**GMT\_Encode\_Options**

GMT syntax has evolved over decades and predates our attempts at building an API and accessing GMT from external interfaces, such as MATLAB or Python. This means there are some problems we have to overcome without redesigning all of GMT, pissing off all our users in the process. Key among our problems is the ability to determine what kind of data a module wants and how it wants it. Also, since the data we wish to pass from the external interface to GMT will already be in memory it places some limitations on the syntax we can use. Each module communicates this information to the outside via its *keys*. The keys are sets of 3-letter words (usually, but they can be longer) that encode much of this information. We will think of a key as the three-letter combination **XYZ** and examine what each letter stands for. Once this is clear we will examine special cases for these letters and finally see why we will go beyond three letters for more complicated scenarios.

**0. The context**

Before we discuss the keys and they purposes, remember that we are no longer using the UNIX command line but instead calling GMT from some external interface, and we will do so with the API call

EXTERN\_MSC int GMT\_Call\_Module (void \*API, const char \*module, int mode, void \*args);

We give the module name in quotes and typically we pass mode as GMT\_MODULE\_CMD and args as a command-line string. To give a simple example, imagine we want to run blockmean on the dataset in the file raw.txt and save the output to file ave.txt, for the given region and increment. This would work (we ignore error checking here):

GMT\_Call\_Module (API, “blockmean”, GMT\_MODULE\_CMD,”raw.txt -R30/70/-20/50 -I2 > ave.txt”);

Our task is to understand how this command would change if we already had the data from raw.txt in memory (say, in a n x 3 double precision matrix) and we actually want to obtain the blockmean’ed results back as another m x 3 double precision matrix in memory, without going through intermediate file i/o. We do this by using GMT’s *memory files*, which is our bookkeeping scheme of assigning fake filenames to memory locations and passing those filenames to modules as if they were real files. Because the GMT API understands what these means we are able to associate the memory locations with the fake filenames. A memory file may be named @GMTAPI@-000012. The trick is that the modules themselves are unaware of what is going on as all the i/o is handled by the GMT API. By *registering* the memory locations with the API we obtain *ID* values and these can be formatted to memory file names and used accordingly.

If we just wanted to call GMT from a C or Fortran program then this is all we would need to know to get started. However, calling GMT from MATLAB, Octave, Julia, Python, R, etc. involves more work since we need to provide an intermediate layer between the GMT API and the external environment. This layer is a piece of C code that handles the complexities of the interface. For instance, in the case of MATLAB this is done by gmtmex.c (which relies on gmtmex\_parser.c) and is called by gmt.m from MATLAB. Other interfaces may be similarly designed. Unlike programmers in C and Fortran, users of MATLAB or Python have far less experience and we thus wish to hide much of the complexity alluded to here. This is where the keys come in to allow MATLAB users with no understanding of C to have full access to all of GMT. This means we need to create valid GMT\_Call\_Module calls and take care of passing data in and out.

**1. Simple key cases.**

**X** represents a specific module option (e.g., **L** for **–L**, **F** for **–F**, and so on). For command-line input files (i.e., data-tables, text-tables or grids) we use the option **<** while for data written to *stdout* we use the option **>**. [Note these are actual options in GMT as you can give ‘**–>**test.txt’ and see that a module will write to test.txt.]

**Y** represents the data type, with **C** = CPT, **D** = Dataset/Point, **L** = Dataset/Line, **P** = Dataset/Polygon, **G** = Grid, **I** = Image, **T** = Textset, and **X** = PostScript. If the geometry of a dataset is not known (or does not matter) then we still just use **D**. In addition, we sometimes will specify the type as **?**, and this mark means the actual data type is not known until run-time and will be specified via a module option [more on this later].

**Z** holds the code signifying direction. We distinguish between *primary* inputs '**{**', primary output '**}**', *secondary* input '**(**', and secondary output '**)**'. On the UNIX command line, a module’s primary inputs and outputs are *required* items and must always be provided. If not given their absence will result in a syntax error. However, the external APIs (MATLAB, Python) can override this behavior and supply the missing sources or destinations via memory references that supply the inputs or accept the outputs. Secondary inputs means they should only be assigned if an *explicit* option is given. If the input or output designation is irrelevant for an option we use '-'.

Before we proceed we will give some example of these simple cases. Consider the module blockmean: Its synopsis says

**usage: blockmean [<table>] -I<xinc>[<unit>][=|+][/<yinc>[<unit>][=|+]]**

**-R<west>/<east>/<south>/<north>[r] [-C] [-E[p]] [-S[m|n|s|w]] [-V[<level>]] [-W[i][o]]**

**[-a<col>=<name>[,...]] [-b[i|o][<ncol>][t][w][+L|B]] [-d[i|o]<nodata>] [-f[i|o]<info>]**

**[-h[i|o][<nrecs>][+c][+d][+r<remark>][+t<title>]] [-i<cols>[l][s<scale>][o<offset>][,...]]**

**[-o<cols>[,...]] [-r] [-:[i|o]]**

Most of these options modify behavior and only a few specify inputs and outputs. In fact, most of that again is hidden. We see that [<table>] is an optional input, since it can be specified by one or more data files or be served via redirection from a pipe. As for output we know the results are simply written to *stdout* and it is up to the user to redirect that elsewhere. There are no other data associated with blockmean, except there is the possibility that a user may specify the region (**–R**) by using an grid (e.g., –Rmy\_data.grd). Now consider having both your input (x,y,z) data and that grid in memory in MATLAB or Python and wanting to use that with blockmean. That external interface needs to know what blockmean can handle, and blockmean states this information in its keys near the top of blockmean.c:

#define THIS\_MODULE\_KEYS "<D{,>D},RG-"

This information is available to GMT\_Encode\_Options and returned as the above string. Let us examine the three keys in turn:

1. **<D{**, according to the simple rules above, says that data are of the *dataset* type and is a required (*primary*) *input* that must be provided. In the context of the external interface this means if you did not give a filename via the command string you passed to GMT\_Call\_Module then GMT\_Encode\_Options will compare your options to the keys, detect the discrepancy, and add a *marker* to represent the missing data set. The marker is typically a special character, such as **$**, and it will be replaced by an actual memory-file by the calling environment before GMT\_Call\_Module is let loose.
2. **>D}** says that this module will write a *dataset* to *stdout* and that this is a *required* task. Hence, you will need to make provisions to capture this data into memory since no redirection is available. As for input, if you fail to provide an output file then GMT\_Encode\_Options will add one via a marker. Now, you may say “but blockmean does not have an output option, it just writes to stdout” but you would be wrong. The output option for blockmean, blockmedian, and filter1d (and many other similar modules) is the option **–>***filename,* but on the UNIX command line we typically use the conventional form “> filename” or “| another-process”.
3. **RG-** says that if the **–R** *option* is given then we may allow for the possibility that the region may be given via a *grid* instead of a hardwired w/e/s/n string. The **R** is the option, **G** is the data type, and **-** says the direction is irrelevant (while we are obviously *reading* the grid we are not using it as data).

Many modules are similar to blockmean and thus have similarly very simple keys.

2**. More complicated XYZ key cases.**

Before we extend the rules for the keys we will discuss some of the reasons why the above is not enough. Here are some cases that we could not handle with what we have introduced so far.

* pscoast usually writes *PostScript* to *stdout* (so one key would be **>X**}) but if **–M** is given the we actually write a data set instead.
* psxy expects primary input data set (so one key would be **<D{**) but when **–T** is given it should not read any data.
* The special modules read and write (which are only available via the API) does not know what data type they handle and hence we use ? in the keys, but at some point we need to know what we actually will read or write.
* gmt2kml expects to read a text set but if options **–i** or **–b** are selected then it actually expects to read a dataset.

We handle these complications with *magic* keys. Each of the letters **X**, **Y**, and **Z** can take on special values that triggers more complicated behavior:

**X** is **-** (hyphen). This means the default primary input or output (the **Z** setting determines if it is input or output) has a data type that is not known until runtime. A module option will specify which type it is, and this option is encoded in the **Y** letter. So a **–Y**<type> option is *required* and that is how GMT\_Encode\_Options can update the primary data type. As an example, the special module read can read any of the six GMT objects but requires the option **–T**<type>. It thus has the keys "<?{,>?},-T-". Hence, we examine **–T**<type> and replace ? with the dataset implied by <type> both for input *and* output (since **Z** was not indicating a particular direction). Use i|o if only input or output should have this treatment.

**Y** is **-** (hyphen). If the **–X** option is given then either the input or output (depending on the direction specified by **Z**) will NOT be required. As an example of this behavior, consider psxy, which has a **–T** option that means "read no input, just write the *PostScript* trailer". So the key "**T-(**" in psxy means that when **–T** is used then *no* input is required. This implies that the primary input key "**<D{**" is changed to become "**<D(**" (secondary, hence not required) and no attempt is made to connect external input to the psxy input. If **Z** is none of () then we expect **Z** to be one of the options with required input (or output) and we change that option to supplemental input (or output). Example: grdtrack has two required inputs (the grid(s) and the track/point file). However, if **–E** is set then the track/point file is not needed so we need to change it to secondary. We thus add **E-<** which then will change **<D{** to **<D(**.

**Z** is a letter not among **{(})-**. This means that normally these modules will produce the output type specified by the primary setting, but if the "**–Z**" option is given the primary output type will be changed to the type given by **Y**.

Before we go any further let us examine some keys that implement these behaviors:

* pscoast normally writes *PostScript* but pscoast **–M** will instead export coastline data to *stdout*, thus its keys must contain the entry "**>DM**", which means that when **–M** is active then the default *PostScript* output key "**>X}**" morphs into "**>D}**" and thus allows for data set export instead.
* grdinfo normally writes a text set (key "**>T}**") but if **–C** is given then it should write a dataset instead. It uses the magic key "**>DC**": If the **–C** option is given then the default primary output key "**>T}**" is morphed to become "**>D}**".
* grdcontour normally writes *PostScript* but grdcontour **–D** will instead export a data set to a file specified by **–D**, so its key contains the entry "**DDD**": When **–D** is active then the default *PostScript* output key "**>X}**" morphs into "**DD}**" and thus allows for a data set export instead.

3**. XYZ modifiers.**

Because of our GMT syntax, there are more complicated behavior not addressed so far, hence we allow for modifiers to be added to the keys. The full syntax for this scheme is

**XYZ+abc...-def...**

The modifiers are listed as one or more letters following a plus sign (+) and one or more letters following a minus sign (-). Either set is optional.

3**.1 Enhanced X magic.**

3**.2 Enhanced Y magic.**

3**.3 Enhanced Z magic.**

Consider the case of the magic **Z** discussed earlier. When modifiers are involved we perform the substitution of output type to **Y** *only* if all of these statements are true:

1. The **–Z** option is given. This was the only situation we studied earlier.
2. The **-Z** option contains *all* of the modifiers **+a**, **+b**, **+c**, and so on.
3. The -**Z** option contains *at least one* of the modifiers **+d**, **+e**, **+f**, and so on.

This enhanced **Z** magic is a bit confusing so here are a few examples:

* gmtspatial : This module normally reports a data set. However if **–N+r** is given there will be no data output, just a report on *stdout*. Consequently, the key “”**>TN+r**” means if **–N**[...]**+r** is given (that is, **–N** has arguments and the modifier **+r** is found somewhere among the arguments) then enforce **>T}**. In this case just giving **–N** without the given modifier **+r** would *not* trigger the change and data output would still take place.
* pscoast: As we have seen, this module typically writes PostScript to *stdout* or a dataset if **–M** was given. However, it may also utilize the DCW dataset to determine a suitable region rectangle for a specified landmass, such as France, augmented by a 1 degree padding by passing the option **-EFR+R**1**+w**. Hence, the key "**>TE+w-rR**" means if **–E** is given with modifier **+w** and one of **+r** or **+R** are present then enforce **>T}**, which will change the output type to textset, suitable for the string that we will return, which in this case would be -R-6.142231/10.561553/40.338783/52.092812.