gpsmanshp
A Tcl Package to Read and Write Shapefiles

Miguel Filgueiras

DCC-FC & LIACC, Universidade do Porto

March 2003

This document describes the Tcl commands defined in the `gpsmanshp` package that provides the means for creating and reading files in the ESRI Shapefile format for keeping 2 or 3 dimensional points and polylines.

`gpsmanshp` was developed for use in GPSMan, a manager of GPS receiver data, available from: `http://www.ncc.up.pt/~mig/gpsman.html`. It is written in C and is based on `shapelib` made available by Frank Warmerdam (`warmerda@home.com`).

Along with this package, `gpstr2shp.c` was also developed. It translates GPStrans data files into Shapefile ones.

1 Tcl Commands Implemented

The general philosophy is that of dealing with the Shapefile files (the data file .shp, the index file .shx, and the attribute file .dbf) as a single entity that is called below a set of files. There are Tcl commands for creating a new set and for opening (for reading) an existing one, as well for closing a set that was created or opened.

Each set has a type and dimension associated to it. The basic types are waypoint, route and track, that correspond to the Shapefile point and polylines shapes. When reading from a Shapefile polyline the distinction between routes and tracks can only be made if its attribute field names are the same as those written by `gpsmanshp`. The type `UNKNOWN` is used if this distinction cannot be made.

Input/output of polylines is made by first getting/creating an internal description and then reading/writing each point at a time. At the same time there can be no more than one route and one track being created, and no more than one polyline being read from each set.

1.1 General Command

`GSHPCloseFiles ID`

- closes a set of files identified by the integer `ID` returned by the procedures that create new files (`GSHPCreateFiles`) or open existing files for reading (`GSHPOpenInputFiles`)
- returns 0 if `ID` is not a valid identifier; otherwise 1

1.2 Output Commands

`GSHPCreateFiles BASEPATH TYPE DIM`

- creates a new set of SHP files (.shp, .shx, .dbf) for writing
- `TYPE` is one of `WP`, `RT` or `TR` that are mapped into Shape types with the measure field set to 0 and fields in .dbf file as follows
  - `WP` (waypoint)
    - `name`, string up to 50 characters
    - `commt`, comment, string up to 128 characters
    - `date`, string up to 25 characters
  - `RT` (route, a sequence of waypoints)
    - `id`, identifier, string up to 50 characters
    - `commt`, comment, string up to 128 characters
  - `TR` (track, a sequence of track points)
    - `name`, string up to 50 characters
    - `commt`, comment, string up to 128 characters
- `DIM` is either 2 or 3 for plane or spatial coordinates
- returns a positive integer that identifies uniquely this set (until the files are closed), or 0 on error opening the files, -1 if `TYPE` is invalid, -2 if `DIM` is invalid, -3 if could not create .dbf fields, -4 if out of memory

`GSHPWriteWP ID X Y ?Z? NAME COMMENT DATE`

- writes waypoint to `ID` file set; the `Z` coordinate should be given only if the dimension for the file set was declared to be 3
- returns 1 on sucess, -1 if `ID` is invalid, -2 if type of `ID` set is not waypoint or the dimension is wrong, -3 if out of memory, -4 if failed to write .dbf field

`GSHPCreateRT DIM RTID COMMENT`

- starts creating a representation of a route; only one such representation can be handled at a time
- `DIM` is either 2 or 3 for plane or spatial coordinates
- returns 1 on success, 0 if there is already a route representation, -1 if `DIM` is invalid

`GSHPForgetRT`

- destroys current route representation
- returns 1 on success, 0 if there is no route representation

`GSHPAddWPToRT X Y ?Z?`

- adds waypoint to current route representation
- the `Z` coordinate can only be given if the route dimension is 3
- waypoints can be added after the current route has been written by `GSHPWriteRT`, but no files will be changed by this
- returns 1 on sucess, -1 if there is no route representation or the route dimension is not compatible with number of arguments, -2 if out of memory

`GSHPWriteRT ID FORGET`

- writes current route representation to `ID` file set; if the integer `FORGET` is different from 0 forget the route representation after writing it
- returns 1 on sucess, -1 if there is no route representation, -2 if there are no waypoints, -3 if `ID` is invalid, -4 if type of `ID` set is not route or the dimension is different, -5 if out of memory, -6 if failed to write .dbf field

`GSHPCreateTR DIM NAME COMMENT`

- starts creating a representation of a track; only one such representation can be handled at a time
- `DIM` is either 2 or 3 for plane or spatial coordinates
- returns 1 on success, 0 if there is already a track representation, -1 if `DIM` is invalid

`GSHPForgetTR`

- destroys current track representation
- returns 1 on success, -1 if there is no track representation

`GSHPAddTPToTR X Y ?Z?`

- adds track point to current track representation
- the `Z` coordinate can only be given if the track dimension is 3
- track points can added after the current track has been written by `GSHPWriteRT`, but no files will be changed by this
- returns 1 on sucess, -1 if there is no track representation or it has a dimension incompatible with the number of arguments, -2 if out of memory

`GSHPWriteTR ID FORGET`

- writes current track representation to `ID` file set; if the integer `FORGET` is different from 0 forgets the track representation after writing it
- returns 1 on sucess, -1 if there is no track representation, -2 if there are no waypoints, -3 if `ID` is invalid, -4 if type of `ID` set is not track or the track has a different dimension, -5 if out of memory, -6 if failed to write .dbf field

1.3 Input Commands

`GSHPOpenInputFiles BASEPATH`

- opens for input a set of SHP files (.shp, .shx, .dbf)
- .dbf file is only used if it has exactly the same fields as those written by any of the output commands above
- returns a positive integer that identifies uniquely this set (until the files are closed), or 0 on error opening the .shp or .shx files, -1 if files are empty, -2 if files type is invalid, -3 if out of memory

`GSHPInfoFrom ID`

- returns information on the contents of the `ID` input file set; the normal result is a list with
  - the type of objects in the file set: one of `WP`, `RT`, `TR`, or `UNKNOWN`; the latter is used for a polyline that can be taken either as a route or a track, as there is no available information from the .dbf file
  - the number of objects in the file (always positive); this is needed because objects will be indexed (by `GSHPGetObj`) from 0 to this number less 1
  - the dimension (2 or 3) of the objects
  - if the type is `RT`, `TR`, or `UNKNOWN`, the index of the next point to be read; the initial value is -1 when no object is being read or after all points of an object have already been read; the index is set to 0 by a call to `GSHPGetObj`, increased by `GSHPReadNextPoint` after a successful call, and set by this procedure to -1 when there are no more points
- possible error result: 0 no such input file set

`GSHPGetObj ID INDEX`

- reads an object from the `ID` input file set; `INDEX` is a number from 0 to the number of objects less one; the normal result is, depending on the type,
  - `WP`: a list with name, comment, date, x- and y-coordinates, and z-coordinate if dimension is 3; the first three can be empty strings if no information is available; numeric values of less than -10^{35} should be considered as undefined
  - `RT`: a list with identifier, comment, number of waypoints; the first two can be empty strings if no information is available; the waypoints should be read by `GSHPReadNextPoint`
  - `TR`: a list with name, comment, number of track points; the first two can be empty strings if no information is available; the track points should be read by `GSHPReadNextPoint`
  - `UNKNOWN`: the number of points; the points should be read by `GSHPReadNextPoint`

- the result is an empty list if the object is null (these objects may appear in Shapefile files and should be skipped)
- possible error results (note that the number of points returned when the type is `UNKNOWN` may be 0): -1 no such input file set, -2 if `INDEX` is invalid

`GSHPReadNextPoint ID`

- reads the next point from a route or track input file set with number `ID`; this can only be used after a call to `GSHPGetObj` returning a `RT`, `TR` or `UNKNOWN` type, and before a call of `GSHPReadNextPoint` for this set that gave a "no more points" result
- the normal result is a list with 2 or 3 coordinates depending on the dimension; numeric values of less than -10^{35} should be considered as undefined
- possible error results: 0 no such input file set, -1 if no route or track is currently being processed (no previous call to `GSHPReadNextPoint` with a suitable type result, or all points have already been read), -2 if there are no more points (after which a -1 result will be given)

2 Downloading and Installation

This package has been tested only under a Linux system. It is required that `shapelib` (version 1.2.9 was the one used) is installed, otherwise it cannot be compiled.

`gpsmanshp` and `gpstr2shp.c` are distributed under the GNU Public License with absolutely no warranties.

2.1 Downloading

`gpsmanshp` and `gpstr2shp.c` are available from `http://www.ncc.up.pt/~mig/gpsmanshp`

2.2 Installation

The enclosed `Makefile` defines `TCLVERSION` and assumes that

- `tclsh` is called as `tclsh$(TCLVERSION)`,
- the Tcl include directory is `/usr/include/tcl$(TCLVERSION)`,
- the Tcl library is `libtcl$(TCLVERSION)` and the `shapelib` library is `libshp`,
- `/usr/lib/tcl$(TCLVERSION)` is in the Tcl pre-defined list `$auto_path` (so that packages in this directory and its sub-directories are visible from Tcl) and that this package will be installed under its sub-directory `gpsmanshp`.

Using `make` will (in a Unix/Linux system) create the library `gpsmanshp.so` and the corresponding Tcl index `pkgIndex.tcl`, while `make install` will copy these files to the installation directory (but note that the index file must be copied to `/usr/lib/tcl$(TCLVERSION)` if this is not the installation directory) and `make clean` removes the binary files and the index file.

Acknowledgements

The work presented here has been partially supported by funds granted to LIACC through the Programa de Financiamento Plurianual, Fundação para a Ciência e Tecnologia and Programa POSI.