

Installation Instructions for MEGAlib

Written by Andreas Zoglauer (zog@ssl.berkeley.edu)

Copyright by Andreas Zoglauer

Version of May 8, 2014

1. What is MEGAlib?

MEGAlib stands for “Medium-Energy Gamma-ray Astronomy library”. Originally MEGAlib was the simulation, data acquisition, and data analysis software of the MEGA prototype, a combined Compton and pair telescope working in the energy range from ~ 0.2 up to ~ 50 MeV.

However since those days it has been successfully applied to a variety of hard X-ray and gamma-ray telescopes, including ACT, NCT, NuSTAR, GRI, GRIPS, COMPTEL, and others.

MEGAlib encompasses the complete data analysis path ranging from simulations with Geant4 or MGGPOD, event reconstruction, and image reconstruction. For more details see one of the overview papers.

2. System and software requirements

Make sure you have an up-to-date Linux or MacOS X operating system with a current gcc (4.2 or higher recommended) or Intel’s icc (11 or higher recommended). In theory MEGAlib should run on any Linux operating system on which you can compile (not just download and install) ROOT and Geant4, and where the GNU core utilities are installed. However, Ubuntu 12.04 is the current development system, and thus the most tested operating system. MEGAlib has also been successfully run on MacOS X from Snow Leopard to Mountain Lion, but since the main developer only uses Linux, using Mac’s is somewhat experimental.

Windows is not supported, although you can find a Visual Studio 2008 solution file in the main MEGAlib directory which is able to compile the main MEGAlib programs (Geomega, Revan & Mimrec, but not Cosima) – for more details see section 8.

To download, configure, compile, and run MEGAlib, you need a wide range of GNU/Linux utilities, such as git, bash, make, cmake, gcc, g++, cvs, gdb, bash, awk, grep, expect, etc. Just make sure you have a full installation of the Linux development tools or Apple’s Xcode including the Xcode command-line tools plus cmake which you might have to download and install by yourself on a Mac).

In addition you need to install the following software packages (but don’t start yet, because you first have to download MEGAlib to determine which version you need):

- ROOT (absolutely mandatory)
- Geant4 (mandatory if you want to do simulations or need response files, i.e. mandatory in 98% of all cases)
- Doxygen (optional & part of all standard Linux distributions)
- HEAsoft (optional)

But before you install ROOT and Geant4 yourself try the setup script of MEGAlib, which on most systems should be able to install ROOT, Geant4, and MEGAlib.

3. The MEGAlib setup script

On the setup page of the MEGAlib website, you can find a link to a small script called setup.sh (you can also find it in the root (not ROOT) directory of your MEGAlib installation):

<http://www.megalibtoolkit.com/setup>

Assuming you have followed the instructions in section 2 (especially did you make sure to install all the libraries ROOT needs to compile, and have git and cmake installed? On Mac OS X make sure you have xcode, and the xcode command-line tools installed (e.g. via “xcode-select –install” in the command line) as well as XQuartz), then running the install script should be a breeze. In the directory where you want to install MEGAlib execute:

```
bash setup.sh
```

Now the script will do the following:

1. Check if we already have a MEGAlib installation - by default it will search in the directory MEGAlib, or check if it is called from within a MEGAlib installation. If this is the case it will load the command line parameters from the previous run of setup. These are overwritten by the new options you give at the command line.
2. MEGAlib download: It will check if MEGAlib is present. If not it will download it, if it is present it will update it (but it will make a backup of your src-directory, just in case). If you decided to switch from a release to a development version, it will do that.
3. ROOT installation: If you have given a path to an existing ROOT installation (--root option), or if it has found an old installation done via this script, it will check if this version is still up to date. If it is not up to date and you have given this ROOT version via the --root option, it will stop here. Otherwise it will download and install the latest version of ROOT compatible with this MEGAlib version.
4. Geant4 installation: This is exactly the same process as for ROOT.
5. MEGAlib: It will configure and compile MEGAlib.
6. Finally it will create a little script which you have to source in order to setup the environment for MEGAlib

Assuming you don't get an error it will tell you will point something like the following (obviously the path will point to your installation):

In order to run any MEGAlib programs, you have to source the following file first:

```
source /home/andreas/Soft/MEGAlib/bin/source-megalib.sh
```

You can do this in your .bashrc file, or everytime you want to use MEGAlib.

This script sets the environment variables for ROOT, Geant4, and MEGAlib. Ideally you should put it into your .bashrc file.

If you do `bash setup.sh --help`, then you will see a lot of options for the setup script. You can set the name and path of your MEGAlib installation (--megalib-path), you can set the directory where you want to install ROOT and Geant4 (--external-path), or you can give the path to an existing ROOT (--root) or Geant4 (--geant4) installation, which you want to use instead of downloading it, you can decide if you want to have the latest release or development version (--release), and you can decide which repository you want to use (--repository, but please stick with git).

The setup script also doubles as update script. Just call it from within your MEGAlib installation and it will update MEGAlib, and if it installed ROOT and Geant4 previously it will also update them.

4. Getting MEGAlib

If you reached this sections of the installation manual, the MEGAlib setup script has failed, and you have to install MEGAlib on your own...

There are multiple ways to get MEGAlib. The recommended one is the git repository, if you are a local programmer you might have access to the svn repository, or if you have an old installation you might be used to the cvs repository.

The recommended way is to get MEGAlib from the git repository and to install it into one of your own directories instead of system wide, since MEGAlib is under permanent development and therefore this is the easiest way to get the latest updates. There is no precompiled version of MEGAlib. Therefore you will always have to compile it yourself.

4.1. The git repository

MEGAlib's public repository is on github.com. Thus do the following to checkout MEGAlib (attention: on some systems a copy-and-paste of the cvs commands might give errors):

MEGAlib folder

Change to the directory where MEGAlib should be installed

Clone the git repository:

```
git clone https://github.com/zoglauer/megalib.git MEGAlib
```

Now cd to the MEGAlib directory:

```
cd MEGAlib
```

Release or development version:

Now you can choose if you want to choose the development version, which is on the branch master, or one of the release versions, which is in a branch named like MEGAlib_vX.YZ, e.g. MEGAlib_v2.26.

To get the development version in the MEGAlib directory do:

```
git checkout master
```

Normally it should tell you that you are already on master.

The get the latest release version you first need to get a list of branches to determine the latest MEGAlib version (this is one line):

```
git ls-remote --heads git://github.com/zoglauer/megalib.git | grep  
MEGAlib_v
```

This will list all release branches. Just choose the highest version number. Assuming MEGAlib_v2.26 is the highest number, do:

```
git checkout MEGAlib_v2.26
```

You should now have the latest MEGAlib on your computer.

4.2. The cvs repository - depreciated

The old method to get MEGAlib is from the cvs repository. As of version 2.6 the repository is no longer maintained – as long as it is running only major versions will be

To check out MEGAlib via anonymous access to the cvs repository do the following (attention: on some systems a copy-and-paste of the cvs commands might give errors):

Folder

Change to the directory where MEGAlib should be installed

Login to the repository:

```
cvs -d :pserver:anonymous@cvs.mpe.mpg.de:/home/zoglauer/Repository login
```

The password for the user “anonymous” is “gamma-ray”.

You only need to do this once. Cvs will store the password on your hard drive and reuse it later.

Please make sure to setup your cvs system, so that it stores this password (which is the default). Otherwise you will get some password requests when the MEGALib update mechanism tries to download the version file from the cvs server and fails.

Check out the source:

Development version

To check out the latest version, do (this is one line!):

```
cvs -d :pserver:anonymous@cvs.mpe.mpg.de:/home/zoglauer/Repository -z3  
co -P MEGALib
```

Do not forget the “-P”! This command checks out the latest development version of MEGALib, but since the cvs repository is not the main development repository, but just the public repository, it is unlikely that you will find a badly unstable version there.

You can ignore errors such as:

```
cvs checkout: cannot open directory  
/home/zoglauer/Repository/MEGALib/src/herty: Permission denied
```

Release version

To check out the latest stable branch (e.g. if the latest version is X.YZ then use “MEGALib_vX-YZ”, e.g. MEGALib_v2-24) you have to issue the following commands (one line):

```
cvs -d :pserver:anonymous@cvs.mpe.mpg.de:/home/zoglauer/Repository -z3  
co -r MEGALib_vX-YZ -P MEGALib
```

Don't forget to replace the X-YZ! However, if you check out a branch, you have to pay attention, because when you update, you only update to the latest version in the branch, not the latest version overall!

Again, you can ignore errors such as:

```
cvs checkout: cannot open directory  
/home/zoglauer/Repository/MEGALib/src/herty: Permission denied
```

5. Installing additional software

5.1. ROOT

The first step is to determine which version of ROOT you need. Change into the MEGALib directory and execute the following command:

```
bash config/check_rootversion --get-max  
$ 5.34
```

Ok, this tells you, you need ROOT version 5.34. So navigate to the ROOT homepage (see <http://root.cern.ch> → <http://root.cern.ch/drupal/content/downloading-root>) and download version 5.34 with the latest patch. You have to download the source tar-ball, not any of the binaries! Also take a look at the following ROOT page, and ensure you have all the required packages installed: <http://root.cern.ch/drupal/content/build-prerequisites>

Then copy the downloaded ROOT tar ball to the directory where you want to install ROOT. MEGALib comes with a ROOT build script: config/build-root.sh. Simply copy this script to the directory where the ROOT source code is and execute it with the ROOT tar ball as parameter:

```
bash build-root.sh --tarball=root_v5.34.07.source.tar.gz
```

In most cases everything should go well, but if not try to install ROOT following the instructions on the ROOT webpage.

The final step is to set all the environment variables. If you used the build-root.sh script, you can use the config/env.sh script. Since env.sh will set all environment variables required for MEGALib, we will do this later. If you installed ROOT without the script, then continue following the ROOT webpage when setting the environment variables.

5.2. GEANT4

The first step is to determine which version of Geant4 you need. Changed into the MEGALib directory and execute the following command:

```
bash config/check_geant4version --get-max
$ 9.5
```

Ok, you need Geant4 version 9.5 (with the latest patch). So navigate to the Geant4 homepage (see <http://geant4.cern.ch>) and download version 9.5 with the latest patch. You have to download the source tarball, not any of the binaries! On a Mac, please make also sure you have installed cmake which is required to compile Geant4 9.5 and higher.

Then copy the downloaded Geant4 tar ball to the directory where you want to install Geant4. MEGALib comes with a Geant4 build script: config/build-geant4.sh. This will only work for version 9.5 or higher! Simply copy this script to the directory where the Geant4 source code is and execute it with the Geant4 tar ball as parameter:

```
bash build-geant4.sh --tarball=geant4.9.5.p02.tar.gz
```

In most cases everything should go well, but if not try to install Geant4 following the instructions on the Geant4 webpage.

The final step is to set all the environment variables. If you used the build-geant4.sh script, you can use the config/env.sh script. Since env.sh will set all environment variables required for MEGALib, we will do this later. If you installed Geant4 without the script, then continue following the Geant4 webpage when setting the environment variables.

5.3. Doxygen & dot (graphviz) - for doxygen source code documentation

There is the possibility to generate source code documentation in a browsable form. If you want to this you need to have Doxygen and dot (also known as graphviz) installed.

5.4. HEASoft - ONLY required for MGGPOD simulations

If you want to use MGGPOD, and convert the ACT/INIT fits files into the MEGALib file format via the program ConvertMGGPOD, you also need to install HEASoft. Make sure that the environment variable HEADAS is set correctly, or MEGALib will not find HEASoft.

Attention:

Please make sure to initialize HEASoft before ROOT in your .bashrc, .bash_login, etc., file., since it has its own “libreadline.so” file which might interfere with “libreadline.so” of your operating system and cause havoc!

6. Configure and compile MEGALib

Tip: The last section of this document describes some common problems and their solutions.

Setting the environment variables

Since setting all the environment variables for ROOT, Geant4, clhep, HEASoft, and MEGALib can be error prone, MEGALib comes with a small bash script to do it for you, called `config/env.sh`. You call it before using MEGALib (and even before installing it), or even better, you put this line into your `.bashrc` file (this is one line):

```
source env.sh --prefix=<optional path prefix common to all other paths>
--root=<path to root> --geant4=<path to geant4> --clhep=<path to clhep>
--heasoft=<path to heasoft> --megalib=<path to megalib>
```

So for example (one line):

```
source env.sh -p=/home/andreas/Software -r=root_v5.34.04 -g= --
geant4_v9.5 -h=heasoft_v6.13 -m=MEGALib_v2.26
```

You of course can also skip the `--prefix` or shorter `-p` and give the full paths.

See `./env.sh --help` for more information.

If you want to setup everything by yourself instead of using the `env.sh` script, first make double sure you have correctly set all paths to ROOT and GEANT4 and if you need fits support, HEADAS.

Then set the following shell variables to your `.bashrc` file or whatever shell you use:

```
export MEGALIB=<path to MEGALib>
export PATH=$PATH:$MEGALIB/bin
```

For Linux:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$MEGALIB/lib
```

For Mac OS X:

```
export DYLD_LIBRARY_PATH=$DYLD_LIBRARY_PATH:$MEGALIB/lib
```

If you use cvs (one line!):

```
export
CVSROOT=:pserver:yourusername@cvs.mpe.mpg.de:/home/zoglauer/Repository
```

Configuring

Then go to your MEGALib directory and just type:

```
./configure
```

`configure` is a bash script, thus if you do not use or don't have bash installed you will get strange errors!

`./configure --help` will give you all the options. I recommend compiling the software in normal debug mode until you are sure that it works without problems. There is even an option "`--debug=strong`" to show even more debug output such as depreciated functions or implementation limitations. When you are sure your code works, you can remove "`--debug=on`" and just leave the "`--optimize=on`". You can replace the latter with "`--optimize=strong`" for even better performance (requires gcc 4.2 or higher). Type "`./configure --help`" for a list of all options.

If the configuration script complains about either a ROOT or Geant4 version which is more current than the ones the configuration script wants, you can try to increase the `RootVersionMax` or `Geant4VersionMax` values in the scripts `config/configure_rootversiontest` or `config/configure_geant4versiontest`. However this procedure is experimental and no

guarantee can be given that it works. This procedure is for adventurous users only – all other users should install one of the recommended versions.

Compiling

If no error message was produced, then compile MEGALib with:

```
make
```

If you have several processors/cores, you can use the “-j” option. For example if you have 4 cores do:

```
make -j4
```

But be aware that on some older system this sometimes crashes make!

You can also compile individual modules like the geometry program geomega by invoking make with “make geomega”. Alternative options are “make cosima”, “make revan”, “make mimrec”, etc.

Recompiling

To recompile, issue the following commands:

```
make clean
```

```
make
```

A well known root/xft bug

If you start one of the GUI based programs the first time, and it immediately crashes, or no GUI window appears, or you have “funny” fonts, you most likely encountered a bug in libXft which shows up in combination with ROOT. The solution is to configure ROOT with “--disable-xft”, recompile ROOT and MEGALib. As an alternative, in the file \$MEGALIB/src/global/misc/src/MGlobal.cxx you can also comment out the line:

```
gEnv->SetValue("X11.UseXft", "true");
```

7. Updating

If you have use the flag --update=on during configuration, then MEGALib will regularly (once a week) check if newer versions are available in the cvs repository. It will use the script config/configure_updatetest for this task. This will only be successful if you have installed MEGALib via the cvs repository and if you have write access to the MEGALib directory.

If an updated version is found, make a backup and then issue the following command in your MEGALib directory:

```
make update
```

This will update to the latest version – if you are in the main development trunk it will update to the latest development version there, if you are in a release branch it will update to the latest version in this branch or, if a newer release branch is available, it will switch to the latest release branch. If there is a previous successful configuration of MEGALib, it will then reconfigure MEGALib using the previous options and start a recompilation.

You can also update to the latest version by hand via switching to the MEGALib directory and issuing the command

```
cvs update -d
```

If you forget the final “-d” then new directories are not updated!

You then have to reconfigure and recompile.

If you want to switch to a certain branch, do

```
cvs update -r MEGALib_v2-26 -d
```

If you want to switch to the main trunk, do

```
cvs update -A -d
```

8. Further documentation

Further documentation for the individual packages – if there is any – can be found in the “doc” directory. In addition you can generate Doxygen documentation of the source code by issuing the command:

```
make doxygen
```

You can then browse the source code through the following html file: doc/html/index.html

9. Tips for Windows users

Windows is not supported.

Nevertheless there exists a Visual Studio 2008 solutions file Megalib.sln in the main MEGALib directory, which can be used to compile the main MEGALib programs Geomega, Revan, Mimrec, and Sivan using Visual Studio 2008 (either the full or the Express Edition). The simulation program Cosima can currently not be compiled under Windows.

The following procedure has been tested on Windows 7 using Visual Studio 2008:

- Install ROOT: Use the ROOT version which is “recommended” on the ROOT website with the following characteristics: VC++ 9.0, MSI, Release (NOT Debug).
- Check out MEGALib from cvs.
- Set an environment variable “MEGALIB” pointing to the MEGALib directory (e.g. MEGALIB → %HOMEPATH%/Documents/Software/MEGALib). Using Windows 7 you do this in: Control panel → System and security → System → Advanced System Settings → Advanced → Environment Variables → User variables
- Set the “ROOTSYS” environment variable to point to the ROOT directory (e.g. ROOTSYS → C:/root/)
- Open Megalib.sln with Visual Studio.
- Make sure you setup Visual Studio to compile only one project at a time or Visual Studio screws up. Using Visual Studio 2008 you find this option in: Tools → Options → Projects And Solutions → Build and Run → Maximum number of parallel project builds: 1. In release mode individual source files within one project are still compiled in parallel, just the projects are not build in parallel.
- Build the solution – make sure you don’t mix debug and release builds, i.e. if you have a Release version of ROOT you can compile MEGALib also only in Release mode
- You can find the binaries in %MEGALIB%/bin (they will not be installed anywhere!)

Attention: The solution and project files are not updated on a regular basis. As consequence it is possible – if you install the latest and greatest version from cvs – that some newly added classes have not yet been included. However, for the average MEGALib/Visual Studio user it should be straight forward to do this by yourself.

10. Bug reports

Bug reports go to Andreas Zoglauer: zog@ssl.berkeley.edu - please be specific, provide me with information about what version you are using, let me know how to reproduce the problem, and send me an example! It is also recommended that you run `./config/configure_checkenvironment` and email me the output along with your bug report. The script dumps all environment variables which MEGALib requires, and will reveal most misconfigurations.

11. Solution to some error messages

Example configuration

For Linux with bash the configuration files should look similar to this. Do NOT simply copy and paste this to your .bashrc-file! You have to adapt it to your own system! This is just an example, to check if you forgot something!

Attention: The sequence does matter, since e.g. HEASoft and ROOT have libraries which are named the same way. In addition, HEASoft provides its own version of libreadline.so, which might interfere with system maintenance as super-user root.

```
PRG=/prg

# HEASOFT
if [ "$USER" != "root" ]; then
    export HEADAS=${PRG}/headas/i686-pc-linux-gnu-libc2.5
    alias heainit=". $HEADAS/headas-init.sh"
    source $HEADAS/headas-init.sh
fi

# ROOT
export ROOTSYS=${PRG}/root
export PATH=$PATH:$ROOTSYS/bin
export LD_LIBRARY_PATH=$ROOTSYS/lib:$LD_LIBRARY_PATH

# Geant3/MGGPOD
export CERN=${PRG}/geant3
export CERN_LEVEL=
export CERN_ROOT=${PRG}/geant3
export GLECS=${PRG}/glecs
export GLEPS=${PRG}/gleps
export GLECS_DATA=${PRG}/glecs/data
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CERN/lib
export PATH=$CERN_ROOT/bin:$PATH

# Geant4
export G4SYSTEM=Linux-g++
export G4INSTALL=${PRG}/geant4
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$G4INSTALL/lib/$G4SYSTEM

export G4LEDDATA=${G4INSTALL}/data/G4EMLOW
export G4RADIOACTIVEDATA=${G4INSTALL}/data/RadioactiveDecay
export G4LEVELGAMMADATA=${G4INSTALL}/data/PhotonEvaporation
export G4NEUTRONHPDATA=${G4INSTALL}/data/G4NDL
export G4ABLADATA=${G4INSTALL}/data/G4ABLA

# CLHEP
export CLHEP_BASE_DIR=${PRG}/CLHEP
export CLHEP_INCLUDE_DIR=${PRG}/CLHEP/include
```

```
# MEGAlib
export MEGALIB=${SOFTWARE}/MEGAlib
export PATH=${MEGALIB}/bin:$PATH
export LD_LIBRARY_PATH=${MEGALIB}/lib:${LD_LIBRARY_PATH}
```

ROOT compilation problems

Error message similar to:

Compiling XrdNetDNS.cc

```
g++ -c -D_LARGEFILE_SOURCE -D_LARGEFILE64_SOURCE -D_FILE_OFFSET_BITS=64 -D_REENTRANT -D_GNU_SOURCE -Wall -D__macos__ -Wno-deprecated -undefined dynamic_lookup -multiply_defined suppress -O2 -DXrdDEBUG=0 -I. -I. XrdNetDNS.cc -o ../../obj/XrdNetDNS.o
```

```
XrdNetDNS.cc: In static member function 'static int XrdNetDNS::getHostAddr(const char*, sockaddr*, int, char**)':
```

```
XrdNetDNS.cc:73: error: 'gethostbyname_r' was not declared in this scope
```

```
XrdNetDNS.cc:82: error: 'gethostbyaddr_r' was not declared in this scope
```

```
XrdNetDNS.cc: In static member function 'static int XrdNetDNS::getPort(const char*, const char*, char**)':
```

```
XrdNetDNS.cc:393: error: 'getservbyname_r' was not declared in this scope
```

```
make[5]: *** [../../obj/XrdNetDNS.o] Error 1
```

```
make[4]: *** [Darwinall] Error 2
```

```
make[3]: *** [all] Error 2
```

```
make[2]: *** [XrdNet] Error 2
```

```
make[1]: *** [all] Error 2
```

```
make: *** [net/xrootd/src/xrootd/lib/libXrdSec.so] Error 2
```

The Xrd component of ROOT has sophisticated dependencies, but it is not required for MEGAlib. Thus simply disable it during ROOT configuration with:

```
./configure -disable-xrootd
```

MEGAlib configuration/compilation problems

Error message similar to:

(2) ROOT

```
Found ROOT: /home/andreas/prg/root/bin/root
```

```
Found ROOT version: 5.20/00 (minimum: 5.18, maximum: 5.20)
```

```
[: 108: ==: unexpected operator
```

You didn't use "bash" to run configure

Error message:

Generating dictionary... This may take a while...

rootcint: error while loading shared libraries: libCint.so: cannot open

shared object file: No such file or directory

Something is wrong with your ROOT installation:

- Did you install ROOT correctly?
- Does your LD_LIBRARY_PATH contain the correct settings for ROOT?

Error message similar to:

```
./lib/Linux-g++/cosima/libcosima.a(MCMain.o): In function `MCMain::Initialize(int, char**)':  
/MEGAlib_trunk/src/cosima/src/MCMain.cc:122: undefined reference to  
`G4UITerminal::G4UITerminal(G4VUIshell*, bool)'
```

You did not compile the Geant4 UI modules. Enable them during the configuration of Geant4.

Error message similar to:

```
bin/Linux-g++/Cosima: error while loading shared libraries: libCLHEP-1.9.2.3.so: cannot  
open shared object file: No such file or directory
```

The CLHEP library, a library which Geant4 requires, is not found on your system.

Possibilities are:

- You have not compiled and installed CLHEP
- The library is not in the correct place, e.g. in /usr/local/lib or whatever you gave as path during Geant4 installation
- On Linux you might have to do a "ldconfig" (as root) after you installed CLHEP
- You have installed a different version of CLHEP. Here reconfiguring and recompiling MEGAlib might help.
- If you have installed CLHEP in some strange place, you might have to add
export LD_LIBRARY_PATH=\$LD_LIBRARY_PATH:<path to libCLHEP.so>
to your .bashrc file (use "DYLD" on Mac OS X).

Error messages similar to:

```
usr/local/include/CLHEP/Random/Randomize.h:64: undefined reference to  
`HepRandom::createInstance()'
```

CLHEP is not correctly installed. Two hints:

1. Does .config/bin/Linux-g++/config.sh contain the CLHEP/lib directory?
2. Do you have a library called libCLHEP.a (and not e.g. libCLHEP-g++.1.8.0.0.a)?

Error message:

```
libG4UIbasic.so: undefined reference to `Xm...
```

motif (Xm) is not correctly installed on your system. Either install it or don't compile it into Geant4.

Error message similar to:

*/opt/prg/geant4/lib/Linux-g++/libG4UIbasic.so: undefined reference to
`XawDialogGetValueString'*

xaw is not correctly installed: Either install it or don't compile it into Geant4.

Error message:

xxx.so kann nicht geladen werden

xxx.so cannot be loaded

Something is wrong with your Geant4 installation:

1. Is the Geant4-lib path set?
2. Is the Geant4-lib path set correctly (e.g. does it contain lib/Linux-g++ on Linux systems)?

Error message similar to:

src/MCPhysicsList.cc:36:31: G4LECSRayleigh.hh: File not found

src/MCPhysicsList.cc:37:30: G4LECSCompton.hh: File not found

You did either not install the Mark Kippen's G4LECS package correctly or you don't have read rights to the files! From MEGALib 2.21 on you don't need G4LECS any more.

Error message similar to:

Linking Cosima ...

/usr/X11R6/lib/libGLU.so: undefined reference to `operator new(unsigned)@GLIBCPP_3.2'

/usr/X11R6/lib/libGLU.so: undefined reference to `vtable for

__cxxabiv1::__class_type_info@GLIBCPP_3.2'

Something is wrong with GLU. If GLU/OpenGL was deactivated during Geant4 compilation, commented it out by hand in the geant4/config/sys file.

MEGALib execution problems

All MEGALib GUI programs crash with an error message like (or you see no GUI at all or have "funny fonts"):

Attaching to program: /proc/29042/exe, process 29042 done.

[Thread debugging using libthread_db enabled]

[New Thread 0x7f9e513606f0 (LWP 29042)]

0x00007f9e49a1ffd5 in waitpid () from /lib/libc.so.6 error detected on stdin

The program is running. Quit anyway (and detach it)? (y or n) [answered Y; input not from terminal]

Detaching from program: /proc/29042/exe, process 29042

Some Xft implementations (Xft is used for font smoothing) seem to have problems how ROOT is using them. You have to reconfigure ROOT with the option "--disable-xft", recompile ROOT and recompile MEGALib.

As an alternative you can also comment out the line:

```
gEnv->SetValue("X11.UseXft", "true");  
in the file $MEGALIB/src/global/misc/src/MGlobal.cxx
```

Error message similar to:

Error in <TUnixSystem::DynamicPathName>: MathMore[.so | .sl | .dl | .a | .dll] does not exist in <long list of paths>

Error in <ROOT::Math::IntegratorOneDim::CreateIntegrator>: Error loading one dimensional GSL integrator

Something is wrong with your ROOT installation. Either you did not compile ROOT's MathMore library (did you say `-disable-mathmore` during configuring ROOT?), or the MathMore library couldn't be compiled because GSL ("GNU scientific library") isn't installed on your system. Either way make sure GSL is installed on your system and you have configured ROOT to compile MathMore. Fixing this requires a recompilation of ROOT.

Other

Where is ConvertMGGPOD?

It is located in `$(MEGALIB)/bin` like all other programs.

But since it needs `fitsio` it is only compiled if HEASoft is correctly installed, i.e. when the path to HEADAS exists and is ok.

I have in my path:

```
export HEADAS=/prg/headas/i686-pc-linux-gnu-libc2.5  
alias heainit=". $HEADAS/headas-init.sh"  
source $HEADAS/headas-init.sh
```

Then `ConvertMGGPOD` is automatically compiled during the standard make process.

If you have initially HEASoft not installed, you have to reconfigure MEGALib (e.g. `sh configure - linux`). Otherwise MEGALib will not find it.

It seems that (at least on Linux) in order to build `ConvertMGGPOD` you must have the shared version of the `cfitsio` library. The default static library is not sufficient. Use "make shared" than "make install" from the `cfitsio` directory to build the shared library.