**Revision to GMT 6 APIs**

**1. Revision to GMT APIs rec-by-rec i/o**

Typical data records (ASCII) that GMT modules need to ingest are of the general form

[<*array of doubles*>][<*trailing text*>]

At the moment, many modules (e.g., pstext) must read such files as pure text records and do special formatting to extract the leading coordinates. This makes passing data to these modules from external environments complicated since some modules expect a GMT\_TEXTSET and others expect GMT\_DATASET containers, and in some cases the module expects one or the other depending on module options. We wish to simplify i/o operations for such data records so that we do not have to make special arrangements. On the container level, we have added a ***char \*\*text*** to the GMT\_DATASEGMENT structure. This is NULL if there are no texts, otherwise it is an array with text strings. Likewise, we should be able to read just plain text and have the ***double \*\*data*** array be NULL. For mixed records, both will contain information. This is now implemented for i/o involving GMT\_Read\_Data and GMT\_Write\_Data and GMT\_Get\_Record and GMT\_Put\_Record. These these functions return and accept a ***void \*record*** pointer. We have retained these two function declarations. However, while before the void \*record pointer would either be a character string or an array of doubles, it is now always a pointer to a new internal data structure:

struct GMT\_RECORD {

double \***data;** /\* This is a pointer to the GMT->current.io.curr\_rec array \*/

char \***text;** /\* This is a pointer to the GMT->current.io->start\_of\_text string \*/

};

This change necessitated the following changes in modules using these i/o functions:

1. After In = GMT\_Get\_Record(…), one must use In->data to access coordinates and In->text to get the trailing text. This will eliminate awkward special cases in mapproject, gmtselect and many others where we are trying to guess if input is ASCII text. Modules can easily check if any of these pointers contain information.
2. For output, modules will either pass the ***In*** pointer (e.g., gmtselect) or use another GMT\_RECORD structure for arrays and text that may differ from the input (e.g., mapproject).
3. For the API, some combinations of containers and such data records are not compatible, e.g., GMT\_GRID does not support both data rows and text, but both GMT\_MATRIX and GMT\_VECTOR have a char \*\*text pointer that can handle trailing text. These containers facilitate data input/output from external environments.
4. We expect the GMT\_TEXTSET container to be deprecated as GMT\_DATASET can handle the more flexible format. We will remove it from the documentation.
5. The GMT/MATLAB toolbox is currently using GMT\_TEXTSET. Here we will need to consider if we shall modify what MATLAB passes and receives from GMT.

**Mode Clarifications**

Both GMT\_Get\_Record and GMT\_Put\_Record takes a mode as 2nd argument. For GMT\_Get\_Record the available modes are

GMT\_READ\_DATA Only read the leading numerical columns; ignore trailing text

GMT\_READ\_TEXT Read the entire record as a text string

GMT\_READ\_MIXED Read both leading numerical columns and trailing text

while for GMT\_Put\_Record the available modes are

GMT\_WRITE\_DATA Only write the leading numerical columns and not trailing text

GMT\_WRITE \_TEXT Only write the (trailing) text string

GMT\_WRITE \_MIXED Write both leading numerical columns and trailing text

**2. The Virtual file names in GMT API [PROPOSAL FOR 6.x]**

In GMT5, virtual files were given string names such as @GMTAPI@-######. When the GMT i/o machinery detects such file names, it extracts the internal object number from ######, searches for the corresponding memory object ***S*** in the internal API->object list, and makes numerous checks to determine if the object is the right type for the current context. However, this would be much more efficient if the virtual files instead were called

**@GMTAPI@-<*level*>-<*dir*>-<*actualfamily*>-<*family*>-<*geometry*>-<*messenger*>-########**

so that we can immediately discover the container type, geometry, etc., without having to go deep into the API and obtain the object first. Since API users are not generating these names manually I don't think there will be any backwardness issues, and nobody will be able to link the GMT6 lib with GMT5-compatible codes.  Furthermore, the @GMTAPI@ encodes/decodes are all done by internal functions, including the API function GMT\_Encode\_ID, which no longer should be part of the documented API (it should quietly be deprecated since the GMT\_\*\_VirtualFile functions handle these assignments under the hood).  Finally, because I foolishly chose the stupid *enum* name GMT\_STR16 to hold the length of virtual file names (which is presently 16), we would simply introduce a better name such as GMT\_VF\_LEN = 32 and let GMT\_STR16 be redefined as 32 as well for backwardness.

In summary, the various parameters to be encoded are:

level: The level indicates if the file was provided as a primary (P) or secondary (S) input/output. Primary items are given on the command line and correspond to the main inputs to a module. For output this may be the default primary output of a module (e.g., stdout). Internally, the variable S->module\_input is used for checks. If not known then we default to U (undefined).

direction: Indicates if this object is serving as input (I) or output (O) to the receiving module.

actualfamily: Holds the actual family of the container: C, D, G, I, M, P, V.

family: The container type may be of the given actual family, but we may intend for it to be used as the datatype given by family. Often these are the same, but for instance when the container is a matrix (actualfamily = M) destined for plotting lines in **plot** then the family is a dataset (D) despite the nature of the container.

geometry: One of the allowed geometries, such as point (T), line (L), polygon (P) any of these (A), surface (G), none (N), or possibly undefined (U).

messenger: Output containers that are just dummies and will be replaced by a brand-new container in the module are called messengers, so this field is either yes (Y), no (N), or undefined (U).

######: The running object ID number, ≥ 0.

E.g., a matrix holding line segments that is passed to ***plot*** (which expects datasets) would have the virtual file name

@GMTAPI@-P-I-M-D-L-N-00000001

and a simple parser will immediately know what we have.