nc_spm_08)> python v.transects.orig.py input=NH_2008_1m output=N H_2008_transects1 transect_spacing=50 50.0 ERROR: Unable to open ASCII file

Verify Polylines of Various Shapes Failed Test Cases

Test Case 6 – Verify Correct Handling of a Network of Lines

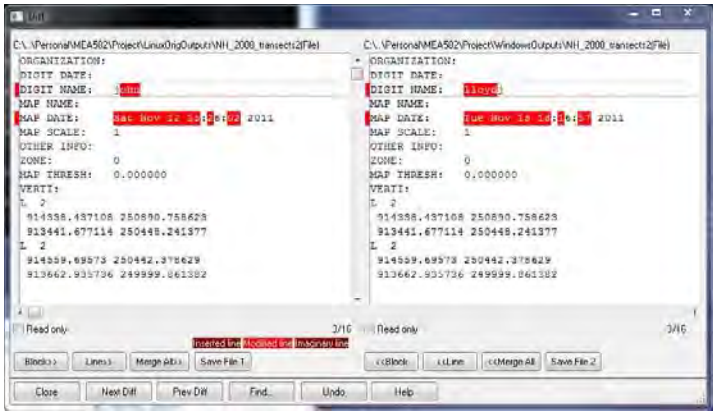
When I ran the script with an input dataset of a layer containing a network of streets, the script failed to terminate after two hours of execution. This is the sign of an infinite loop.

WGS Dataset Failed Test Cases

The results of the test cases in this group were the same as running in the Linux environment. The script did not create any transects in the first two test cases.

Automate Diff ASCII versions of Windows and Linux

I exported the transect layers created by the test cases to ASCII format and then used a text difference tool to compare the ASCII versions produced in the Linux environment with those produced in the Windows environment. Every layer contained differences in the header information (*Figure 12 Differences in Header Info*). There were also differences in the values for coordinates of 0.000001m (*Figure 13 Differences in Coordinates*). Neither difference was significant.



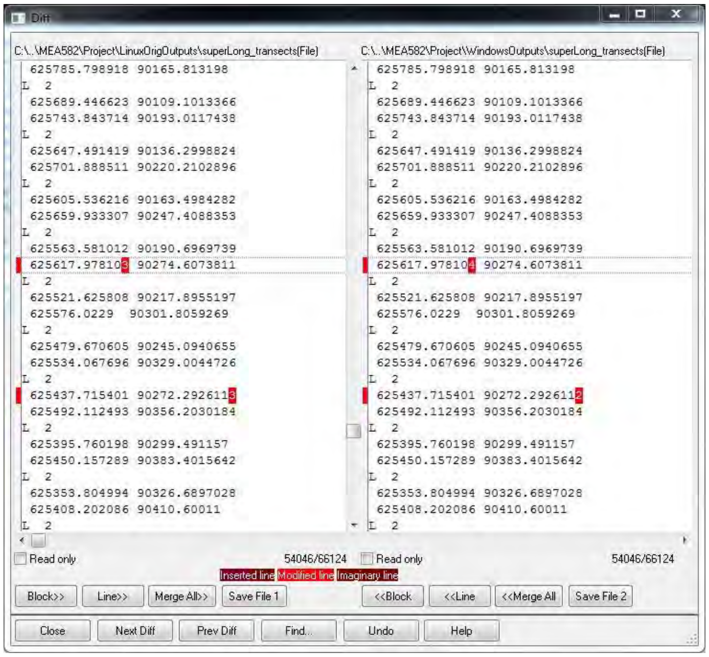
C:\..\Personal\MEA582\Project\LinuxOrigOutputs\NH_2000_transects2(File)

```
ORGANIZATION:
DIGIT DATE:
DIGIT NAME:    john
MAP NAME:
MAP DATE:      Sat Nov 12 13:26:02 2011
MAP SCALE:     1
OTHER INFO:
ZONE:          0
MAP THRESH:    0.000000
VERTI:
L  2
 914338.437108 250890.758623
 913441.677114 250448.241377
L  2
 914559.69573 250442.378629
 913662.935736 249999.861382
```

C:\..\Personal\MEA582\Project\WindowsOutputs\NH_2000_transects2(File)

```
ORGANIZATION:
DIGIT DATE:
DIGIT NAME:    lloyd
MAP NAME:
MAP DATE:      Tue Nov 15 10:16:37 2011
MAP SCALE:     1
OTHER INFO:
ZONE:          0
MAP THRESH:    0.000000
VERTI:
L  2
 914338.437108 250890.758623
 913441.677114 250448.241377
L  2
 914559.69573 250442.378629
 913662.935736 249999.861382
```

Figure 12 Differences in Header Info



C:\..\MEA582\Project\LinuxOrigOutputs\superLong_transects(File)

```
 625785.798918 90165.813198
L  2
 625689.446623 90109.1013366
 625743.843714 90193.0117438
L  2
 625647.491419 90136.2998824
 625701.888511 90220.2102896
L  2
 625605.536216 90163.4984282
 625659.933307 90247.4088353
L  2
 625563.581012 90190.6969739
 625617.978103 90274.6073811
L  2
 625521.625808 90217.8955197
 625576.0229  90301.8059269
L  2
 625479.670605 90245.0940655
 625534.067696 90329.0044726
L  2
 625437.715401 90272.2926113
 625492.112493 90356.2030184
L  2
 625395.760198 90299.491157
 625450.157289 90383.4015642
L  2
 625353.804994 90326.6897028
 625408.202086 90410.60011
L  2
```

C:\..\MEA582\Project\WindowsOutputs\superLong_transects(File)

```
 625785.798918 90165.813198
L  2
 625689.446623 90109.1013366
 625743.843714 90193.0117438
L  2
 625647.491419 90136.2998824
 625701.888511 90220.2102896
L  2
 625605.536216 90163.4984282
 625659.933307 90247.4088353
L  2
 625563.581012 90190.6969739
 625617.978104 90274.6073811
L  2
 625521.625808 90217.8955197
 625576.0229  90301.8059269
L  2
 625479.670605 90245.0940655
 625534.067696 90329.0044726
L  2
 625437.715401 90272.2926112
 625492.112493 90356.2030184
L  2
 625395.760198 90299.491157
 625450.157289 90383.4015642
L  2
 625353.804994 90326.6897028
 625408.202086 90410.60011
L  2
```

Figure 13 Differences in Coordinates

Phase 3 - Run Script with fixes applied in Linux

I re-ran the script in Linux. This final phase verified that the changes made to accommodate Windows did not introduce defects in the Linux environment. I did not discover any new defects in this phase. However, the test case using a network of streets for the input dataset produced correct output in this phase when it failed in the other phases (*Figure 14 Network of Lines Test Case 6 in Linux*).

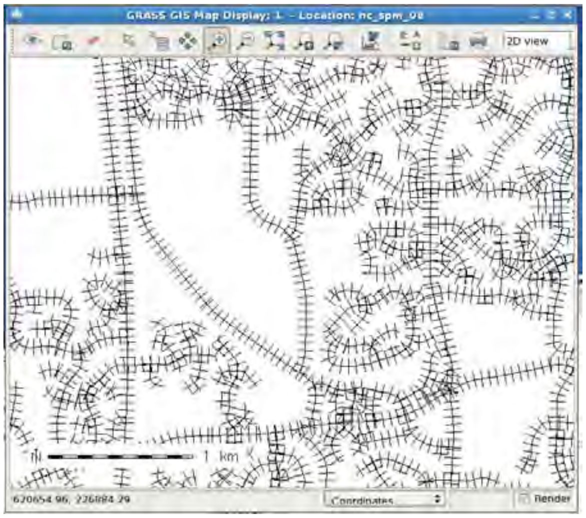


Figure 14 Network of Lines Test Case 6 in Linux

Discussion

GRASS from Source

Installing GRASS from source code rather than using the pre-built binaries can be tedious. If the prerequisite packages were not included during the Linux installation then missing packages must be identified, downloaded, and installed before building GRASS. The easiest method for identifying missing packages is by running the GRASS configure script included with the source. Identified packages can then be downloaded and installed with a Linux tool such as "yum".

Another problem installing from source is setting the environment variables. This was particularly an issue in the Windows environment. When using the prebuilt binaries the installer sets the variables and registry settings. However when installing from source, I had to determine the required variables and then modify the GRASS start script to set them to the correct values.

While installing GRASS from source can be tedious, there are several advantages. When installing from source the developer is not restricted to the levels of the pre-built binaries for his environment. This makes it possible to use the latest updates and fixes. Using source also helps understand the algorithms implemented. If a result looks incorrect the developer can review the source to verify that the algorithm is as expected. If the algorithm is not as expected or not appropriate for the current application then the developer can modify the source to suit. Finally, using the source code the developer can debug defects in the tools.

v.transect Test Results

The script as provided did minimal parameter checking. It was able to detect missing parameters and unused extraneous parameters. I added code to handle the wrong type of input (i.e. string instead of numeric). I also added code to detect the wrong type of vector input (points or polygon instead of lines). I used the GRASS v.info command from within the script to determine if the input was a line vector layer that contained at least one line.

The test case that used a network of streets as the input layer failed in the Linux and Windows environments with the script as provided. However, the final updated script generated transects with this input in the Linux environment even though I did not make an explicit change to fix the defect. Since the error message generated by the original script in Linux shows a problem writing to a file, it is possible that changing the temporary file creation system function from NamedTemporaryFile() to mkstemp() caused this side effect.

I ran the script in a GRASS location using the WGS84 projection. Although running the script in this projection is not likely a requirement, I was curious to see what would happen. The transect_spacing, dleft, and dright parameters are most likely interpreted as degrees in this GRASS location. Although degrees are not a convenient representation for transect spacing, the script should still work. The first tests in this group used 50 for the transect spacing value. When the script generated no transects, this confirmed that the parameter was not being interpreted as meters as in the NC State Plane test cases. The final test in this group used 0.01 for the transect spacing. The script then generated transects approximately 1000m apart. Given that 1/60th of a degree latitude equals one nautical mile or 1852m then 1/100 (0.01) of a degree is approximately 1000m. Therefore, the script is interpreting the transect spacing as degrees and functioning correctly in the WGS84 projection.

When I completed testing I extended the existing man page for the script. The man page can be viewed at http://www4.ncsu.edu/~jlloyd/Lloydj_MEA582f11/v_transects.html. The HTML is included in the appendix.

Conclusions

Setting up the GRASS environments from source code for both CentOS Linux 5.7 and Microsoft Windows7 was time consuming and, at times, frustrating. However, now it is complete I have a better understanding of the application's architecture. I am also prepared to continue working in GRASS development by modifying GRASS source code for defect fixes or enhancements.

However, those not wanting to enhance the GRASS distribution for their needs can still extend the GRASS functionality by writing python scripts. I have demonstrated in this project that this is possible in both Linux and Windows environments. I have also demonstrated that testing in both environments is required if the script is to be distributed publicly. Implementation differences in Linux and Windows libraries mean that a script functioning in one environment is not guaranteed to function in the other.

Appendix

GRASS Start Script for Windows

This is the modified GRASS start script for Windows, which sets the environment variables need for GRASS and python.

```
rem c:\OSGeo4W\bin\grass64.bat
@echo on
SET OSGEO4W_ROOT=C:\OSGeo4W
SET WINGISBASE=%OSGEO4W_ROOT%\apps\grass\grass-6.4.2svn
SET PYTHONPATH=%WINGISBASE%\etc\python
call %OSGEO4W_ROOT%\bin\o4w_env.bat
call %OSGEO4W_ROOT%\apps\grass\grass-6.4.2svn\etc\env.bat
"%WINGISBASE%"\etc\init.bat %*
```

Test Case Commands

I used these commands from a GRASS text mode prompt to execute the test cases.

Parameter Verification Test Cases Commands

```
# 1 Input Parameter Required
python v.transects.py output=NH_2008_transects transect_spacing=50
g.remove vect=NH_2008_transects
# 2 Output Parameter Required
python v.transects.py input=NH_2008_1m transect_spacing=50
g.remove vect=NH_2008_transects
# 3 Transect_spacing Parameter Required
python v.transects.py input=NH_2008_1m output=NH_2008_transects
g.remove vect=NH_2008_transects
# 4 Transect spacing parameter number
```

```
python v.transects.py input=NH_2008_1m output=NH_2008_transects transect_spacing=xxx
g.remove vect=NH_2008_transects
# 5 Dleft parameter number
python v.transects.py input=NH_2008_1m output=NH_2008_transects transect_spacing=52 dleft=xxx
g.remove vect=NH_2008_transects
# 6 Dright parameter number
python v.transects.py input=NH_2008_1m output=NH_2008_transects transect_spacing=52 dright=xxx
g.remove vect=NH_2008_transects
# 7 Input does not exist
python v.transects.py input=xxx output=NH_2008_transects transect_spacing=52
# g.remove vect=NH_2008_transects – don't remove
# 8 Output exists
python v.transects.py input=NH_2008_1m output=NH_2008_transects transect_spacing=52
python v.transects.py input=NH_2008_1m output=NH_2008_transects transect_spacing=52
g.remove vect=NH_2008_transects
# 9 Invalid Option
python v.transects.py xxx=NH_2008_1m output=NH_2008_transects transect_spacing=52
g.remove vect=NH_2008_transects
# 10 Invalid Type Option
python v.transects.py xxx=NH_2008_1m output=NH_2008_transects transect_spacing=52 type=xxx
g.remove vect=NH_2008_transects
```

Shoreline Dataset Test Cases Commands

```
# Remove any outputs from previous runs
g.remove vect=NH_2008_transects1
g.remove vect=NH_2008_transects2
g.remove vect=NH_2008_transects3
g.remove vect=NH_2008_transects4
g.remove vect=NH_2008_transects5
g.remove vect=NH_2008_transects6
g.remove vect=NH_2008_transects7
g.remove vect=NH_2008_transects8
g.remove vect=NH_2008_transects9

# 1 Parameters used in paper
python v.transects.py input=NH_2008_1m output=NH_2008_transects1 transect_spacing=50
# 2 Large Spacing
python v.transects.py input=NH_2008_1m output=NH_2008_transects2 transect_spacing=500
# 3 Zero Spacing
python v.transects.py input=NH_2008_1m output=NH_2008_transects3 transect_spacing=0
# 4 One Meter Spacing
python v.transects.py input=NH_2008_1m output=NH_2008_transects4 transect_spacing=1
# 5 Left/Right Different
python v.transects.py input=NH_2008_1m output=NH_2008_transects5 transect_spacing=52 dleft=25
dright=100
# 6 Zero Left
python v.transects.py input=NH_2008_1m output=NH_2008_transects6 transect_spacing=52 dleft=0
# 7 Zero Right
python v.transects.py input=NH_2008_1m output=NH_2008_transects7 transect_spacing=52 dright=0
# 8 Parameters used in paper with area
python v.transects.py input=NH_2008_1m output=NH_2008_transects8 transect_spacing=50 type=area
# 9 Parameters used in paper with line
python v.transects.py input=NH_2008_1m output=NH_2008_transects9 transect_spacing=50 type=line
```

Verify Polylines of Various Shapes Test Cases Commands

```
# Remove any outputs from previous runs
g.remove vect=noLines_transects
g.remove vect=multLines_transects
g.remove vect=circle_transects
g.remove vect=rightAngle_transects
g.remove vect=interLines_transects
g.remove vect=streets_wake_transects
g.remove vect=superLong_transects
g.remove vect=freeHand_transects
g.remove vect=multLines_area
g.remove vect=multLines_line

# 1 No lines in input
```

```
python v.transects.py input=noLines output=noLines_transects transect_spacing=50
# 2 Multiple lines in input
python v.transects.py input=multLines output=multLines_transects transect_spacing=50
# 3 Circle in input
python v.transects.py input=circle output=circle_transects transect_spacing=50
# 4 Right angle in input
python v.transects.py input=rightAngle output=rightAngle_transects transect_spacing=50
# 5 Intersecting lines in input
python v.transects.py input=interLines output=interLines_transects transect_spacing=50
# 6 Network of lines (streets)
python v.transects.py input=streets_wake output=streets_wake_transects transect_spacing=50
# 7 Superlong line in input
python v.transects.py input=superLong output=superLong_transects transect_spacing=50
# 8 Freehand in input
python v.transects.py input=freeHand output=freeHand_transects transect_spacing=50
# 9 Points in input
python v.transects.py input=points output=points_transects transect_spacing=50
# 10 Raster in input
python v.transects.py input=raster output=raster_transects transect_spacing=50
# 11 Multiple lines in input with area
python v.transects.py input=multLines output=multLines_area transect_spacing=50 type=area
# 12 Multiple lines in input with line
python v.transects.py input=multLines output=multLines_line transect_spacing=50 type=line
```

WGS Dataset Test Cases Commands

```
# Remove any outputs from previous runs
g.remove vect=multLinesWGS_transects
g.remove vect=freeHandWGS_transects
g.remove vect=multLinesWGS01_transects

# 1 Multiple lines in input
python v.transects.py input=multLinesWGS output=multLinesWGS_transects transect_spacing=50
# 2 Freehand in input
python v.transects.py input=freeHandWGS output=freeHandWGS_transects transect_spacing=50
# 3 Multiple lines in input 0.01
python v.transects.py input=multLinesWGS output=multLinesWGS01_transects transect_spacing=0.01
```

Save Resulting Transects to ASCII Files Commands

I created scripts to save the datasets created by the test cases to ASCII files. Since the NC State Plane Meters and WGS84 datasets are accessible via different GRASS locations, I needed two scripts. After running the scripts I used the SlickEdit difference tool to compare outputs. The scripts are given below.

Script to Save NC State Plane Meters Datasets Created by v.transect to ASCII Files

```
v.out.ascii input=NH_2008_transects1 output=NH_2008_transects1 format=standard
v.out.ascii input=NH_2008_transects2 output=NH_2008_transects2 format=standard
# Error message v.out.ascii input=NH_2008_transects3 output=NH_2008_transects3 format=standard
v.out.ascii input=NH_2008_transects4 output=NH_2008_transects4 format=standard
v.out.ascii input=NH_2008_transects5 output=NH_2008_transects5 format=standard
v.out.ascii input=NH_2008_transects6 output=NH_2008_transects6 format=standard
v.out.ascii input=NH_2008_transects7 output=NH_2008_transects7 format=standard
v.out.ascii input=NH_2008_transects8 output=NH_2008_transects8 format=standard
v.out.ascii input=NH_2008_transects9 output=NH_2008_transects9 format=standard
v.out.ascii input=noLines_transects output=noLines_transects format=standard
v.out.ascii input=multLines_transects output=multLines_transects format=standard
v.out.ascii input=circle_transects output=circle_transects format=standard
v.out.ascii input=rightAngle_transects output=rightAngle_transects format=standard
v.out.ascii input=interLines_transects output=interLines_transects format=standard
# failed v.out.ascii input=streets_wake_transects output=streets_wake_transects format=standard
v.out.ascii input=superLong_transects output=superLong_transects format=standard
v.out.ascii input=freeHand_transects output=freeHand_transects format=standard
v.out.ascii input=multLines_line output=multLines_lines format=standard
v.out.ascii input=multLines_area output=multLines_area format=standard
```

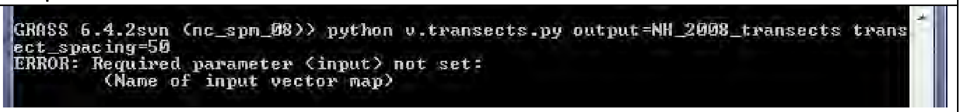
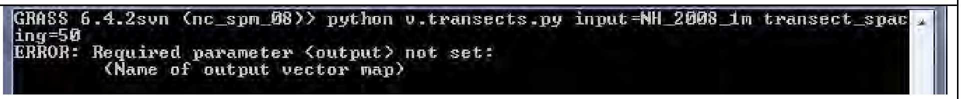
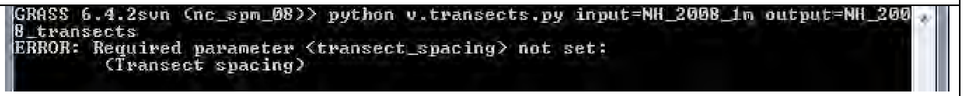
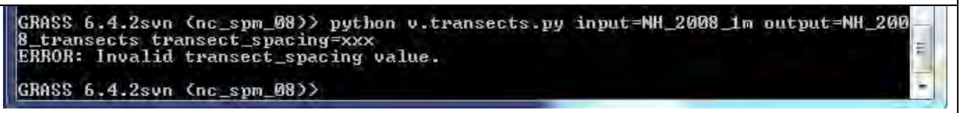
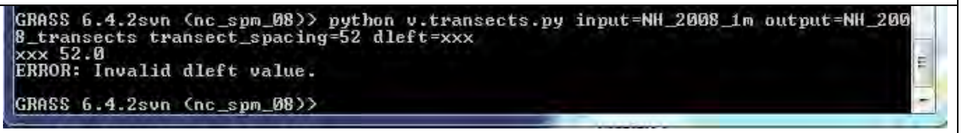
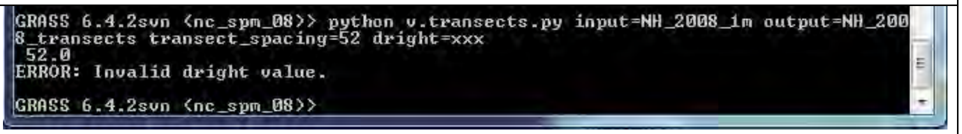
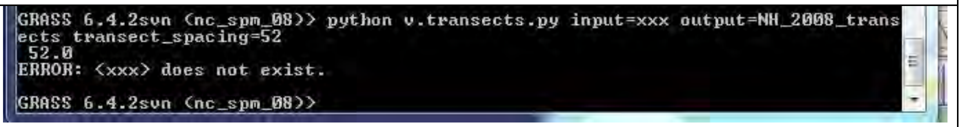
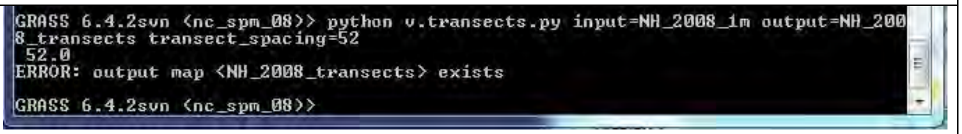
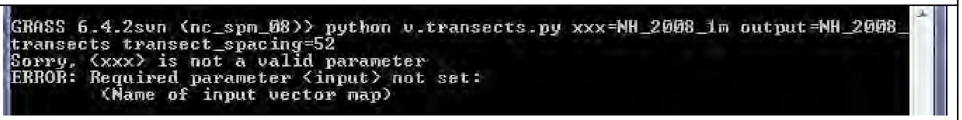
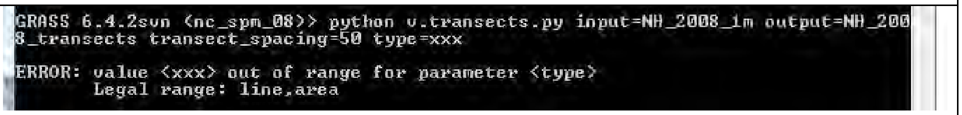
Script to Save WGS84 Datasets Created by v.transect to ASCII Files

```
v.out.ascii input=multLinesWGS_transects output=multLinesWGS_transects format=standard
v.out.ascii input=freeHandWGS_transects output=freeHandWGS_transects format=standard
v.out.ascii input=multLinesWGS01_transects output=multLinesWGS01_transects format=standard
```

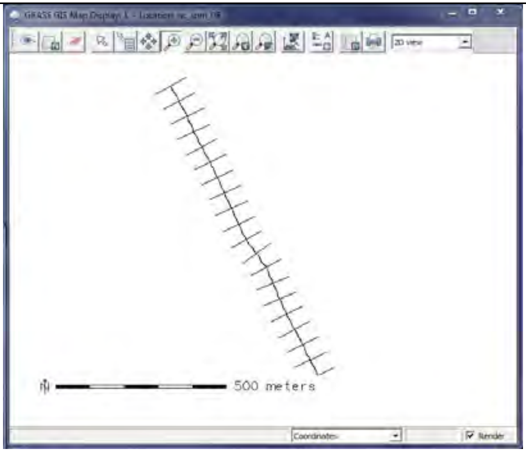
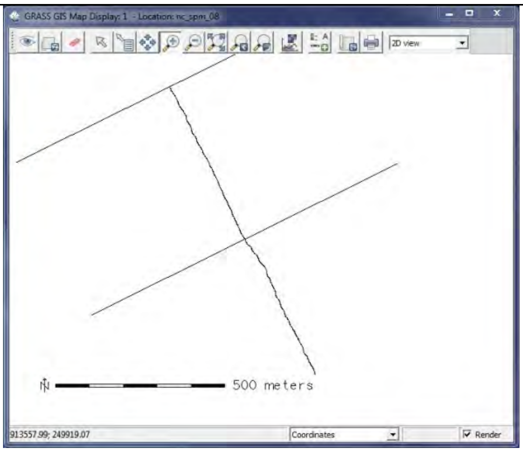
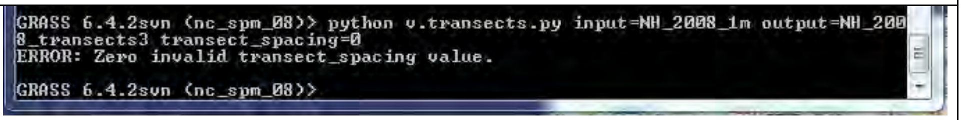
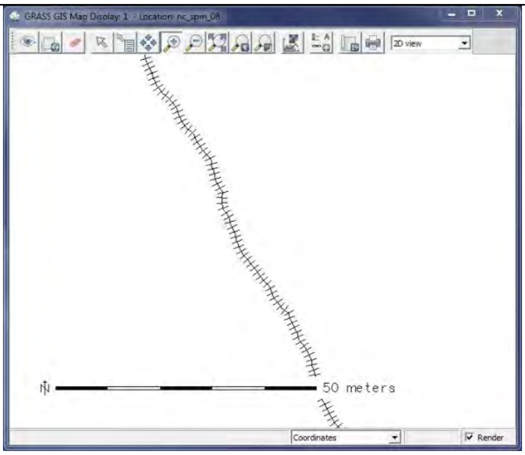
Test Case Results

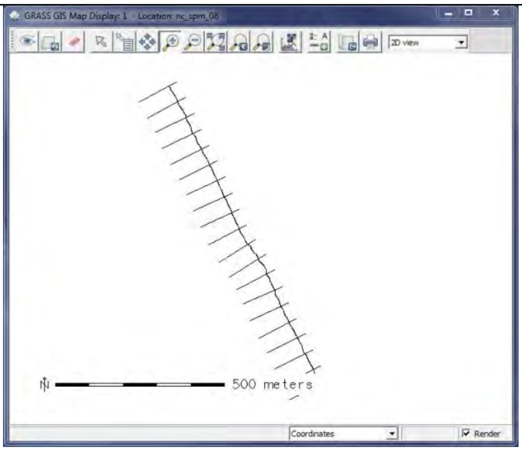
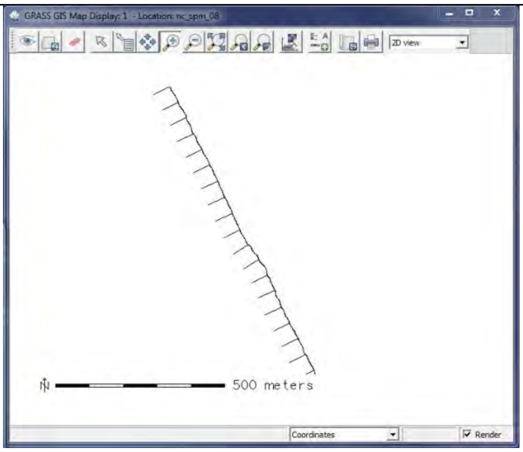
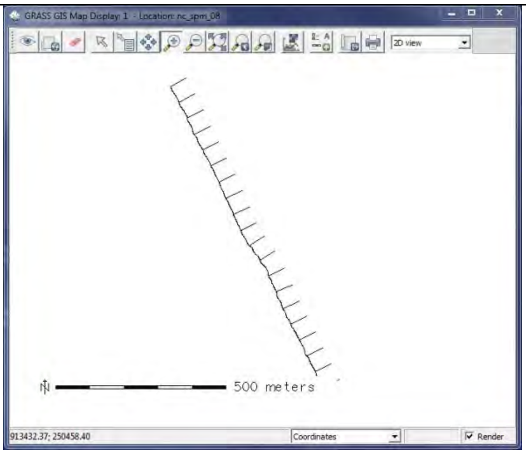
The results in this section are the outputs of the final v.transects.py python script in the Microsoft Windows7 environment.

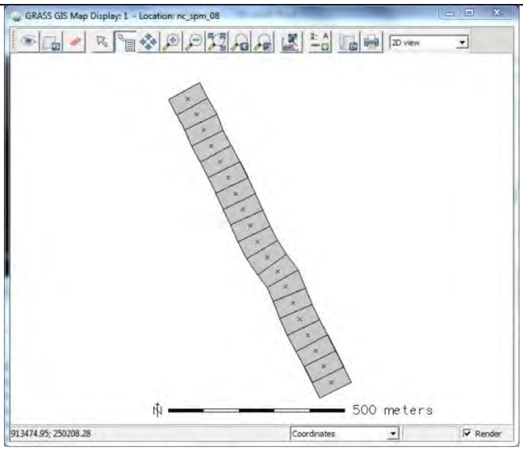
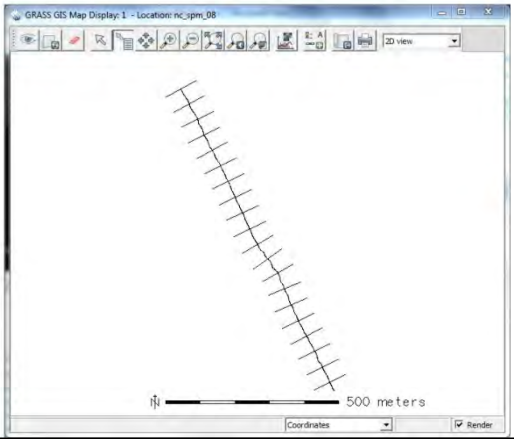
Parameter Verification Test Cases Results

#	Description	Output
1	Input parameter required	 GRASS 6.4.2svn (nc_spm_08)> python v.transects.py output=NH_2008_transects trans ect_spacing=50 ERROR: Required parameter <input> not set: (Name of input vector map)
2	Output parameter required	 GRASS 6.4.2svn (nc_spm_08)> python v.transects.py input=NH_2008_1m transect_spac ing=50 ERROR: Required parameter <output> not set: (Name of output vector map)
3	Transect_spacing parameter required	 GRASS 6.4.2svn (nc_spm_08)> python v.transects.py input=NH_2008_1m output=NH_200 8_transects ERROR: Required parameter <transect_spacing> not set: (Transect spacing)
4	Transect parameter number	 GRASS 6.4.2svn (nc_spm_08)> python v.transects.py input=NH_2008_1m output=NH_200 8_transects transect_spacing=xxx ERROR: Invalid transect_spacing value. GRASS 6.4.2svn (nc_spm_08)>
5	Dleft parameter number	 GRASS 6.4.2svn (nc_spm_08)> python v.transects.py input=NH_2008_1m output=NH_200 8_transects transect_spacing=52 dleft=xxx xxx 52.0 ERROR: Invalid dleft value. GRASS 6.4.2svn (nc_spm_08)>
6	Dright parameter number	 GRASS 6.4.2svn (nc_spm_08)> python v.transects.py input=NH_2008_1m output=NH_200 8_transects transect_spacing=52 dright=xxx 52.0 ERROR: Invalid dright value. GRASS 6.4.2svn (nc_spm_08)>
7	Input does not exist	 GRASS 6.4.2svn (nc_spm_08)> python v.transects.py input=xxx output=NH_2008_trans ects transect_spacing=52 52.0 ERROR: <xxx> does not exist. GRASS 6.4.2svn (nc_spm_08)>
8	Output exists	 GRASS 6.4.2svn (nc_spm_08)> python v.transects.py input=NH_2008_1m output=NH_200 8_transects transect_spacing=52 52.0 ERROR: output map <NH_2008_transects> exists GRASS 6.4.2svn (nc_spm_08)>
9	Invalid Option -	 GRASS 6.4.2svn (nc_spm_08)> python v.transects.py xxx=NH_2008_1m output=NH_2008_ transects transect_spacing=52 Sorry, <xxx> is not a valid parameter ERROR: Required parameter <input> not set: (Name of input vector map)
10	Invalid Type Option	 GRASS 6.4.2svn (nc_spm_08)> python v.transects.py input=NH_2008_1m output=NH_200 8_transects transect_spacing=50 type=xxx ERROR: value <xxx> out of range for parameter <type> Legal range: line,area

Shoreline Dataset Test Cases Results

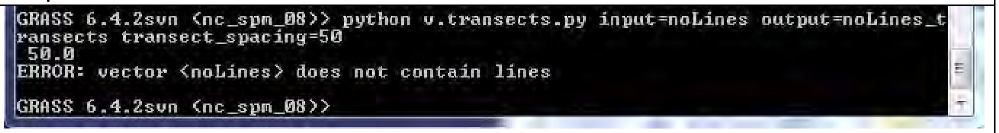
	Description	Output
1	Parameters used in paper	 GRASS GIS Map Display: 1 - Location: nc_spm_08 2D view 500 meters Coordinates Render
2	Large transect spacing	 GRASS GIS Map Display: 1 - Location: nc_spm_08 2D view 500 meters 913557.99; 249919.07 Coordinates Render
3	Zero transect spacing	 GRASS 6.4.2svn (nc_spm_08)> python v.transects.py input=NH_2008_1m output=NH_2008_transects3 transect_spacing=0 ERROR: Zero invalid transect_spacing value. GRASS 6.4.2svn (nc_spm_08)>
4	One meter transect spacing	 GRASS GIS Map Display: 1 - Location: nc_spm_08 2D view 50 meters Coordinates Render

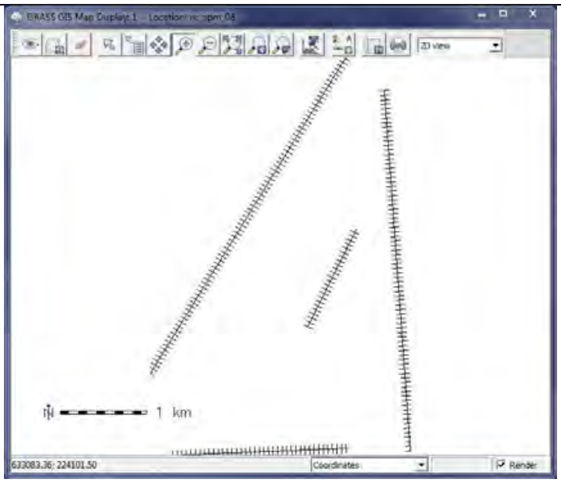
5	Left/right transect spacing different	 GRASS GIS Map Display: 1 - Location: nc_spm_08 2D view 500 meters Coordinates ☑ Render
6	Zero transect spacing on left	 GRASS GIS Map Display: 1 - Location: nc_spm_08 2D view 500 meters Coordinates ☑ Render
7	Zero transect spacing on right	 GRASS GIS Map Display: 1 - Location: nc_spm_08 2D view 500 meters 913432.37; 250458.40 Coordinates ☑ Render

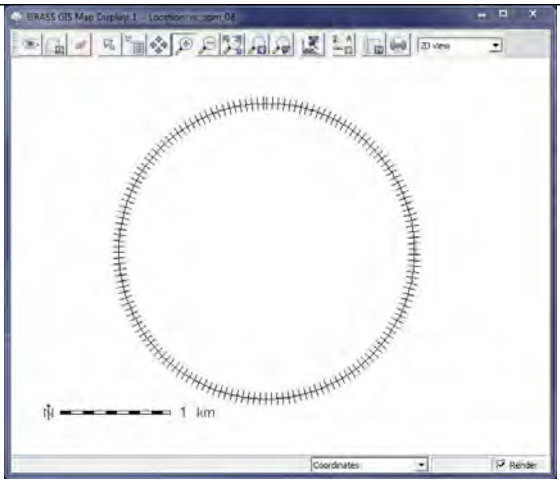
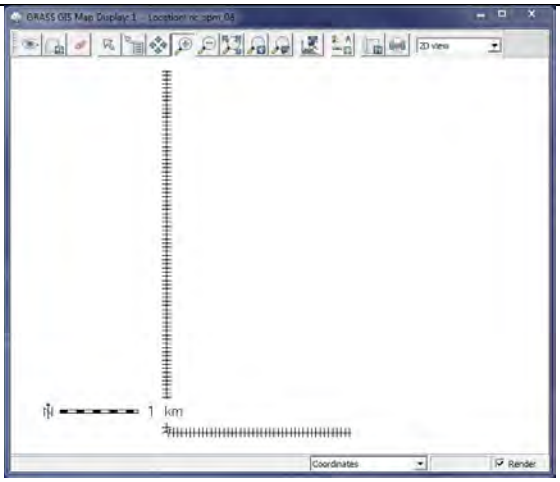
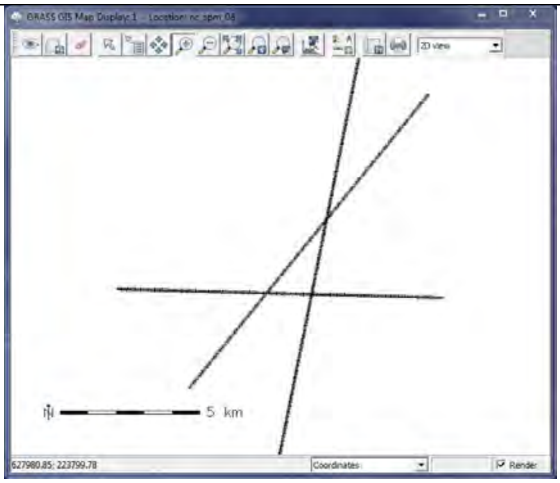
8	Parameters used in paper with area	 GRASS GIS Map Display: 1 - Location: nc_spm_08 2D view 500 meters 913474.95; 250208.28 Coordinates Render
9	Parameters used in paper with line	 GRASS GIS Map Display: 1 - Location: nc_spm_08 2D view 500 meters Coordinates Render

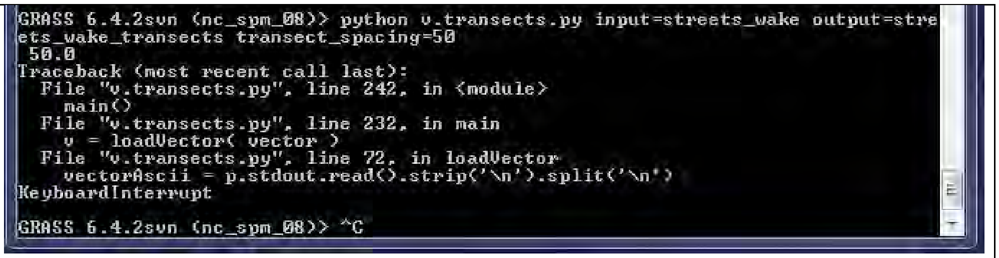
Verify Polylines of Various Shapes Test Cases Results

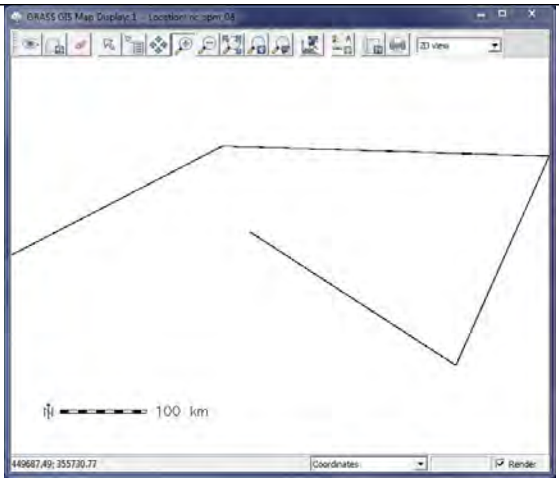
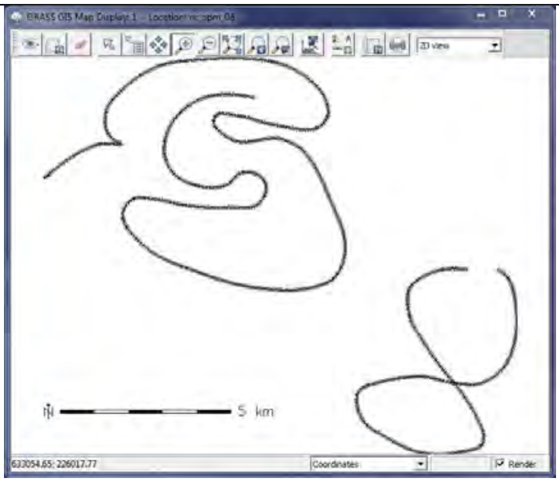
	Description	Output

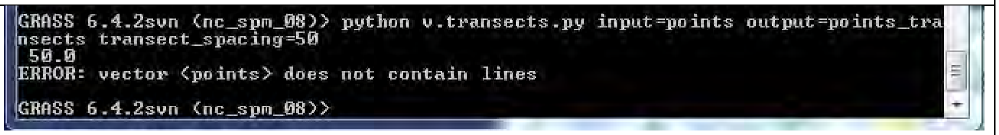
1	No lines in input	`GRASS 6.4.2svn (nc_spm_08)> python v.transects.py input=noLines output=noLines_t`
		`ransects transect_spacing=50`
		` 50.0`
		`ERROR: vector <noLines> does not contain lines`
		`GRASS 6.4.2svn (nc_spm_08)>`

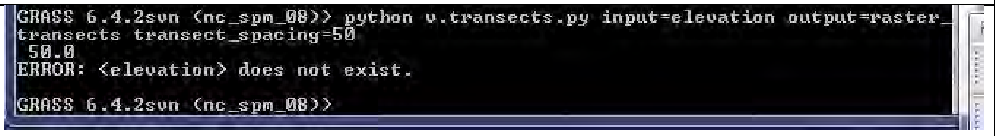
2	Multiple lines in input	 GRASS GIS Map Display: 1 - Location: nc_spm_08 2D view 1 km 633083.36; 224101.50 Coordinates Render

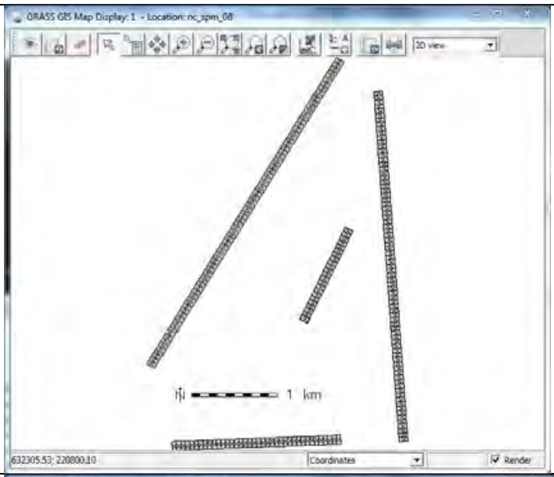
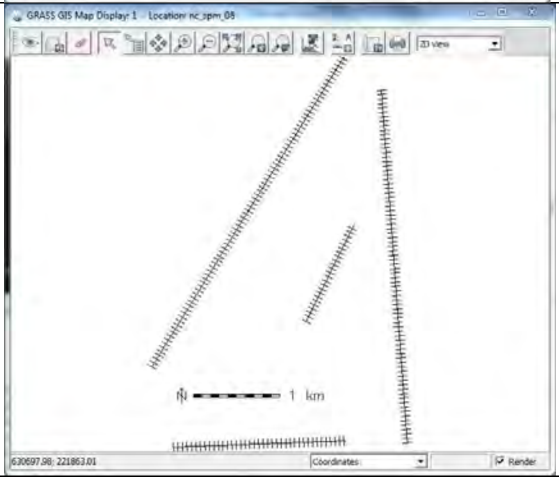
3	Circle in input	 GRASS GIS Map Display: 1 - Location: nc_spm_08 2D view 1 km Coordinates Render
4	Right angle in input	 GRASS GIS Map Display: 1 - Location: nc_spm_08 2D view 1 km Coordinates Render
5	Intersecting lines in input	 GRASS GIS Map Display: 1 - Location: nc_spm_08 2D view 5 km 627980.85; 223799.78 Coordinates Render

6	Network of Lines (streets)	
		`GRASS 6.4.2svn (nc_spm_08)> python v.transects.py input=streets_wake output=streets_wake_transects transect_spacing=50`
		` 50.0`
		`Traceback (most recent call last):`
		`  File "v.transects.py", line 242, in <module>`
		`    main()`
		`  File "v.transects.py", line 232, in main`
		`    v = loadVector( vector )`
		`  File "v.transects.py", line 72, in loadVector`
		`    vectorAscii = p.stdout.read().strip('\n').split('\n')`
		`KeyboardInterrupt`
		
		`GRASS 6.4.2svn (nc_spm_08)> ^C`

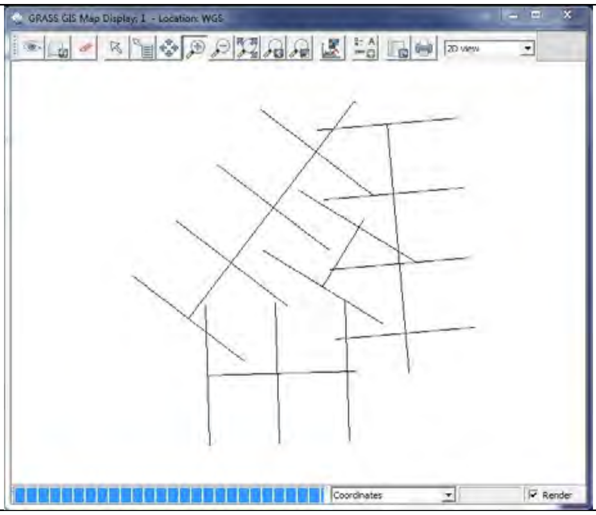
7	Super long line	 GRASS GIS Map Display: 1 - Location: nc_spm_08 2D view 100 km 449687.49; 355730.77 Coordinates Render
8	Freehand	 GRASS GIS Map Display: 1 - Location: nc_spm_08 2D view 5 km 633054.65; 226017.77 Coordinates Render

9	Points	
		`GRASS 6.4.2svn (nc_spm_08)> python v.transects.py input=points output=points_transects transect_spacing=50`
		` 50.0`
		`ERROR: vector <points> does not contain lines`
		
		`GRASS 6.4.2svn (nc_spm_08)>`

10	Raster	
		`GRASS 6.4.2svn (nc_spm_08)> python v.transects.py input=elevation output=raster_transects transect_spacing=50`
		` 50.0`
		`ERROR: <elevation> does not exist.`
		
		`GRASS 6.4.2svn (nc_spm_08)>`

11	Multiple lines in input with area	 GRASS GIS Map Display: 1 - Location: nc_spm_08 2D View 1 km 632305.53; 220800.10 Coordinates ☑ Render
12	Multiple lines in input with line	 GRASS GIS Map Display: 1 - Location: nc_spm_08 2D View 1 km 630697.98; 221863.01 Coordinates ☑ Render

WGS Dataset Test Cases Results

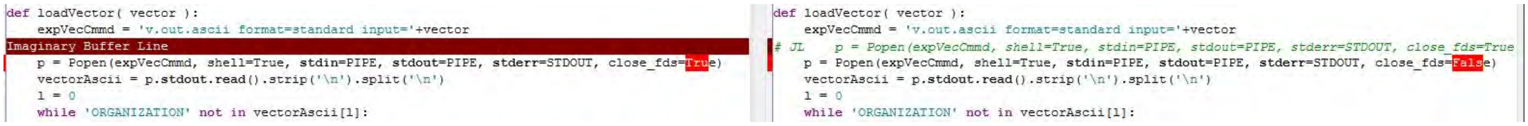
	Description	Output
1	Multiple lines in input	No features created
2	Freehand	No features created
3	Multiple lines in input	 GRASS GIS Map Display: 1 - Location: WGS 2D view Coordinates ☑ Render

v.transects.py Script

Updates to Script

The original v.transects.py script is in the left window and the updated final script in the right window.

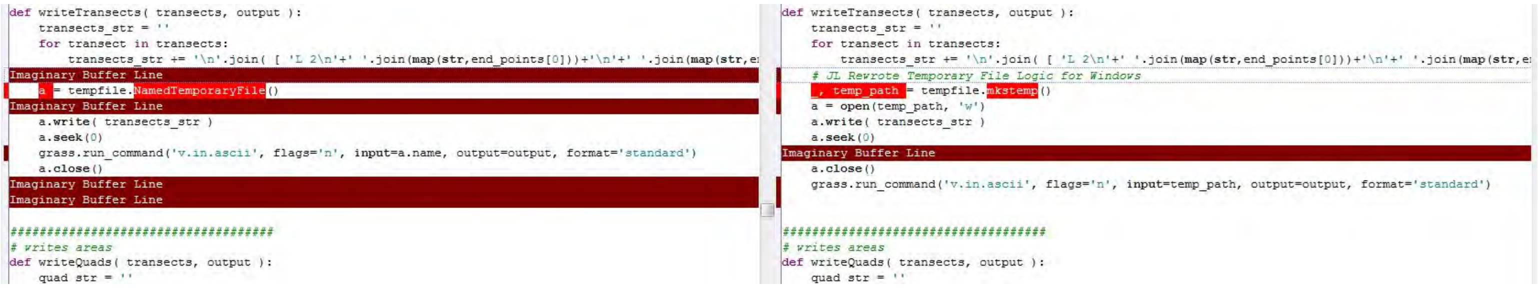
Update for Windows Implementation of Popen



```
def loadVector( vector ):
    expVecCmmd = 'v.out.ascii format=standard input='+vector
Imaginary Buffer Line
    p = Popen(expVecCmmd, shell=True, stdin=PIPE, stdout=PIPE, stderr=STDOUT, close_fds=True)
    vectorAscii = p.stdout.read().strip('\n').split('\n')
    l = 0
    while 'ORGANIZATION' not in vectorAscii[l]:
```

```
def loadVector( vector ):
    expVecCmmd = 'v.out.ascii format=standard input='+vector
# JL    p = Popen(expVecCmmd, shell=True, stdin=PIPE, stdout=PIPE, stderr=STDOUT, close_fds=True
    p = Popen(expVecCmmd, shell=True, stdin=PIPE, stdout=PIPE, stderr=STDOUT, close_fds=False)
    vectorAscii = p.stdout.read().strip('\n').split('\n')
    l = 0
    while 'ORGANIZATION' not in vectorAscii[l]:
```

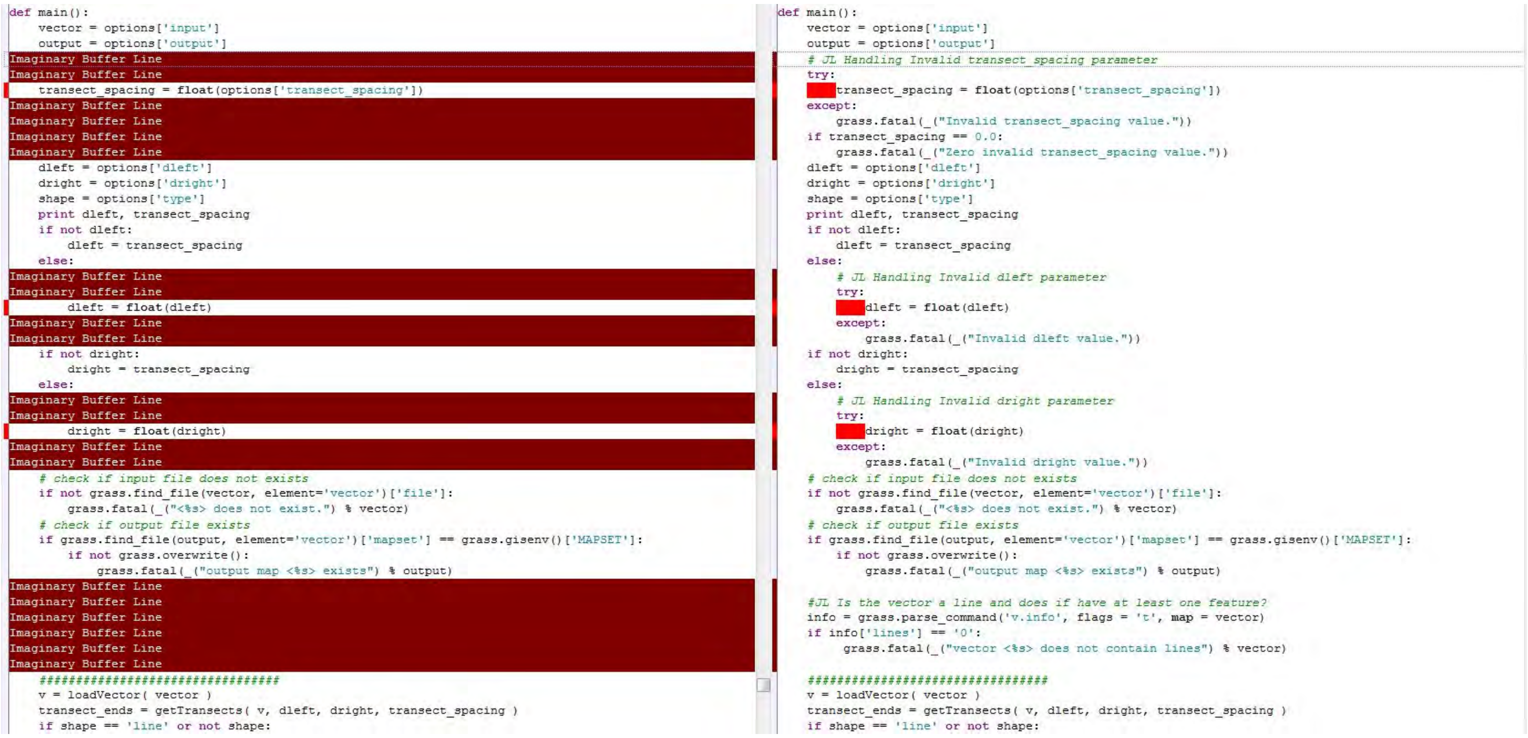
Update for Windows Implementation of NamedTemporaryFile



```
def writeTransects( transects, output ):
    transects_str = ''
    for transect in transects:
        transects_str += '\n'.join( [ 'L 2\n'+' '.join(map(str,end_points[0]))+'\n'+' '.join(map(str,e
Imaginary Buffer Line
    a = tempfile.NamedTemporaryFile()
Imaginary Buffer Line
    a.write( transects_str )
    a.seek(0)
    grass.run_command('v.in.ascii', flags='n', input=a.name, output=output, format='standard')
    a.close()
Imaginary Buffer Line
Imaginary Buffer Line

####################################
# writes areas
def writeQuads( transects, output ):
    quad str = ''
```

```
def writeTransects( transects, output ):
    transects_str = ''
    for transect in transects:
        transects_str += '\n'.join( [ 'L 2\n'+' '.join(map(str,end_points[0]))+'\n'+' '.join(map(str,e
    # JL Rewrote Temporary File Logic for Windows
    _, temp_path = tempfile.mkstemp()
    a = open(temp_path, 'w')
    a.write( transects_str )
    a.seek(0)
Imaginary Buffer Line
    a.close()
    grass.run_command('v.in.ascii', flags='n', input=temp_path, output=output, format='standard')


####################################
# writes areas
def writeQuads( transects, output ):
    quad str = ''
```

Updates for Parameter Checking



```
def main():
    vector = options['input']
    output = options['output']
Imaginary Buffer Line
Imaginary Buffer Line
    transect_spacing = float(options['transect_spacing'])
Imaginary Buffer Line
Imaginary Buffer Line
Imaginary Buffer Line
Imaginary Buffer Line
    dleft = options['dleft']
    dright = options['dright']
    shape = options['type']
    print dleft, transect_spacing
    if not dleft:
        dleft = transect_spacing
    else:
Imaginary Buffer Line
Imaginary Buffer Line
        dleft = float(dleft)
Imaginary Buffer Line
Imaginary Buffer Line
    if not dright:
        dright = transect_spacing
    else:
Imaginary Buffer Line
Imaginary Buffer Line
        dright = float(dright)
Imaginary Buffer Line
Imaginary Buffer Line
    # check if input file does not exists
    if not grass.find_file(vector, element='vector')['file']:
        grass.fatal(_("<%s> does not exist.") % vector)
    # check if output file exists
    if grass.find_file(output, element='vector')['mapset'] == grass.gisenv()['MAPSET']:
        if not grass.overwrite():
            grass.fatal(_("output map <%s> exists") % output)
Imaginary Buffer Line
Imaginary Buffer Line
Imaginary Buffer Line
Imaginary Buffer Line
Imaginary Buffer Line
Imaginary Buffer Line
    ####################################
    v = loadVector( vector )
    transect_ends = getTransects( v, dleft, dright, transect_spacing )
    if shape == 'line' or not shape:
```

```
def main():
    vector = options['input']
    output = options['output']
    # JL Handling Invalid transect_spacing parameter
    try:
        transect_spacing = float(options['transect_spacing'])
    except:
        grass.fatal(_("Invalid transect_spacing value."))
    if transect_spacing == 0.0:
        grass.fatal(_("Zero invalid transect_spacing value."))
    dleft = options['dleft']
    dright = options['dright']
    shape = options['type']
    print dleft, transect_spacing
    if not dleft:
        dleft = transect_spacing
    else:
        # JL Handling Invalid dleft parameter
        try:
            dleft = float(dleft)
        except:
            grass.fatal(_("Invalid dleft value."))
    if not dright:
        dright = transect_spacing
    else:
        # JL Handling Invalid dright parameter
        try:
            dright = float(dright)
        except:
            grass.fatal(_("Invalid dright value."))
    # check if input file does not exists
    if not grass.find_file(vector, element='vector')['file']:
        grass.fatal(_("<%s> does not exist.") % vector)
    # check if output file exists
    if grass.find_file(output, element='vector')['mapset'] == grass.gisenv()['MAPSET']:
        if not grass.overwrite():
            grass.fatal(_("output map <%s> exists") % output)

    #JL Is the vector a line and does if have at least one feature?
    info = grass.parse_command('v.info', flags = 't', map = vector)
    if info['lines'] == '0':
         grass.fatal(_("vector <%s> does not contain lines") % vector)

    ####################################
    v = loadVector( vector )
    transect_ends = getTransects( v, dleft, dright, transect_spacing )
    if shape == 'line' or not shape:
```

Man page

The HTML for the man page associated with the v.transects.py script is below. The man page can be viewed here http://www4.ncsu.edu/~jlloyd/Lloydj_MEA582f11/v_transects.html.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
```

```
<head>
<title>GRASS GIS manual: v.transects.py</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<link rel="stylesheet" href="grassdocs.css" type="text/css">
</head>
<body bgcolor="white">

<img src="grass_logo.png" alt="GRASS logo"><hr align=center size=6 noshade>

<h2>NAME</h2>
<em><b>v.transects.py</b></em>  - Python script to create vector line or area transects along a
line
<h2>KEYWORDS</h2>
vector, transects, shoreline

<h2>SYNOPSIS</h2>
<b>v.transects.py</b> <b>input</b>=<em>name</em>  <b>output</b>=<em>name</em>
<b>transect_spacing</b>=<em>float</em>   [<b>dleft</b>=<em>float</em>]
[<b>dright</b>=<em>float</em>]   [<b>type</b>="line" | "area"]



<h3>Parameters:</h3>
<DL>
<DT><b>input</b>=<em>name</em></DT>
<DD>Name for input line vector map</DD>

<DT><b>output</b>=<em>name</em></DT>
<DD>Name for output transects vector map</DD>

<DT><b>transect_spacing</b>=<em>float</em></DT>
<DD>Distance between transects</DD>

<DT><b>dleft</b>=<em>float</em></DT>
<DD>Distance transects extend on left side of line</DD>
<DD>Default: <em>transect_spacing</em></DD>

<DT><b>dright</b>=<em>float</em></DT>
<DD>Distance transects extend on right side of line</DD>
<DD>Default: <em>transect_spacing</em></DD>

<DT><b>type</b>="line" | "area"</DT>
<DD>Specifies the geometry of the transect</DD>
<DD>Default: "line"</DD>

</DL>


<H2>DESCRIPTION</H2><EM>v.transects</EM> creates equally spaced geometries along
input lines. The geometries can be lines or quadrilateral areas. Lines
and areas are generated to be perpendicular to the input line.


<H2>NOTES</H2>Input vector lines that are shorter than <B>transect_spacing</B>
are ignored.


<H2>EXAMPLES</H2>In these examples, the Nags Head location is used to generate a
shoreline and shore-perpendicular geometries:

<h3>Example 1) - Generate line transects along shoreline</h3>
```

```
<P>Generate 20 cross-shore transects along 2008 shoreline (1m contour)<BR>
<DIV class=code><PRE>r.contour input=NH_2008_1m output=NH_2008_1m level=1 cut=100
v.report map=NH_2008_1m option=length
# cat|level|length
# 1|1|1037.86684790028
# 1038m / 20transects = 52m per transect (value for transect_spacing)
v.transects input=NH_2008_1m output=NH_2008_transects transect_spacing=52
v.info NH_2008_transects
</PRE></DIV>

<h3>Example 2) - Generate line transects specifying the left and right length</h3>

<P>Generate longer, more parallel transects by specifying dleft and dright and
smoothing the input line<BR>
<DIV class=code><PRE>r.contour input=NH_2008_1m output=NH_2008_1m level=1 cut=100
v.generalize input=NH_2008_1m output=NH_2008_1m_smoothed \
  method=sliding_averaging look_ahead=201
v.transects input=NH_2008_1m_smoothed \
  output=NH_2008_transects_long_smoothed transect_spacing=52 \
  dleft=20 dright=300
</PRE></DIV>

<h3>Example 3) - Generate area transects along shoreline</h3>

<P>Generate longer, more parallel transects by specifying dleft and dright and
smoothing the input line<BR>
<DIV class=code><PRE>r.contour input=NH_2008_1m output=NH_2008_1m level=1 cut=100
v.transects input=NH_2008_1m output=NH_2008_areas \
  transect_spacing=52 dleft=20 dright=300 type=area
v.db.addtable NH_2008_areas
v.db.addcolumn map=NH_2008_areas columns='demStats DOUBLE PRECISION'
v.rast.stats vector=NH_2008_areas raster=NH_2008_1m column_prefix=NH2008
v.db.select NH_2008_areas
</PRE></DIV>


<H2>SEE ALSO</H2><EM><A
href="http://www4.ncsu.edu/~ejhardi2/v.generalize.html">v.generalize</A>, <A
href="http://www4.ncsu.edu/~ejhardi2/r.transect.html">r.transect</A> </EM>


<H2>AUTHOR</H2>Eric Hardin, Helena Mitasova, Updates by John Lloyd
<P><I>Last changed: $Date$</I> </P></BODY></HTML>
```

Final Script

My changes are marked with the initials "JL". The script is also available here http://www4.ncsu.edu/~jlloyd/Lloydj_MEA582f11/v.transects.py.

```
#!/usr/bin/env python
#
##############################################################################
#
# MODULE:       v.transects.py
# AUTHOR(S):    Eric Hardin
# PURPOSE:      Creates lines or quadrilateral areas perpendicular to a polyline
# COPYRIGHT:    (C) 2011
#
#               This program is free software under the GNU General Public
#               License (>=v2). Read the file COPYING that comes with GRASS
#               for details.
#
# UPDATES:     John Lloyd November 2011
#
#############################################################################
#%Module
#% description: Transect Generator
#%End
#%option
#% key: input
#% type: string
#% description: Name of input vector map
#% required : yes
#%end
#%option
#% key: output
#% type: string
#% description: Name of output vector map
#% required : yes
#%end
#%option
#% key: transect_spacing
#% type: double
#% description: Transect spacing
#% required : yes
#%end
#%option
#% key: dleft
#% type: double
#% description: distance transect extends to the left of input line.
#% required : no
#%end
#%option
#% key: dright
#% type: double
#% description: distance transect extends to the right of input line.
#% required : no
#%end
#%option
#% key: type
#% type: string
#% description: Feature type
#% required : no
#% options: line,area
#% answer : line
#%end

from subprocess import Popen, PIPE, STDOUT
from numpy import array
from math import sqrt
import grass.script as grass
import tempfile
####################################
# load vector lines into python list
# returns v
# len(v) = number of lines in vector map
# len(v[i]) = number of vertices in ith line
# v[i][j] = [ xij, yij ] ,i.e., jth vertex in ith line
def loadVector( vector ):
    expVecCmmd = 'v.out.ascii format=standard input='+vector
# JL    p = Popen(expVecCmmd, shell=True, stdin=PIPE, stdout=PIPE, stderr=STDOUT, close_fds=True)
```

```
    p = Popen(expVecCmmd, shell=True, stdin=PIPE, stdout=PIPE, stderr=STDOUT, close_fds=False)
    vectorAscii = p.stdout.read().strip('\n').split('\n')
    l = 0
    while 'ORGANIZATION' not in vectorAscii[l]:
        l += 1
    while ':' in vectorAscii[l]:
        l += 1
    v = []
    while l < len(vectorAscii):
        line = vectorAscii[l].split()
        if line[0] in ['L','B','A']:
            skip = len(line)-2
            vertices = int(line[1])
            l += 1
            v.append([])
            for i in range(vertices):
                v[-1].append( map(float,vectorAscii[l].split()[:2]) )
                l += 1
            l += skip
        elif line[0] in ['P','C','F','K']:
            skip = len(line)-2
            vertices = int(line[1])
            l += 1
            for i in range(vertices):
                l += 1
            l += skip
        else:
            grass.fatal(_("Problem with line: <%s>") % vectorAscii[l])
    if len(v) < 1:
        grass.fatal(_("Zero lines found in vector map <%s>") % vector)
    return v

####################################
# get transects from input vector
def getTransects( vector, dleft, dright, transect_spacing ):
    transect_locs = [] # holds locations where transects should intersect input vector lines
    for line in vector:
        transect_locs.append([line[0]])
        # i starts at the beginning of the line.
        # j walks along the line until j and i are separated by a distance of transect_spacing.
        # then, a transect is placed at j, i is moved to j, and this is iterated until the end of
the line is reached
        i = 0
        j = i+1
        while j < len(line):
            d = dist( line[i],line[j] )
            if d > transect_spacing:
                r = (transect_spacing - dist(line[i],line[j-1]) )/( dist(line[i],line[j]) -
dist(line[i],line[j-1]) )
                transect_locs[-1].append(  (r*array( line[j] ) + (1-r)*array( line[j-1]
)).tolist()  )
                i = j-1
                line[i] = transect_locs[-1][-1]
            else:
                j += 1
    transect_ends = []
    for transect in transect_locs:
        if len(transect) < 2: # if a line in input vec was shorter than transect_spacing
            continue # then don't put a transect on it
        transect_ends.append([])
        transect = array( transect )
        v = NR( transect[0], transect[1] ) # vector pointing parallel to transect
        transect_ends[-1].append( [ transect[0]+dleft*v, transect[0]-dright*v ] )
        for i in range(1,len( transect )-1,1):
            v = NR( transect[i-1], transect[i+1] )
            transect_ends[-1].append( [ transect[i]+dleft*v, transect[i]-dright*v ] )
        v = NR( transect[-2], transect[-1] )
        transect_ends[-1].append( [ transect[-1]+dleft*v, transect[-1]-dright*v ] )
    return transect_ends

####################################
# calculate scalar distance between two points
def dist( ip, fp ):
    return sqrt( (ip[0]-fp[0])**2 + (ip[1]-fp[1])**2 )

####################################
# take a vector, normalize and rotate it 90 degrees
def NR( ip, fp ):
    x = fp[0] - ip[0]
    y = fp[1] - ip[1]
    r = sqrt( x**2 + y**2 )
    return array([ -y/r, x/r ])

####################################
# write transects
def writeTransects( transects, output ):
    transects_str = ''
    for transect in transects:
        transects_str += '\n'.join( [ 'L 2\n'+' '.join(map(str,end_points[0]))+'\n'+'
'.join(map(str,end_points[1]))+'\n' for end_points in transect ] )
    # JL Rewrote Temporary File Logic for Windows
    _, temp_path = tempfile.mkstemp()
    a = open(temp_path, 'w')
    a.write( transects_str )
    a.seek(0)
    a.close()
    grass.run_command('v.in.ascii', flags='n', input=temp_path, output=output, format='standard')


####################################
# writes areas
def writeQuads( transects, output ):
    quad_str = ''
    cnt = 1
    for line in transects:
        for tran in range( len(line)-1 ):
            pt1 = ' '.join( map(str,line[tran][0]) )
            pt2 = ' '.join( map(str,line[tran][1]) )
            pt3 = ' '.join( map(str,line[tran+1][1]) )
            pt4 = ' '.join( map(str,line[tran+1][0]) )
            pt5 = pt1
            # centroid is the average of the four corners
            c = ' '.join( map(str,[
0.25*(line[tran][0][0]+line[tran][1][0]+line[tran+1][0][0]+line[tran+1][1][0]),
0.25*(line[tran][0][1]+line[tran][1][1]+line[tran+1][0][1]+line[tran+1][1][1]) ]) )
            quad_str += 'B 5\n' + '\n'.join([pt1,pt2,pt3,pt4,pt5]) + '\n'
            quad_str += 'C 1 1\n' + c + '\n1 ' + str(cnt) + '\n'
            cnt += 1
    # JL Rewrote Temporary File Logic for Windows
    _, temp_path = tempfile.mkstemp()
    a = open(temp_path, 'w')
    a.write( quad_str )
    a.seek(0)
    a.close()
    grass.run_command('v.in.ascii', flags='n', input=a.name, output=output, format='standard')
    a.close()

####################################
# Main method
def main():
    vector = options['input']
    output = options['output']
    # JL Handling Invalid transect_spacing parameter
    try:
        transect_spacing = float(options['transect_spacing'])
    except:
        grass.fatal(_("Invalid transect_spacing value."))
    if transect_spacing == 0.0:
        grass.fatal(_("Zero invalid transect_spacing value."))
    dleft = options['dleft']
    dright = options['dright']
    shape = options['type']
    print dleft, transect_spacing
    if not dleft:
        dleft = transect_spacing
    else:
        # JL Handling Invalid dleft parameter
        try:
            dleft = float(dleft)
        except:
            grass.fatal(_("Invalid dleft value."))
    if not dright:
        dright = transect_spacing
    else:
        # JL Handling Invalid dright parameter
        try:
            dright = float(dright)
        except:
            grass.fatal(_("Invalid dright value."))
    # check if input file does not exists
    if not grass.find_file(vector, element='vector')['file']:
        grass.fatal(_("<%s> does not exist.") % vector)
    # check if output file exists
    if grass.find_file(output, element='vector')['mapset'] == grass.gisenv()['MAPSET']:
        if not grass.overwrite():
            grass.fatal(_("output map <%s> exists") % output)

    #JL Is the vector a line and does if have at least one feature?
    info = grass.parse_command('v.info', flags = 't', map = vector)
    if info['lines'] == '0':
         grass.fatal(_("vector <%s> does not contain lines") % vector)

    #################################
    v = loadVector( vector )
    transect_ends = getTransects( v, dleft, dright, transect_spacing )
    if shape == 'line' or not shape:
        writeTransects( transect_ends, output )
    else:
        writeQuads( transect_ends, output )

if __name__ == "__main__":
   options, flags = grass.parser()
   main()
```