How to Download, Build, and Install MB-System

David W. Caress
November 5, 2023

General Advice for Building and Installing MB-System

MB-System is an open source software package intended for Unix-like operating systems which is distributed as source code. In order to install MB-System, a source distribution must be downloaded, unpacked, compiled, linked, and installed locally. MB-System now includes two methods for building and installing the software, the first based on GNU Autotools and the second using the CMake package.

The GNU Autotools build system includes a script named **configure** at the top of the distribution structure which, when executed, generates a set of files named "Makefile" throughout the structure that hold compilation and linkage instructions used by the program **make**. These compilation and linkage instructions vary depending on the operating system and which versions of prerequisite software packages are installed, and in some cases special arguments must be included in the configure command. Once configure has been successfully executed, running the program **make** at the top level then actually invokes the local compiler and linker to build and install MB-System. Several Autotools programs, including autoconf, automake, and libtool, are used to generate the configure script included in each distribution. This build system has been the sole means by which MB-System could be built since 2011.

The sequence of commands by which one builds and installs a package using CMake is very different from Autotools. One creates a new directory to hold the package build (commonly named "build" by convention) in the top level of the source structure, then cd's into that directory, and then executes **cmake** with the argument "..", which provides **cmake** the path to the top level of the source directory. **Cmake** then copies the source into a build tree and generates Makefiles in that tree to enable compilation, linkage, and installation, again using the program **make**.

The CMake build system is new to MB-System with version 5.8.0, and is intended to replace the older Autotools approach for current and future operating systems. The old Autotools build system will continue to be included in MB-System distributions for the forseeable future to enable building and installation on legacy operating systems. The Autotools build system will not be maintained to enable it's use on current or future operating systems, and the CMake build system will not be modified to support building on legacy systems.

MB-System depends on a number of other software packages. For some operating systems, special arguments to the configure script are required to integrate the MB-System source to prerequisite software packages.

Among the software packages that are prerequisite for MB-System are:

- GMT (Generic Mapping Tools)
- Proj
- GDAL
- netCDF
- FFTW (Fastest Fourier Transform in the West)
- X11
- OpenMotif
- OpenGL
- OpenCV
- PCL (Point Cloud Library) (planned, not yet in release 5.8.0)

The sections below provide some instructions for building MB-System on a few common operating system distributions, including MacOs, Ubuntu Linux, Debian Linux, and the CygWin environment on Windows. These instructions include the installation of the prerequisite software packages using a package manager relevant to each OS and the special arguments needed for the configure script.

How to download an MB-System source distribution

The source code for MB-System is available from a repository on Github:
https://github.com/dwcaress/MB-System

This link will open the page for the most recent stable release of MB-System. The source code distribution can be downloaded as zip or tar.gz archives from the "Assets" section at the bottom.
https://github.com/dwcaress/MB-System/releases/latest

This link will open a page listing all recent MB-System releases, with the most recent release at the top. The most recent release could be a beta or pre-release. The source code distribution can be downloaded as zip or tar.gz archives from the "Assets" section at the bottom.
https://github.com/dwcaress/MB-System/releases/releases

This link will download the current state of the master branch of the MB-System repository:
https://github.com/dwcaress/MB-System/archive/refs/heads/master.zip

---

MacOS 13 Ventura (and MacOs 10 High Sierra, 11 Big Sur, and 12 Monterey)

The use of MacPorts to install the MB-System prerequisite packages is recommended on Apple Mac computers, particularly because this approach enables cleanly building with a complete X11 + Motif + OpenGL infrastructure separate from the libraries and header files associated with the XQuartz package. One usually still installs XQuartz and uses it as the X11 display server, but confining the MB-System compilation and linking to headers and libraries within the MacPorts structure avoids several issues. At present, we are not able to successfully run the MB-System graphical utilities when the prerequisite packages have been installed using the Fink or Homebrew package managers.

This example is relevant for MacOs 10.13 High Sierra to the current MacOs 13 Ventura on both Intel and ARM (Apple Silicon) architecture computers.

- Install Xcode and the Xcode command line tools, which includes the LLVM compiler suite.

- Install the XQuartz X11 server from
  https://www.xquartz.org
  XQuartz 2.8.5 or later is required for all MB-System installations.

- Install MacPorts using the appropriate downloadable installer package from:
  https://www.macports.org/install.php

- After MacPorts installation, first make sure the default port packages are current by running selfupdate and then install the MB-System prerequisites.

  ```
  sudo port -v selfupdate
  sudo port install gmt6 fftw-3 mesa libGLU openmotif opencv4
  ```

- Also make sure that a current version of Python3 is available. First list

the available Python3 versions, install the most recent, and then set port
to link that version to python3:

```
port select --list python
```

Versions of Python available through MacPorts (November 6, 2023):

- python27
- python310
- python311
- python38

The most recent version is python311, so install it and make it the active installation:

```
sudo port install python311
sudo port select --set python python311
sudo port select --set python3 python311
```

Also install the most recent Python imaging library Pillow:

```
sudo port install py311-Pillow
```

- Download the MB-System source package from the repository at GitHub:
  https://github.com/dwcaress/MB-System
  There are often beta releases that are more recent than the current stable release. The most recent beta release will be at the top of this page:
  https://github.com/dwcaress/MB-System/releases
  We recommend using Chrome for the download rather than Safari, because Chrome does not prematurely unpack the tarball. Although MB-System can be installed from a source tree located anywhere, we generally place the tarball in /usr/local/src and then unpack and build MB-System there.

- Unpack the MB-System distribution tarball

```
cd /usr/local/src
tar xvzf MB-System.5.7.9.tar.gz
```

and then cd into the top directory of the resulting structure:

```
cd MB-System-5.7.9
```

- Option 1: **CMake Build System**
  At that location, create a working directory named "build", **cd** into "build", and then execute **cmake**. This command should successfully enable building the entire current MB-System (5.8.0 or later) on any Mac computer with the prerequisites installed through MacPorts. This has been tested with computers running Ventura.

  ```
  mkdir build
  cd build
  cmake ..
  ```

- Option 2: **Autotools Build System**
  At that location, execute the configure script, named **configure**, with the options necessary for your context. The XCode compiler tools do not look for header files or libraries in the locations used by MacPorts, and so it is necessary to specify these locations for several of the prerequisite packages. This command should successfully enable building the current core MB-System (5.7.9 or later) on any Mac computer with the prerequisites installed through MacPorts. This has been tested with computers running Ventura and Monterey.

  ```
  ./configure \
    --prefix=/usr/local \
    --disable-static \
    --enable-shared \
    --enable-hardening \
    --enable-test \
    --with-proj-lib=/opt/local/lib/proj9/lib \
    --with-proj-include=/opt/local/lib/proj9/include \
    --with-gmt-config=/opt/local/lib/gmt6/bin \
    --with-fftw-lib=/opt/local/lib \
    --with-fftw-include=/opt/local/include \
    --with-x11-lib=/opt/local/lib \
    --with-x11-include=/opt/local/include \
    --with-motif-lib=/opt/local/lib \
    --with-motif-include=/opt/local/include \
    --with-opengl-include=/opt/local/include \
    --with-opengl-lib=/opt/local/lib \
    --with-otps-dir=/usr/local/src/otps
  ```

  The MB-System codebase includes some components that are optional when using the Autotools build system, such as OpenCV based photomosaicing (enabled with --enable-opencv) and a realtime Terrain Relative Navigation infrastructure and toolset (--enable-mtrn and --enable-mbtnav). This configure command should enable building the entire MB-System package, including these optional tools:

```
./configure \
  --prefix=/usr/local \
  --disable-static \
  --enable-shared \
  --enable-hardening \
  --enable-test \
  --with-proj-lib=/opt/local/lib/proj9/lib \
  --with-proj-include=/opt/local/lib/proj9/include \
  --with-gmt-config=/opt/local/lib/gmt6/bin \
  --with-fftw-lib=/opt/local/lib \
  --with-fftw-include=/opt/local/include \
  --with-x11-lib=/opt/local/lib \
  --with-x11-include=/opt/local/include \
  --with-motif-lib=/opt/local/lib \
  --with-motif-include=/opt/local/include \
  --with-opengl-include=/opt/local/include \
  --with-opengl-lib=/opt/local/lib \
  --enable-mbtrn \
  --enable-mbtnav \
  --enable-opencv \
  --with-opencv-include=/opt/local/include/opencv4 \
  --with-opencv-lib=/opt/local/lib/opencv4 \
  --with-otps-dir=/usr/local/src/otps
```

- Once the makefiles have been generated by cmake or configure, build and install MB-System using:

```
make
make check
sudo make install
```

Unless one specifies otherwise, the MB-System executable programs, libraries, and header files will be installed in directories named /usr/local/bin /usr/local/lib and /usr/local/include respectively. In order for the MB-System programs to execute from a command line, the /usr/local/bin directory must be included in a user's $PATH environment variable. This is typically accomplished by including a statement like:

```
 export PATH=/usr/local/bin:$PATH
```

in an environment file in the user's home directory named .zshenv (if using the zsh shell), or .profile (if using the bash shell). Other user environment files can be used, such as .zshrc for zsh or .bashrc for bash.

When one updates to a new MB-System version, we recommend uninstalling the previous version before installing the next. In the directory for the previous version (e.g., /usr/local/src/MB-System5.7.9beta52 if using Autotools or /usr/local/src/MB-System5.7.9beta52/build if using CMake), use the following command whether using the Autotools or the CMake build system:

```
    sudo make uninstall
    sudo make clean
```

Then follow the steps above starting with downloading the next version from GitHub.

Ubuntu Jammy Jellyfish 22.04

Install Ubuntu 20 from ISO, and then update the starting packages: sudo apt upgrade

Install compilers: sudo apt install build-essential

Install MB-System prerequisites: sudo apt install libnetcdf-bin libnetcdf-dev libgdal-dev \ gmt libgmt6 libgmt-dev libproj-dev \ libfftw3-3 libfftw3-dev libmotif-dev \ xfonts-100dpi libglu1-mesa-dev \ libopencv-dev gfortran

Run configure to build all of MB-System: export LD*LIBRARY*PATH=/usr/local/lib:$LD*LIBRARY*PATH LDFLAGS="-Wl,-rpath -Wl,/usr/local/lib" \ ./configure \ --enable-mbtrn --enable-mbtnav --enable-opencv \ --with-opencv-include=/usr/include/opencv4 \ --with-opencv-lib=/lib/x86_64-linux-gnu

Build MB-System: make make check sudo make install

Post-Installation Actions: cpan Parallel::ForkManager gmt gmtset IO*NC4*CHUNK*SIZE classic gmt gmtset GMT*CUSTOM_LIBS /usr/local/lib/mbsystem.so

Ubuntu Focal Fossa 20.04

Install Ubuntu 22 from ISO, and then update the starting packages: sudo apt upgrade

Install compilers: sudo apt install build-essential

Install MB-System prerequisites: sudo apt install libgdal-dev \ gmt libgmt6 libgmt-dev libproj-dev \ libfftw3-3 libfftw3-dev libmotif-dev \ xfonts-100dpi libglu1-mesa-dev libopencv-dev \ gfortran

Run configure to build all of MB-System: export LD*LIBRARY*PATH=/usr/local/lib:$LD*LIBRARY*PATH LDFLAGS="-Wl,-rpath -Wl,/usr/local/lib" \ ./configure \ --enable-mbtrn --enable-mbtnav --enable-opencv \ --with-opencv-include=/usr/include/opencv4 \ --with-opencv-lib=/lib/x86_64-linux-gnu

Build MB-System: make make check sudo make install

Post-Installation Actions: cpan Parallel::ForkManager gmt gmtset IO*NC4*CHUNK*SIZE classic gmt gmtset GMT*CUSTOM_LIBS /usr/local/lib/mbsystem.so

---

Ubuntu Bionic Beaver 18.04

Note that the photomosaicing tools utilising OpenCV cannot be built under Ubuntu 18.

Install Ubuntu 22 from ISO, and then update the starting packages: sudo apt upgrade

Install compilers: sudo apt install build-essential

Install MB-System prerequisites: sudo apt install libgdal-dev \ gmt libgmt5 libgmt-dev gmt-common proj-bin proj-data libproj-dev \ libfftw3-3 libfftw3-dev libmotif-dev \ xfonts-100dpi libglu1-mesa-dev \ gfortran

Run configure to build all of MB-System: export LD*LIBRARY*PATH=/usr/local/lib:$LD*LIBRARY*PATH LDFLAGS="-Wl,-rpath -Wl,/usr/local/lib" \ ./configure \ --enable-mbtrn --enable-mbtnav

Build MB-System: make make check sudo make install

Post-Installation Actions: cpan Parallel::ForkManager gmt gmtset IO*NC4*CHUNK*SIZE classic gmt gmtset GMT*CUSTOM_LIBS /usr/local/lib/mbsystem.so

---

Debian 12

Install Debian 12 from ISO, and then update the starting packages: sudo apt upgrade

Install compilers: sudo apt install build-essential

Install MB-System prerequisites: sudo apt install netcdf-bin libnetcdf-dev libgdal-dev \ gmt libgmt6 libgmt-dev libproj-dev \ libfftw3-3 libfftw3-dev libmotif-dev \ xfonts-100dpi libglu1-mesa-dev \ libopencv-dev gfortran

Run configure to build all of MB-System: export LD*LIBRARY*PATH=/usr/local/lib:$LD*LIBRARY*PATH LDFLAGS="-Wl,-rpath -Wl,/usr/local/lib" \ ./configure \ --enable-mbtrn --enable-mbtnav --enable-opencv \ --with-opencv-include=/usr/include/opencv4 \ --with-opencv-lib=/lib/x86_64-linux-gnu

Build MB-System: make sudo make install

Post-Installation Actions: cpan Parallel::ForkManager gmt gmtset IO*NC4*CHUNK*SIZE classic gmt gmtset GMT*CUSTOM_LIBS /usr/local/lib/mbsystem.so

CygWin

CygWin is a collection of software tools that augment Windows computers with a Unix-style environment within which one can build, install, and run Unix-y packages like MB-System.

If Cygwin is installed, then one must install a number of prerequisite packages before building MB-System. These include:

```
gcc, g++, rpc-devel, gambas3-devel, libproj-devel, libproj12, libnetcdf-dev
```

Run configure to build MB-System without any graphical tools:: ./configure --enable-mbtrn --enable-mbtnav --disable-dependency-tracking --disable-mbtools

Build MB-System: make sudo make install

Post-Installation Actions: cpan Parallel::ForkManager gmt gmtset IO*NC4*CHUNK*SIZE classic gmt gmtset GMT*CUSTOM_LIBS /usr/local/lib/mbsystem.so

Docker Container with MB-System

An updated MB-System Docker Image is generated each time that a new release is created in the MB-System Github repository. This Docker is based on CentOs 7, and can be run on MacOs, Linux, and Windows computers. Data present on the host computer's filesystems can be processed using the MB-System programs in the Docker container.

The MB-System docker image is available at https://hub.docker.com/r/mbari/mbsystem

Documentation is available at: https://github.com/dwcaress/MB-System/tree/master/docker/user https://github.com/dwcaress/MB-System/blob/master/docker/user/README-win11.md

---

The MB-System build system based on GNU Autotools was begun by Bob Covill in 2011, and then completed through a distributed, multi-continental effort by Bob Covill, Christian Ferreira, Hamish Bowman, Kurt Schwehr, and David Caress during 2013.

To use the build system...

Obtain the MB-System source tree by either downloading and unpacking an MB-System source distribution tarball (e.g. mbsystem-5.7.9.tar.gz) from the Github repository: https://github.com/dwcaress/MB-System/releases

To generate the makefiles needed to build MB-System, in a shell cd to the top of the MB-System source tree and run ./configure with the options appropriate for your situation. Some examples are given below.

After configure you can run the make utility in the usual fashion make make install

Some other useful make commands include: make check (build and execute a limited set of unit tests) make clean (to delete compiled object files in the source code tree) make uninstall (to fully uninstall the installed libraries, headers, and programs)

Configure script command line options:

Installation location:

--prefix - This is the common installation prefix for all files. If exec*prefix is defined to a different value, prefix is used only for architecture-independent files. [Default: /usr/local] --exec*prefix - The installation prefix for architecture-dependent files. By default it's the same as prefix. You should avoid installing anything directly to exec*prefix. However, the default value for directories containing architecture-dependent files should be relative to exec*prefix. [Default: ${prefix} ==> /usr/local] --datarootdir - The root of the directory tree for read-only architecture -independent data files. [Default: ${exec*prefix}/share ==> /usr/local/share] --bindir - The directory for installing* *executables that users run. [Default: ${exec*prefix}/bin ==> /usr/local/bin] --libdir - The directory for installing object code libraries. [Default: ${exec*prefix}/lib ==> /usr/local/lib] --includedir - The directory for installing C header files. [Default: ${exec*prefix}/include ==> /usr/local/include]

MB-System configure options:

--prefix=install - location for mbsystem (/usr/local/mbsystem) (optional) --enable-hardening - Enable compiler and linker options to frustrate memory corruption exploits (e.g. -fPIE and -pie) (optional) --enable-test - Enable building unit tests in test/ and third-party/ --with-netcdf-config - location of NetCDF config script nc-config (optional) --with-gdal-config - location of GDAL config script gdal-config (optional) --with-gmt-config - location of GMT config script gmt-config (optional) --with-proj-lib - location of PROJ libs (optional) --with-proj-include - location of PROJ headers (optional) --with-fftw-lib - location of FFTW3 libs (optional) --with-fftw-include - location of FFTW3 headers (optional) --with-motif-lib - location of Motif libs (optional) --with-motif-include - location of Motif headers (optional) --with-opengl-lib - location of OpenGL libs (optional) --with-opengl-include - location of OpenGL headers (optional) --with-otps-dir - location of OTPS installation (optional) --with-opencv-lib - location of OpenCV libs (optional) --with-opencv-include - location of OpenCV headers (optional) --enable-opencv - enable building tools using OpenCV (optional) --enable-mbtrn - enable building terrain relative navigation (TRN) tools (optional) --enable-mbtnav - enable building terrain relative navigation (TRN) tools (optional) --disable-mbtools - disable building graphical tools (most often used with --enable-mbtrn and --enable-mbtnav) --enable-qt - Enable building graphical tools using the Qt5 framework --with-vtk-include - location of VTK8.2+ headers (required if qt enabled) --with-vtk-lib - location of VTK8.2+ libraries (required if qt enabled) --with-debug - Set compiler flags to allow full debugging

MacOs Example Using MacPorts to Install Prerequisite packages:

The use of MacPorts to install the MB-System prerequisite packages is recommended on Apple Mac computers, particularly because this approach enables cleanly building with a complete X11 + Motif + OpenGL infrastructure separate from the libraries and header files associated with the XQuartz package. One usually still installs XQuartz and uses it as the X11 display server, but confining the MB-System compiliation and linking to headers and libraries within the MacPorts structure avoids several issues.

This example is relevant for MacOs 10.13 High Sierra to the current MacOs 13 Ventura on both Intel and ARM (Apple Silicon) architecture computers.

Install Xcode and the Xcode command line tools, which includes the LLVM compiler suite.

Install the XQuartz X11 server from https://www.xquartz.org XQuartz 2.8.5 or later is required for all MB-System installations.

Install MacPorts using the appropriate downloadable installer package from: https://www.macports.org/install.php

After MacPorts installation, first make sure the default port packages are current by running selfupdate and then install the MB-System prerequisites. sudo port -v selfupdate sudo port install gmt6 fftw-3 mesa libGLU openmotif

Also make sure that a current version of Python3 is available. First list the available Python3 versions, install the most recent, and then set port to link that version to python3: port select --list python

Result as of July 18, 2023: Available versions for python: none python27 python30 python311 (active) python38

The most recent version is python311, so install it: sudo port install python311 sudo port select --set python python311 sudo port select --set python3 python311 Also install the most recent Python imaging library Pillow sudo port install py311-Pillow

Download the MB-System source package from the repository at GitHub: https://github.com/dwcaress/MB-System There are often beta releases that are more recent than the current stable release. For instance, to download 5.7.9.beta42 go to: https://github.com/dwcaress/MB-System/archive/refs/tags/5.7.9beta58.tar.gz

Unpack the MB-System distribution tarball, and then cd into the top directory of the resulting structure. This will typically be named something like "MB-System-5.7.9". At that location, execute the configure script, named "configure", with the options necessary for your context. The XCode compiler tools do not look for header files or libraries in the locations used by MacPorts, and so it is necessary to specify these locations for several of the prerequisite packages.

This command should successfully enable building the current MB-System (5.7.9 or later) on any Mac computer with the prerequisites installed through MacPorts. This has been tested with computers running Ventura and Monterey. ./configure \ --prefix=/usr/local \ --disable-static \ --enable-shared \ --enable-hardening \ --enable-test \ --with-proj-lib=/opt/local/lib/proj9/lib \ --with-proj-include=/opt/local/lib/proj9/include \ --with-gmt-config=/opt/local/lib/gmt6/bin \ --with-fftw-lib=/opt/local/lib \ --with-fftw-include=/opt/local/include \ --with-x11-lib=/opt/local/lib \ --with-x11-include=/opt/local/include \ --with-motif-lib=/opt/local/lib \ --with-motif-include=/opt/local/include \ --with-opengl-include=/opt/local/include \ --with-opengl-lib=/opt/local/lib \ --with-otps-dir=/usr/local/src/otps

Once the makefiles have been generated by configure, build and install MB-System using: make make check sudo make install

The MB-System codebase includes some optional components, such as OpenCV based photomosaicing (enabled with --enable-opencv) and a realtime Terrain Relative Navigation infrastructure and toolset (--enable-mtrn and --enable-mbtnav). An additional prerequisite for the photomosaicing is OpenCV4, which can be installed by: sudo port install gmt6 fftw-3 libGLU openmotif opencv4 This configure command should enable building the entire MB-System package, including these optional tools. ./configure \ --prefix=/usr/local \ --disable-static \ --enable-shared \ --enable-hardening \ --enable-test \ --with-proj-lib=/opt/local/lib/proj9/lib \ --with-proj-include=/opt/local/lib/proj9/include \ --with-gmt-config=/opt/local/lib/gmt6/bin \ --with-fftw-lib=/opt/local/lib \ --with-fftw-include=/opt/local/include \ --with-x11-lib=/opt/local/lib \ --with-x11-include=/opt/local/include \ --with-motif-lib=/opt/local/lib \ --with-motif-include=/opt/local/include \ --with-opengl-include=/opt/local/include \ --with-opengl-lib=/opt/local/lib \ --enable-mbtrn \ --enable-mbtnav \ --enable-opencv \ --with-opencv-include=/opt/local/include/opencv4 \ --with-opencv-lib=/opt/local/lib/opencv4 \ --with-otps-dir=/usr/local/src/otps

---

Currently Impossible to Build MB-System on MacOs using Fink or Homebrew:

In the past it has been possible to build MB-System on a Mac using the package managers Fink or Homebrew to install the prerequisite software. Unfortunately, currently (July 2023) we do not know how to successfully install MB-System when the prerequisite packages are installed using Fink or Homebrew. Even if libraries and programs compile and link, the X11/Motif/OpenGL programs fail to run correctly.

Ubuntu Linux configure script command line

example:

The following procedure serves to install MB-System on Ubuntu Jammy 22.04.

GMT:

GMT is a key prerequisite for MB-System, which now requires GMT 6.1 or later. One can check which GMT version is available by querying the package manager in a terminal: apt-cache show gmt For Ubuntu 20.04 the available GMT version is 6.0.0. Therefore, it is necessary to install GMT 6.1+ from a personal package archive (PPA) or by compiling, linking, and installing the GMT source distribution manually.

One can enable installation from the UbuntuGIS PPA by entering the following commands: sudo add-apt-repository ppa:ubuntugis/ppa sudo apt-get update sudo apt-get upgrade

Once GMT-6.1.0 or later is available to you, install the primary package and the documentation, library files, development headers, and hierarchical high resolution geography: sudo apt-get install gmt libgmt5 libgmt-dev gmt-gshhg gmt-doc

If you need (or want) to build GMT from source, follow the installation instructions at: https://www.generic-mapping-tools.org

Motif:

MB-System depends on X11, OpenMotif, and OpenGL. These can be installed using: sudo apt-get install libx11-dev xorg-dev libmotif-dev libmotif4 \ libxp-dev mesa-common-dev libsdl1.2-dev libsdl-image1.2-dev \ xfonts-75dpi xfonts-100dpi

Other dependencies:

The many other MB-System dependencies can be installed via: sudo apt-get install build-essential gfortran nautilus-open-terminal \ libfftw3-3 libfftw3-dev libnetcdf-dev netcdf-bin \ libgdal-bin gdal-dev gv csh libgmt-dev libproj-dev

Everything at once:

In fact, it should be possible to install all MB-System dependencies in a single apt-get command: sudo apt-get install gmt libgmt5 libgmt-dev gmt-gshhg gmt-doc \ libx11-dev xorg-dev libmotif-dev libmotif4 \ libxp-dev mesa-common-dev libsdl1.2-dev libsdl-image1.2-dev \ build-essential gfortran nautilus-open-terminal \ libfftw3-3 libfftw3-dev libnetcdf-dev netcdf-bin \ libgdal-bin gdal-dev gv csh libgmt-dev libproj-dev

MB-System:

These steps assume you have downloaded an MB-System distribution tar.gz file from the GitHub repository: https://github.com/dwcaress/MB-System/releases

First, unpack the distribution using tar. This can be done in any location, but if one is working in a root-owned area such as /usr/local/src, obtaining root privileges using sudo is necessary for all steps: sudo tar xvzf MB-System-5.7.9.tar.gz and then cd into the resulting directory: cd MB-System-5.7.9.tar.gz

If the prerequisites have all been installed as shown above, and it is desired to install MB-System in /usr/local, then only a simple call to configure is required: sudo ./configure

Once the makefiles have been generated by configure, build and install using: sudo make sudo make install

In some cases the system and/or user environment impedes the successful use of the GMT and/or MB-System shared libraries. In order to manually allow shared libraries to be found for linking or running, one can either set the CFLAGS environment variable during building or set the LD*LIBRARY*PATH environment variable at login by adding a command to the user's ~/.profile or ~/.bashrc files.

To set the CFLAGS environment variable during building include "-Wl,-rpath -Wl,LIBDIR" in the configure command as shown here:

```
sudo CFLAGS="-Wl,-rpath -Wl,/usr/local/lib" ./configure
```

To augment the LD*LIBRARY*PATH environment variable during login add a line to the ~/.bashrc or ~/.profile file as shown here:

```
export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
```

Completing the build and dealing with user environment issues

On all systems there are user environment setting that are required to use some aspects of MB-System. First, three MB-System tools (mbswath, mbcontour, and mbgrdtiff) are actually GMT modules built into a shared library named "mbsystem.so". GMT only knows about these external modules through a user's configuration, which is defined by a file called "gmt.conf" that is in the user's home directory. To create this file, go to the home directory and direct the output of the gmtdefaults module to a file named "gmt.conf": cd ~ gmt gmtdefaults > gmt.conf

Next, use a text editor of your choice to edit the line of gmt.conf that sets the "GMT*CUSTOM*LIBS" parameter so that this parameter is the full path to the "mbsystem.so" shared library. If you have installed MB-System in /usr/local, then the path should be "/usr/local/lib/mbsystem.so".

The MB-System plotting macros mbm*grdplot, mbm*plot, etc all generate shellscripts that in turn execute a combination of GMT, MB-System, and other programs to generate and then display postscript plots or images. The MB-System program mbdefaults sets the programs you want to use to display postscript files and images on the screen. If, for instance, you want to use "gv" to display postscript and "feh" to display images, then run: mbdefaults -Dgv -Ifeh -V These defaults are stored in a hidden file called ".mbio_defaults" in the user's home directory.

Setting X11 fonts used by mbgrdviz, mbeditviz, mbedit, mbnavedit, mbnavadjust

and mbvelocitytool:

By default the graphical utilities use three fonts: Helvetica, Times New Roman, and Courier. This can be set in the CFLAGS environment variable by including options of the form: -DSANS='"helvetica"' -DSERIF='"times"' -DMONO='"courier"' In the examples below, the CFLAGS environment value is set for the configure script by setting it on the same command line as ./configure. To set the fonts to Lucida, one might add: -DSANS='"lucida"' -DSERIF='"lucida"' -DMONO='"lucidatypewriter"' to the CFLAGS definition

---

To modify the build system...

Edit the file "configure.ac" in the top directory and "Makefile.am" in each directory and then run the following sequence of commands:

Build libtool files for AM*PROG*LIBTOOL libtoolize --force --copy aclocal

Build custom header for configure autoheader automake --add-missing --include-deps autoconf

To update configure files use the following: autoupdate autoreconf --force --install --warnings=all

Reset the autotools version to 2.65 to accomodate some Linux distributions sed -i.bak s/2.69/2.65/ configure.ac

When you run ./configure, a number of configure options are saved to a header file: ./src/mbio/mb*config.h This file has a template: ./src/mbio/mb*config.h.in This file is conditionally included by: ./src/mbio/mb_define.h which is in turn included by essentially every MB-System C source file.

Full autoconf and build sequence after modifying the build system - Do this in the development tree prior to a commit to the source archive or prior to making a source distribution - The example here is for a MacOs environment in which the prerequisite packages have been installed with MacPorts.

First clean up old installation and build make -j uninstall make -j clean

Reconstruct the build system, including the Makefile.in files and the configure script glibtoolize --force --copy aclocal autoheader automake --add-missing --include-deps autoconf autoupdate autoreconf --force --install

Run the configure script - here the prerequisites have been installed with MacPorts, and the experimental OpenCV based photomosaicing and Terrain relative Navigation are all enabled CFLAGS="-g -Wall -Wextra" CPPFLAGS="-g" ./configure \ --prefix=/usr/local \ --disable-static \ --enable-shared \ --enable-hardening \ --enable-test \ --with-proj-lib=/opt/local/lib/proj9/lib \ --with-proj-include=/opt/local/lib/proj9/include \ --with-gmt-config=/opt/local/lib/gmt6/bin \ --with-fftw-lib=/opt/local/lib \ --with-fftw-include=/opt/local/include \ --with-x11-lib=/opt/local/lib \ --with-x11-include=/opt/local/include \ --with-motif-lib=/opt/local/lib \ --with-motif-include=/opt/local/include \ --with-opengl-include=/opt/local/include \ --with-opengl-lib=/opt/local/lib \ --enable-mbtrn \ --enable-mbtnav \ --enable-opencv \ --with-opencv-include=/opt/local/include/opencv4 \ --with-opencv-lib=/opt/local/lib/opencv4 \ --with-otps-dir=/usr/local/src/otps

make make check make install

cd src/htmlsrc ; ./make_mbhtml ; cd ../..

make -j install
