Mapfile structure

Comments

Any text after a # is a comment. Use comments a lot to document your mapfile.

Objects or Groups

A mapfile is composed of several objects. Each object start by the keyword object and finishes by END. Inside an objec
there can be several objects with the object starting by its keyword and finishing by END.
A mapfile starts by the keyword MAP and finishes by END

```
MAP
... some lines...
END # end of map
```

Inside the mapfile there are objects like layers, each layer start by the keyword LAYER and finishes by END

```
MAP
...
  LAYER
  ...
  END # end of layer

  LAYER
  ...
  END # end of layer
END # end of map
```

Do not try to figure how many END you need to put in the mapfile but use indentation to clearly identify each object
with its keyword, text and END keyword. All objects are nested and with proper indentation, they are clearly visible.
Indentation is not required but facilitates understanding a mapfile. Adequate comments help a lot too.

A word on colors

The keyword COLOR, OUTLINECOLOR,.. specify the color in the RGB color space. The first value is the red intensity from 0 (no red) to 255, the second value is the green intensity from 0 (no green) to 255 and the last value is the blue intensity from 0 (no blue) to 255. A *COLOR 0 0 0* indicates black while *COLOR 255 255 255* indicates white. *COLOR 0 255 0* would be pure green. The 3 values are compulsory and must be numbers. To select some correct values, use a color selector like the one inside MS-Paint in accessories. You select the color and the RGB values are indicated.

Mapfile header

he mapfile header is composed of several objects for the representation of the maps on the screen, you have the ma
tself, the web object to define how the web image is created, the reference or overview, the scalebar, the legend anc
if the map should be queryable. It is better to use pre-canned header and modify little parameters to conform to a
specific map.
This is the standard header for a map object

```
#
# Start of map file
#
MAP
NAME "Efate"
STATUS ON
SIZE 400 400
EXTENT 196900 8027100 245000 8073000
UNITS METERS
TRANSPARENT OFF
SHAPEPATH "/var/www/html/map/"
IMAGETYPE png24
FONTSET "fonts/fontset.txt"

OUTPUTFORMAT
  NAME png24
  DRIVER "GD/PNG"
  MIMETYPE "image/png"
  IMAGEMODE RGB
  EXTENSION "png"
END

#
# Start of web interface definition
#
WEB
  TEMPLATE /var/www/html/map/map.html
  IMAGEPATH /var/www/html/map/images/
  IMAGEURL /map/images/
  LOG /var/www/html/map/maplog
END

#
# Start of reference map
#
REFERENCE
  IMAGE /var/www/html/map/data/efate.png
  EXTENT 196900 8027100 245000 8073000
  STATUS ON
  COLOR -1 -1 -1
  OUTLINECOLOR 255 0 0
  SIZE 95 95
END

#
```

```
# Start of legend
#
LEGEND
  KEYSIZE 18 12
  LABEL
    TYPE BITMAP
    SIZE MEDIUM
    COLOR 0 0 89
  END
  STATUS ON
END

#
# Start of scalebar
#
SCALEBAR
  IMAGECOLOR 255 255 255
  LABEL
    COLOR 0 0 0
    SIZE SMALL
  END
  SIZE 350 5
  COLOR 255 255 255
  BACKGROUNDCOLOR 0 0 0
  OUTLINECOLOR 0 0 0
  UNITS kilometers
  INTERVALS 5
  STATUS ON
END

#
# Start of query definitions
#
QUERYMAP
  STATUS ON
  STYLE HILITE
END
```

Tip: A quick way to start a mapfile is to take the above and just add a END at the end of it. Point to a real image in the REFERENCE section and change all the paths to reflect your own installation. You will have then a working mapfile with no layers. You can then add a GRID to it and start to make modification to suit what you really want to achieve.

It is important here to have the following paths set correctly in regards to where you installed the mapfiles:

- SHAPEPATH
- IMAGEPATH
- IMAGEURL
- IMAGE

See their definition later in the text

Map object

```
MAP
NAME "Efate"
#name of the map file
STATUS ON
# is this map on by default
SIZE 400 400
# size in pixel of the image map
EXTENT 196900 8027100 245000 8073000
# geographic extents of the map
UNITS METERS
# units for the geographic extents
TRANSPARENT OFF
# the background is not transparent
SHAPEPATH "/var/www/html/map/"
# where the shapefiles are stored on the
# server also used for directory reference
IMAGETYPE png24
# type of image output, here PNG format in 24bits color
FONTSET "fonts/fontset.txt"
#  file containing the locations of fonts
```

Web

This group defines where to store information on the server and how to provide it to the web browser

```
WEB
  TEMPLATE /var/www/html/map/map.html
# location of the template for results
# the line needs to be here, but it is
# not used (no file at this location)
  IMAGEPATH /var/www/html/map/images/
# location where to store maps images
  IMAGEURL /map/images/
# web path for the maps images
  LOG /var/www/html/map/maplog
# location of the logfile
END
```

Metadata

VIEW

View allows you to create views so you can quckly move to the geographic location, for instance moving from one country to another one.

WEB
METADATA
VIEW1 "American Samoa,181.496593149,-20.0990853659,199.331777248,-8.22408536586"
VIEW2 "Australia,97.5030380511,-48.984521576,168.843774449,-1.48452157598"
VIEW3 "Cook Islands,180.098325105,-27.4401969982,215.768693304,-3.69019699813"
...
END
...
END

Each view needs to be numbered in sequential order. Each field is separated by commas, the first field is the name of the view, the following fields are minx,miny,maxx,maxy of the view.
Cf. Maps MapView for a detailed description.

Reference

This group defines how the overview should be displayed. You use an image representing the area of interest and you specify the coordinates of the location of the edge of the image. The system will draw on the image a rectangle based on the current view, indicatin the boundaries of such view.

```
REFERENCE
  IMAGE /var/www/html/map/data/efate.png
# location of the image
  EXTENT 196900 8027100 245000 8073000
# extent of the image
  STATUS ON
# the overview is on by default
  COLOR -1 -1 -1
# transparent background
  OUTLINECOLOR 255 0 0
# the color of the outline
  SIZE 95 95
# the size of the image in pixels
END
```

The tip to create an image for overview, is to not worry about the image at the begining. Any image will do. When the layers are implemented in the mapfile and the main view sounds pretty enough, then select a default view, click *Redraw* if neccessary to get the minx, maxx, miny, maxy parameters in the URL, (if necessary adjust them manualy). Save the current map as an image onto your local machine, use an image editing software to resize the image to the ize you want to use in the overview (here 95x95 pixel). Upload the image using the *maps->layer manager* and updat the IMAGE link in the REFERENCE group to point to this new image. Finish by entering the correct EXTENTS that corresponds to the minx,miny,maxx,maxy of the map you used to create the overview.

Legend

The legend is linked to each LAYER by the way each object type is represented (POINT, LINE, POLYGON,...) and its COLOR and OUTLINECOLOR and by the NAME used in the CLASS group. It is important to have a meaningful NAME for each CLASS group in every LAYER group. The NAME of the CLASS can be different from the NAME of the layer as a LAYER can have several CLASSes.

```
LEGEND
  KEYSIZE 18 12
# the size of the object representing the
# geographical object
  LABEL
    TYPE BITMAP
# Font type for the legend name of each
# geographical object
    SIZE MEDIUM
# font size
    COLOR 0 0 89
# font color
  END
  STATUS ON
# legend is on by default
END
```

Scalebar

It is interesting to display a scalebar to get information on distances.

```
SCALEBAR
  IMAGECOLOR 255 255 255
# background color of the image placeholder

  LABEL
    COLOR 0 0 0
# color of the labels indicating the distance
    SIZE SMALL
# size of the labels
  END
  SIZE 350 5
# sixe in pixels of the scalebar
  COLOR 255 255 255
# color used in the scalebar
  BACKGROUNDCOLOR 0 0 0
# background color of the drawn scalebar
  OUTLINECOLOR 0 0 0
# outline color of the salebar
  UNITS kilometers
# units to be used
  INTERVALS 5
# How many intervals in total in the
# scalebar
  STATUS ON
# the scalebar is on by default
END
```

Query

This group secify that the map will be queryable and how the selected objects should be represented. Include this group even if you don't have a LAYER that can be queryable. It will certainly come later.

```
QUERYMAP
  STATUS ON
# the map is queryable by default
  STYLE HILITE
# the selected object is highligthed
END
```

Layers

Vector layer

Let's start by a very simple layer that we will add more features to it. A layer start by the keyword LAYER and finishes by END. It can include CLASS groups, METATADATA groups. For a vector layer the CLASS is compulsory.

Layer Type

The two main types of GIS vector are ESRI shapefiles of Mapinfo TAB files. ESRI shapefiles are natively supported while the Mapinfo filesare supported via the OGR library.

All locations of files must be indicated in relative reference to the path indicated by the keyword SHAPEPATH in the MAP group.

f you have a file in "/var/www/html/map/data/mylayer.shp and SHAPEPATH indicates "/var/www/html/map/" then your file location must be "data/mylayer.shp". Using this convention allows you to easily locate files when you upload them using *Maps->Layer Management*.

File names are case sensitives and spaces in file names must be absoutely avoided. The extensions used for files in a layer must stay with the same case sensitivity. A shapefile is usually made of a shp, idx, dbf files while a Mapinfo layer is made of TAB, ID, MAP, DAT (and sometimes IND) files.

Shapefile

```
LAYER
  NAME "My Layer"
  TYPE LINE
  STATUS ON
  DATA "data/myshapefile.shp"
  CLASS
    COLOR 255 0 0
    NAME "My layer legend"
  END # end of class
END # end of layer
```

In this example we see that the file is located in "data/", it is made of LINE objects that will be displayed with the COLOR red. The layer should be drawn on the map by default (STATUS ON), and it will be indicate as "My Layer" in the layer manager on the map, while being indicated as "My layer legend" in the LEGEND.

Mapinfo TAB

The difference with a shapefile is the use of the OGR library to read the Mapinfo files. The keyword CONNECTIONTYPE OGR must be used and the location of the file is given by the keyword CONNECTION instead of DATA. All the rest stay the same.

```
LAYER
  NAME "My Layer"
  TYPE LINE
  STATUS ON
  CONNECTIONTYPE OGR
  CONNECTION "data/mymapinfofile.TAB"
  CLASS
```

```
    COLOR 255 0 0
    NAME "My layer legend"
  END # end of class
END # end of layer
```

Query

To make the layer queryable add anywehere inside the LAYER object the following lines:

TEMPLATE "query.html"
TOLERANCE 3
TOLERANCEUNITS PIXELS

The first parameters is necessary but does not need to pint to a real file. It is only used outsied tikimaps. The second parameters specify the pointing TOLERANCE in TOLERANCEUNITS, here 3 pixels. If you click on the map all the objects from this layer which are at less than 3 pixels from the click on the image will be selected.
The map is then redraw and at the bottom of the page will be the information related from the object.
Note: if you create a GIS layer which fields contain HTML tags, like for IMG or A (anchor/link), they will be rendered accordingly. This allows you to create a layer pointing to images stored on Tiki.

Labels

You can use querying to know which fields a GIS layer contains. From these fields you can select on to be used a objec labels in the map. For instance you have a GIS layer which contains country names. The name of the country is in the field "NAME". You will use LABELITEM "NAME" to tell the mapserver which field to use for labels.
Inside the CLASS object, you would specify how you want the labels to be rendered. For instance in our example the layer would look like this:

```
LAYER
  NAME "Country Names"
  TYPE POINT
  STATUS ON
  METADATA
     DOWNLOAD "T"
  END
  LABELITEM "NAME"
  LABELCACHE ON
  CONNECTIONTYPE OGR
  CONNECTION "data/Country.TAB"
  CLASS
     SYMBOL 0
     COLOR 0 0 0
     NAME "Country Names"
     LABEL
        COLOR 0 0 0
        FONT arial
        TYPE TRUETYPE
        POSITION CC
        PARTIALS TRUE
        SIZE 7
        BUFFER 1
        OUTLINECOLOR 255 255 255
     END
  END
END
```

n this example LABEL is black (COLOR 0 0 0) using the arial FONT which is a TRUETYPE font. The POSITION of the labe is Center/Center in regard to the POINT object. If an object is not fully on the map, the LABEL is still drawn (PARTIAL TRUE). The SIZE of the label is 7 points. There are no labels closer than 1 pixels from each others (BUFFER 1). For this last parameter to work, you need to enable the LABELCACHE. Finally the label is surrounded by a white outline (OUTLINECOLOR 255 255 255).

Thematic Mapping

Each layer contans one or more CLASS. The CLASS defines how each object should be drawn on the screen. by using CLASSITEM, you can use one field to separate objects in classes. For instance all the bathymetric lines which depth is between -100m and -500m belong to one class while each bathymetric lines which depth is between -500m and -1000m belong to another class. The separation into CLASS is made using an EXPRESSION in each CLASS. The EXPRESSION uses simple logic based on the field in CLASSITEM.

For instance in the example below we use the field "value" which contain the water depth of the LINE object to display this line object in various colors depending of the depth. If the EXPRESSION is true then the parameters in the CLASS apply. A CLASS without an EXPRESSION is a default CLASS for all the objects which have not been classified otherwise

```
LAYER
  NAME "Bathymetry 20m"
  TYPE LINE
  STATUS OFF
  METADATA
    WIKI "FijiBathymetry"
  END
  TEMPLATE "query.html"
  TOLERANCE 3
  TOLERANCEUNITS PIXELS
  LABELITEM "Value"
  CLASSITEM "Value"
  LABELCACHE ON
  CONNECTIONTYPE OGR
  CONNECTION "data/fiji/viti_bathy_contour.TAB"
  CLASS
    SYMBOL 0
    COLOR 0 200 255
    NAME "Bathymetry 2.5m >-50m"
    expression ([Value]>-50)
    LABEL
      ANGLE AUTO
      COLOR 0 0 0
      FONT arial
      TYPE TRUETYPE
      POSITION cc
      PARTIALS FALSE
      BUFFER 5
      SIZE 6
      OUTLINECOLOR 200 200 200
    END
  END
  CLASS
    SYMBOL 0
    COLOR 0 100 255
    NAME "Bathymetry 20m >-500m"
    expression ([Value]<-50 AND [Value]>=-500)
    LABEL
      ANGLE AUTO
```

```
      FONT arial
      COLOR 0 0 0
      TYPE TRUETYPE
      POSITION cc
      PARTIALS FALSE
      BUFFER 5
      SIZE 6
      OUTLINECOLOR 200 200 200
    END
  END
END
```

Metadata

Inside a LAYER group you can have a METADATA group. Some of this metadata is used for special purposes inside tikiwiki. There is only one METADATA group inside a LAYER group.

WIKI

Using WIKI creates a link from the layer name to a wiki page. Use the wiki page to indicate some information on the layer:

- custodian
- ownership
- date of creation
- accuracy
- history
- datum/projection
- interesting layer features
- ...

```
METADATA
 WIKI "MyLayerPage"
END
```

DOWNLOAD

If DOWNLOAD is set to "T" then the files that forms the layer can be downloaded by a registered user.

```
METADATA
 DOWNLOAD "T"
END
```

The system selects all the files with the same base name as defined in the LAYER DATA or CONNECTION clause but with different extension. However if one of these files has the extension NDL, the download is disabled. This allows to upload GIS data that can only be viewable.

Raster layer

The easiest way to handle raster layers is to use Geotiff images which contain projection information. However making geotiff images may need advanced remote sensing software. The other way is to use an additional file which contains information about the coordinates of the pixels in the image file.

When several images are used instead of using a layer for each image they can be tiled. A shapefile is created with a rectangel for each image which helps the system to find the right image for the right location.

Geotiff

Using a geotiff image in a layer is simple, the TYPE RASTER is used with DATA pointing to the tiff file. the keyword OFFSITE is used to define which color in the tiff image should be used for transparency. This is useful when tiling or overlapping several images.

```
LAYER
  NAME "DTM 50m"
  TYPE RASTER
  STATUS OFF
  DATA "data/fiji/VLevudtm.tif"
  OFFSITE 0 0 0
END
```

Image Tiles

Images tiles are created using utility tools from the mapserver software. The utility is called gdaltindex and parses mainly geotiff images to get their boundaries and create a shapefile containing an outline for each of the images. Under Maps->Layer Management, at the bottom of the page a utility is avalilable to generate the shapefile. Basically upload the images to the right directory. The images should have the same name prefix. Then reference all these images with a wildcaard and name the shapefile to be created. For instance you can upload coralcoastsigatoka.tif, coralcoastmomi.tif, coralcoastnavua.tif and reference them as coralcoast*.tif and create the shapefile img_index.shp

```
LAYER
  NAME "Coral Coast 4m IKONOS"
  TYPE RASTER
  METADATA
    WIKI "FijiImagery"
  END
  STATUS ON
  TILEINDEX "data/fiji/img_index.shp"
  TILEITEM "Location"
  OFFSITE 0 0 0
END
```

Grid Layer

A grid layer allows you to draw a grid in the local cordinates on your map. It is useful to find location of objects on the map. For best effect the grid must be the last layer in the mapfile to be drawn the last.

```
LAYER
  NAME "Grid"
  TYPE LINE
  STATUS OFF
  CLASS
    COLOR 0 0 0
    LABEL
      FONT arial     # must be in your FONTSET
      TYPE TRUETYPE
      SIZE 8
      COLOR 0 0 0
      OUTLINECOLOR 255 255 255
    END
  END
  GRID
    MINARCS 2
    MAXARCS 6
  END
END
```