Magics++
Installation Guide

Version 2.6.1
October 2008

Author: Stephan Siemen

European Centre for Medium-Range Weather Forecasts
Shinfield Park
Reading
RG2 9AX
United Kingdom

WWW: http://www.ecmwf.int/publications/manuals/magics/magplus/

Mail: magicsplus@ecmwf.int

Overview

For Magics++ the GNU *autotools* were chosen for the installation routine. These are the standard installation tools for most free and commercial software packages to date on Unix.

Autotools' main task is to generate Makefiles for the desired platform on which Magics will be used. This is necessary, because the various platforms differ from each other in various ways, such as compiler options and library/include paths.

Autotools themselves do not need to be installed on the system of the customer. The Graphics Section at ECMWF provides a Unix shell script called *configure* which is executed by the person installing Magics++. The script will then run some tests on the customer's system to find out if required third-party software libraries are available and notes their locations (paths). Based on this information the script produces the Makefiles needed to compile and install Magics.

Static versus shared library

Magics++ can be built as a static library and as a shared library. Using Magics++ as a shared library has an impact on how it is used. **With a shared library it is not only necessary that the library is accessible at the time of compilation but also at run-time.** Changes to the system, such as the removal of an old shared library might cause programs linked with this library to fail. For more information about the use of the shared library please read section "User setup".

The advantages of using shared libraries are that the size of executables will be smaller and that a different version of Magics++ can be used without recompiling the executable (if the libraries are binary compatible).

Requirements

Required third-party software

The following list of software should be installed on your system before you try to install Magics++. If you use a package manager, such as RPM, to install software make sure to include the corresponding development packages with the header files. The *configure* script will test for these libraries and give error messages if one of them is missing.

- NetCDF library with C++ interface (http://www.unidata.ucar.edu/software/netcdf/)
- Expat XML parser
- GhostScript (especially the fonts!)

Optional for raster output you should also install:

- Freetype (2.1.3 or later)
- gd library (2.0.32 or later - www.libgd.org)
- Cairo graphics library (1.4.10 or later - www.cairographics.org)

ECMWF support libraries

To read GRIB and BUFR data formats these two libraries, provided by ECMWF without charge, need to be installed on your system before Magics++ is installed:

- GribAPI (1.6.4 or higher) - www.ecmwf.int/products/data/software/grib_api.html
- EmosLib (330 or higher) - www.ecmwf.int/products/data/software/download/interpolation.html

Please be aware that Magics++ internally works only in double floating point precision and therefore requires the double precision version of EmosLib. If BUFR support is not required and disabled (which is the default!), then EmosLib is not needed. For EmosLib a Fortran 90 compiler with *Cray pointer* support is required. At ECMWF the *Portland Pgf90* compiler 5.2 and 7.1 and *GFortran* 4.1 and newer were tested successfully.

Compilation environment

At ECMWF, Opensuse 10.3 Linux systems (32 and 64bit) were used for testing Magics++. Any C++ Compiler which supports features required for the ANSI C++ standard from 1998 (STL, namespaces, templates) should work with Magics++. At ECMWF we tested GCC's *g++* 3.3 and 4.2 successfully. A Fortran compiler is not required for the compilation of Magics++, but needed if you want to use the Fortran interface or compile a dependent Fortran library (Emoslib).

Compilation and installation

To compile and install the Magics graphics library, the installer must first unpack the *.tar.gz file, provided by ECMWF, to a temporary location.

tar -xzvf Magics++-2.6.1.tar.gz

Generating the make files with *configure*

After changing into the unpacked Magics directory, the user then has to run the *configure* script. The script gives the user feedback on what requirements are fulfilled and what software is still required. Table 1 gives an overview of the different options of *configure*. More options of the script are available and can be listed by typing *configure --help* in the console. The default (without any options) will compile a shared library only and install it in /usr/local/.

Option	Explanation	Default
--help	Outputs all options of configure.	
--prefix=*/your/path/*	Directory in which Magics++ will be installed.	/usr/local
--enable-32bit	Compile 32bit version (only on 64bit Linux)	no
--enable-debug	Add debug information to library to assist debuggers	no
--enable-raster	Enables raster image output (GIF, PNG, JPEG).	yes
--enable-cairo	Enables the Cairo output driver (PNG, EPS, PDF)	no
--enable-bufr	Enables BUFR support (Emoslib is required)	no
--with-grib-api=<path>	Gives the location where GribAPI is installed	/usr/local
--with-netcdf=<path>	Gives the location where NetCDF is installed	/usr/local
--enable-static --disable-shared	Compiles static library instead of shared one.	no yes
--libdir=<path>	Rename directory name if required (such as if you need lib64 names under 64bit)	$prefix/lib

Table 1: Options of the configure scripts.

The C, C++ and Fortran compilers are chosen by *configure*. This can be overwritten by setting the variables **CC**, **CXX** and **F77** on the *configure* command line to the preferred compiler.

Be aware when using Portland Fortran compilers and g++: The PGF90 compiler cannot handle C++ exceptions from the GNU compiler in static libraries. In this case we suggest to first build only the shared library, and then to build a static version of the library using

./configure --disable-exceptions --disable-shared --enable-static F77=pgf90

plus any other *configure* option you want. By setting the variable F77 the user can hard code the Fortran compiler chosen by *configure* (in this case *pgf90*).

The most important option is *--prefix*. Setting the prefix defines where your Magics++ library and executables will be installed. This path is later important for the user setup.

The options to enable/disable output formats allow you to customise your installation. For example, if you have problems on your system with support libraries (see previous section), you might want to try to disable the raster output. The GD library is responsible for most third-party dependencies.

Compiling the code

After the *configure* script has run successfully, the user can compile the library by typing *make* in the same directory.

Testing your build

The Magics++ code contains a directory called *test* in which, in separate subdirectories, tests for the various interfaces of Magics++ are provided. Test programs for the *Fortran* and *C* interfaces are compiled if *make check* is invoked from the root directory. You might have to adjust your $MAGPLUS_HOME variable to do so. MagML example files are also provided and should be processed through the ***magmlx*** interpreter built earlier.

The compiled programs should be run, and their output verified before the library is installed. This setup does not check if the user setup (see next section) is correct, but the code in *test* can be used to do so. More examples of source code and MagML files can be found on the Magics++ web page.

Installing the library

Once the build and tests have been successfully completed, the command *make install* copies the library in the correct location on the system. Administrator permission might be required, depending on the installation directory. You might want to run *make -n install* first, which will show you what will be installed where, without performing any changes to your system.

To free space, the temporary unpacked source directory can be cleaned of the object files with *make clean* after a successful installation.

User setup

This section will discuss the setup needed by users to use the installed Magics++ library and its applications.

Magics++ environment variables

The variable **$MAGPLUS_HOME** is the path to where Magics++ was installed and should be set for all users of Magics++ programs. If you used the *configure* prefix option use the same path as there. If Magics++ reports problems at run-time about missing coastlines or fonts this variable might not be set correctly.

Magics++ provides feedback on the console. You can control the messages plotted by setting any of the following environment variables:

$MAGPLUS_INFO - Information and hints to what Magics++ is doing

$MAGPLUS_DEBUG - Debug information

$MAGPLUS_QUIET - suppresses the Magics++ header, footer and info output – only errors and warnings will cause output

System environment variables

Hint: If you choose a standard system folder, such as */usr/local/*, for your installation you might not have to change any system environment variables.

$LD_LIBRARY_PATH - To work with shared libraries the locations of these libraries need to be known at run time. This environment variable contains a list of paths where at run-time the system loader will look for libraries. To use the shared library version of Magics++ the location of the library needs to be set in $LD_LIBRARY_PATH. **If the variable is not set correctly the loader might pick up a version of Magics++ which was not intended.**

If EmosLib is compiled with Pgf90, it binds in libpgc.so dynamically at runtime and so the path to this library needs to be in $LD_LIBRARY_PATH.

$PATH - Magics++ provides not only libraries, but also some executables. To use these, the $PATH needs to include the location of these programs. Usually this will be ***$MAGPLUS_HOME/bin***.

magics-config

To simplify the user setup Magics++ installs a script called *magics-config*, which prints out the options to compile Magics++ programs. The options are based on the options used to compile Magics++. The program outputs the compilation and linkage options which can be used directly in the compilation instruction, as shown below.

To compile a C program and link it to Magics++ you only need to type

```
gcc shade.c -o shade `magics-config --clibs --cxxflags`
```

To do so **$MAGPLUS_HOME/bin** needs to be added to the **$PATH** first or *magics-config* called with its full path. Below you find the usage help for *magics-config*.

Please be aware that the option '--print-setup' will tell you which environment variables you need to set before you can compile and run Magics++ programs. You might want to add this setup in your shell or start-up scripts.

```
Usage: magics-config [OPTION] ...

Uses by default version 2.3.0 of Magics++.

Generic options
  --version     output Magics++ version information.
  --help        display this help and exit.
  --print-setup print how the environment can be set up

Compilation support options
  --cxxflags    print pre-processor and compiler flags for C/C++
  --libs        print linking flags for C++
  --clibs       print linking flags for C
  --f90static   print library linking information (static) for Fortran
  --f90shared   print library linking information (shared) for Fortran

  --double      include fortran double precision (default single)
  --64bit       include fortran option for 64 bit

When using 'double precision' you need to define -r8 (pgf90) or -fdefault-real-8
(gfortran) at the compilation of Fortran programs! '--double' has NO effect on C/C++
programs.

C/C++ flags use shared libraries if possible, otherwise static libraries are used.

Install directories Magics++ was configured to
  --prefix[=DIR]
```

This script is still quite new and we would appreciate to have feedback from users if they have any suggestions or problems!

FAQ

Why do I get the message `error while loading shared libraries: libMagPlus.so.1: cannot open shared object file: No such file or directory` when running a Magics++ executable?

You need to alter your $LD_LIBRARY_PATH variable, as described in section "User setup". The variable needs to contain the path to your Magics++ library.

Why do I get a message about a missing / not found libpgc.so when trying to run a Magics++ executable?

If your EmosLib is compiled with Portland's *Pgf90*, it requires this library as shared library at run-time. You need to alter your $LD_LIBRARY_PATH variable, as described in section "User setup".

Can I install Magics++ in the same directory as I have installed Magics 6?

Yes, it should be possible. The libraries have different names for configuration files and use different directories (*share* instead of *coast*). You might however prefer a clear separation if you start to phase out Magics 6.

How can I change the coastline files?

Coastline files are given in NetCDF format. Have a look at the directory *gshhs* in the main Magics++ directory. There you will find the program which produces the NetCDF files. You can create your own data files *gshhs* files at http://rimmer.ngdc.noaa.gov/coast/ .

Can I add GIS information, such as rivers and borders to my plots?

Yes. Some rivers have been added with the new coastline files and are visible at higher resolution of coastlines. Magics++ can also read *MapGen* files for definitions of political borders. The default MAPGEN format is lon,lat with new segments separated by a line containing the characters "# -b" in an ASCII file.

Why have the *Printercap* file and the frame in my PostScript disappeared?

With version 2.4 of Magics++ we have changed the set-up of how the PostScript driver was internally organised. The *Printercap* configuration file was created in the past to accommodate differences between printers. This is nowadays no issue any more. Out of the same reason PostScript output was scaled down to 95% to allow room for Printer alignment problems. From version 2.4 onwards the scaling is, as in all other drivers, set back to 100% of the page size.

Why did version 2.4 of Magics++ was followed by version 2.6?

With Magics++ 2.4 we introduced a new numbering scheme were the sub version indicates how stable the version of Magics++ is. Odd numbers (2.5, 2.7, …) indicate unstable development versions and even numbers (2.4, 2.6, …) indicate versions we think are stable and tested enough to be used in your applications.

How can I report bugs or ask for help?

Please write an email to *magicsplus@ecmwf.int* . Please compress any larger files you might need to attach with *gzip*.

Quick installation guide

This is a list of commands needed to install Magics++. It is assumed ">" is the shell prompt. The Magics++ version number may vary and *italic* text is example output from programs and commands run on the console.

>**tar -xzf Magics++-2.6.1.tar.gz**
>**cd Magics++-2.6.1**
>**./configure --prefix=/path/to/where/you/install/Magics++-2.6.1**

```
checking build system type... i686-suse-linux
checking host system type... i686-suse-linux
checking target system type... i686-suse-linux

configuring Magics++ 2.6.1

checking whether build environment is sane... yes
checking for a BSD-compatible install... /usr/bin/install -c
… MUCH MORE OUTPUT …
Magics++ is configured as follows for host type: i686-suse-linux

Option            Configure option = Configured value
-----------------------------------------------------
Shared libraries  --enable-shared=no
Static libraries  --enable-static=yes
Use of exceptions --enable-exception=no

Magics++ option
BUFR              --enable-bufr=yes
ODB               --enable-odb=yes
SPOT              --enable-spot=yes
SVG               --enable-svg=yes
OpenGL            --enable-opengl=no
Raster (GIF,PNG)  --enable-raster=yes
Raster TTF fonts  --enable-ttf=yes

Ghostscript       /usr/local/share/gs (7.07.1)
Ghostscript fonts --with-gs-font-dir=default   /usr/share/ghostscript/fonts/

Options used to compile and link:
  CC       = gcc
  CFLAGS   = -g -m32 -ansi -std=c99 -Wall -W
  CPPFLAGS = -I/usr/local/include -I/usr/include/freetype2 -I/usr/include
  CXX      = g++
  CXXFLAGS = -g -m32 -fno-gnu-keywords -ansi -std=c++98 -Wall -W -Wno-deprecated -
I/usr/local/lib/metaps/lib/grib_api/0.9.0/include
  LDFLAGS  = -L/usr/X11R6/lib  -L/usr/local/apps/gd/lib -L/usr/local/lib
  LIBS     = -lexpat -lgd
  …
Please ensure that the MAGPLUS_HOME environment variable is correctly defined!

Magics++ will be installed in   /usr/local/apps/Magics/2.3.0

Currently $MAGPLUS_HOME is     /usr/local/apps/Magics/develop++
```

> **gmake**
> **gmake install**
> **setenv MAGPLUS_HOME /path/to/where/you/install/Magics++-2.6.1**
> **gmake check** *<< running the test programs*
> **cd test/fortran**
> **./coast**

License