Chapter 3 Basic Operations

This chapter explains set–up steps and describes basic GT.M and system management operations. It is designed to get you started quickly, and provides the basic information required to run and stop GT.M efficiently.

NOTE: Most of the examples in this chapter assume you have added **`gtm_dist`** to your PATH, as described in the "Defining the Environment Variables" section in the following text.

Before You Start

Before starting GT.M at any time:

- The **`gtm_dist`** environmental variable must be defined.
- Sanchez Computer Associates strongly recommends the definition of the **`gtm_log`**, **`gtmgbldir`**, and **`gtmroutines`** environmental variables.

Defining the Environment Variables

The four environment variables GT.M used for normal operation are:

- **`gtm_dist`**
- **`gtm_log`**

- **`gtmgbldir`**
- **`gtmroutines`**

The required **`gtm_dist`** environment variable specifies the path to the directory containing the GT.M system distribution. Although **`gtm_log, gtmgbldir`**, and **`gtmroutines`** are not required for GT.M to operate, Sanchez recommends defining these variables. The variables help ensure that intended database files are accessed and intended directories used for the creation of files generated by GT.M.

gtm_dist

The **`gtm_dist`** environment variable specifies the path to the directory containing the GT.M system distribution. **`gtm_dist`** must be defined for each user. You may want to define **`gtm_dist`** in your login file, or as part of the default system environment.

You can define **`gtm_dist`** as follows:

```
$ gtm_dist=<distribution directory>
$ export gtm_dist
```

For example, if the GT.M distribution is in **`/usr/local/gtm`**, enter:

```
$ gtm_dist=/usr/local/gtm
```

`gtm_dist` is used to establish communications with the GT.M **`gtmsecshr`** daemon and for the location of GT.M components.

Add **`$gtm_dist`** to your PATH as follows:

```
$ PATH=$PATH:$gtm_dist
$ export PATH
```

gtm_log

The **`gtm_log`** environmental variable specifies a directory where the **`gtm_secshr_log`** file is stored. The **`gtm_secshr_log`** file stores information gathered in the **`gtmsecshr`** process. Sanchez Computer Associates recommends that a system–wide default be established for **`gtm_log`** so that **`gtmsecshr`** always logs its information in the same directory, regardless of which user's GT.M process invokes **`gtmsecshr`**.

gtmgbldir

The **`gtmgbldir`** environment variable identifies the Global Directory to be used. A Global Directory maps global variables to physical database files, and is required to access M global variables. If you maintain multiple Global Directories, you must choose one to use. To automate this definition, define **`gtmgbldir`** in your login file.

The following is an example of a **`gtmgbldir`** definition:

```
$ gtmgbldir=/usr/staff/mumps.gld
$ export gtmgbldir
```

gtmroutines

The **`gtmroutines`** environment variable specifies a directory search list of possible locations for M routines. This search enables GT.M to find the routine (program) you want to run even if it is not in your current working directory. Sanchez Computer Associates recommends the inclusion of **`$gtm_dist`** in the list of directories specified in **`gtmroutines`** permits access to the Global Directory Editor (GDE) and to utility routines supplied with GT.M.

The following is an example of a **`gtmroutines`** definition, assuming that **`$ gtm_dist`** has been defined:

```
$ gtmroutines=”. $gtm_dist”
$ export gtmroutines
```

This specifies that GT.M first search for a routine in the current directory, then in the distribution directory (included in the list because it contains the percent routines). You will probably want the search list to contain these at a minimum; in addition, you may also want to include other directories. For details on **gtmroutines** syntax, refer to the "$ZROUTINES" section of the "Intrinsic Special Variables" chapter of the *GT.M Programmer's Guide*.

Providing Access to GT.M

To set up a user's default environment, run **`gtmbase`** from that user's directory, supplying the path to the directory in which GT.M was installed:

```
$ /<path>/gtmbase
```

`gtmbase` copies the default Global Directory **`mumps.gld`** to the current directory and creates a new database in the current directory. It also updates the user's **`.profile`** file (**`.cshrc`**, if using the C shell), setting up the GT.M default path names and aliases. After this is done, you can run GT.M administration scripts on your own directory, without a full path name.

You can also establish the defaults without running **`gtmbase`** by adding the following line to your **`.profile`** (**`.cshrc`**) file. In the Bourne shell this line is:

```
.  <gtm_dist pathname>/gtmprofile
```

In the C shell, the line is:

```
source <gtm_dist pathname>/gtmcshrc
```

In both cases, specify the required full path name to the directory where GT.M was installed.

You can simplify the use of GT.M by defining all the environment variables and aliases in your **`.profile`** file. Production application users generally need most of these commands in their **`.profile`** file as well.

In your **`.profile`** file, replace the options shown in the following **`.profile`** file excerpt as **`<dist dir area>`** and **`<global directory path and file name>`**.

```
gtm_dist=<dist dir area>
export gtm_dist
gtmroutines=”. $gtm_dist”
export gtmroutines
gtmgbldir=<global directory path and file name>
export gtmgbldir
PATH=$PATH:$gtm_dist
export PATH
```

Replace **`<dist dir area>`** with the directory in which GT.M is installed.

Running GT.M

This release of GT.M has three modes: compiler, direct, and auto-start. Invoke these modes by using the **`mumps`** command issued with an appropriate argument.

- To operate in compiler mode, invoke GT.M by entering the **`mumps`** command with a list of file names to compile. GT.M then compiles the specified programs into **`.o`** files. UNIX wildcards (* and ?) are acceptable with the file names.

- To operate in direct mode, invoke GT.M by entering the **`mumps`** command with the **`–direct`** argument. GT.M then enters direct mode, where you can enter M commands interactively.

- To operate in auto-start mode, invoke GT.M by entering the **`mumps`** command with the **`–run`** argument. The next argument is taken to be an M entryref, and that routine is automatically executed, bypassing direct mode. Depending on the shell you are using, you may need to put the entryref in quotes.

When executing M programs, any called programs are incrementally linked. For example, the command

```
GTM> d ^TEST
```

links the object file **`TEST.o`** and executes it; if the TEST program calls other M routines, those are automatically compiled and linked.

NOTE: Depending on the platform, the format of M object modules may not be the standard object file format, so you may not be able to use the UNIX **`ld`** utility to link all M **`.o`** files together. GT.M, however, automatically links and executes these files.