MathCode Fortran90 installation instructions for Linux machines and license administration

Version 1.2.4,
April 6, 2011.

Chapter 1 Installation step by step

Please follow these steps for successful *MathCode* F90 installation.

1.1 Check your Mathematica, GCC version

MathCode will work on any Linux distribution with the proper tools installed.

Mathematica 6.0, 7.0, 8.0 are supported on computers with Linux (32-bit and 64-bit).

MathCode relies on compatibility between GCC and G95 versions.
GCC versions between 3.3 and 4.4 were tested. If you have a different GCC version please read the Section 2.1.

1.2 Choice of compilers

MathCode F90 for Linux requires either G95 or Intel Fortran compiler. The Intel Fortran compiler works more stable on 64-bit Linux computers. For 64-bit Linux computers you should chose the G95 release that matches your hardware. The G95 compiler has some stability issues depending on specific 64-bit processor model or OS release; therefore if the tests fail, Intel Fortran compiler is recommended instead. During MathCode installation you should select the compiler of you choice. To change the compiler you should reinstall MathCode.

1.3 Install G95 properly before MathCode installation

MathCode F90 requires the G95 compiler. It is available from **http://www.g95.org**. Download a stable version for "Linux x86" platform.

G95 version 0.91 (March 2008) was tested. Later releases should work as well.

Choose a directory for installation. In order to make possible for other users to run **g95** you should grant read permissions for this directory.

Unpack the downloaded tarball (e.g. **g95-x86-linux.tgz**) in a directory of your choice:

```
tar -zxvf g95-x86-linux.tgz
```

Create a symbolic link from a directory in your **$PATH** (e.g. **~/bin**) to the executable

```
ln -s $PWD/g95-install/bin/*g95* ~/bin/g95
```

You should now be able to run g95 and create executables.

As an alternative, in order to make possible for other users to run **g95** you should use a common directory which is present in everyone's $PATH, and create a symbolic link with name **/usr/bin/g95** or **/usr/local/bin/g95** :

```
sudo ln -s $PWD/g95-install/bin/*g95* /usr/bin/g95
```

In this case make sure that $PWD is readable for other users.

1.4 Installing Intel Fortran compiler before MathCode installation

Intel Fortran Compiler for Linux is available from Intel Coropration, for 32-bit and 64-bit architectures. The installation guidelines provide detailed instructions. When you are ready with installation the command "ifort" should be made available in your PATH, and the command "ifort -v" should report the actual version of the installed compiler. MathCode F90 has been tested with Intel Fortran For Linux Version 11.1.

1.5 Determine your $MachineID

The $MachineID is needed for registration. It is the identity of the machine you want a license for. To find out your $MachineID, evaluate the following in *Mathematica*:

```
$MachineID
```

1.6 Obtain license key for purchased license

You should register to get a key file that will enable you to use the software. If you purchased the software you can register it online at the following URL:

```
http://www.mathcore.com/register.html
```

Please do not use this page for demo (trial) licenses, see **Section 1.7**!

When you start installation of *MathCode* you can click the button **Register** to register your software.

Within two business days you should receive an e-mail with the key file attached. Save the attachment to a file. Remember where you saved it; you will need to select this location during *MathCode* C++ installation.

1.7 Obtaining license key for demo (trial) license

You apply for demo (trial) license using online demo request form at
http://www.mathcore.com/products/mathcode/
and click on **Download Trial version**

When you start installation of *MathCode* you should not click the **Register** button to register your software.

Within two business days you should receive an e-mail with the key file attached. Save the attachment to a file. Remember where you saved it; you will need to select this location during MathCode C++ installation.

1.8 Previous MathCode installations

You can have ony one MathCode installation available in your UNIX account at a time. The setup will disable any previous installation of MathCode C++ or MathCode F90. The current installation is determined by settings in file (which is created during installation):

~/.Mathematica/Applications/MathCode.m

1.9 Different Mathematica installations

An installed MathCode can be used with ***only one*** Mathematica installation. If you switch to a different Mathematica version you must ***re-install*** MathCode. Otherwise difficult linking error messages will occur.

1.10 Check for the latest release

Since MathCode relies on many other software products that often change their versions and properties please **always download the latest version** from the address you get from us together with your key file; currently it is

http://www.mathcore.com/products/mathcode/download/downloadframe.shtml

1.11 Decide whether you need personal installation or root installation.

We recommend you to log in with your personal user name and install MathCode under your own home directory. *MathCode* will be available for you only.

On **Linux** machines it is possible to install MathCode in system directory such as `/usr/local/MathCode`, and make it available for all users of certain *Mathematica* installation, but it causes additional security problems.

As a **root** you can adjust the *Mathematica* installation for this purpose. The `Demos` and `Licensing` subdirectories of *MathCode* installation should be writable for all users. Otherwise the licensing system and demos will not work.

1.12 Installation procedure

Go to the `linux` directory on the *MathCode* CD or obtain the latest release from `www.mathcore.com`.

You obtain file `mathcode-`**linux**-*version*`.tar`

Use command `tar -xvf mathcode-`**linux**-*version*.tar to unpack this archive.

Run the file `install.`*system*, either by `./install.`*system* (preferred) or `sh install.`*system* (if the file is not flagged as executable on the CD) and follow the on-screen instructions.

The installation script compiles all necessary MathCode runtime libraries, therefore you do not need to care about libc library versions (as it was in MathCode C++ for Linux 1.2.2 and earlier).

If you have any special settings (PATH, GCC flags etc.) when you compile the runtime library, these settings should be preserved when you use MathCode C++ for compilation.

Please run the test `Demos/Verify/testlinux.m` after installation.

1.13 Parallel installations

You can install several installations of MathCode, but only one of them (the latest one) will be used within Mathematica.

1.14 Uninstall

At the end of installation the script tells the name of a file (uninstall-*system*.sh) which contains commands for uninstall.

Chapter 2 Advanced adjustments

2.1 Using different GCC version

If you have a different and unexpected GCC release, then installation may stop. Please install another gcc toolkit and place its directory first in the path, so that shell commands "gcc" and "g++" invoke the tools of different version.
Execute the command Run["echo $PATH"] from Mathematica to see the actual path.

In addition to this you will need to set up a symbolic link so that commands invoked from within Mathematica sessions search for correct g++ binary of correct gcc version. When shell commands are executed from within Mathematica, a modified path is used. Execute the command Run["echo $PATH"] from Mathematica to see the actual path.
Typical commands to adjust the g++ in use can be:

```
su
cd /usr/local/Wolfram/Mathematica/6.0/Executables
ln -s /usr/local/gcc/3.3.4/bin/g++ .
ln -s /usr/local/gcc/3.3.4/bin/gcc .
```

2.2 Using a different Fortran90 compiler

MathCore Engineering AB provides you with scripts needed for different Fortran90 compiler as a consultancy service.

Steps needed for attaching a different Fortran90 compiler on any UNIX-like operating system are:

1. Study **MathCodeConfig.m**, it refers to **u95.unx** and **unix.tmpl**.
2. Study how **unix.tmpl** makes **Global.cmd**.
3. Study how **Global.cmd** calls **System/u95.unx**.
4. Study how **System/u95.unx** calls **lib/sheep/u95.mak**.
5. Change ***.f90** and ***.c** files in **lib/sheep** so that they can be compiled by your Fortran90 and C++ compiler.
6. Change **lib/sheep/u95.mak** so that **sheep.lib** can be created.
7. Investigate whether your Fortran90 and C++ compiler can compile files like **Global.***
8. Change **System/u95.unx** so that GlobalML.exe can be compiled and linked.
9. Possibly adjust **unix.tmpl** if necessary.

Chapter 3 License management

3.1 What are licenses?

For each machine you wish to run *MathCode* on, you should obtain one key file containing the license. *MathCode* uses the same MathID as *Mathematica* does to distinguish between machines. A key file is a text file containing a mix of letters and digits. Key files should be put into the `Licensing` subdirectory of the *MathCode* installation. The names of the key files do not matter.

3.2 Adding a license

When you register for a new *MathCode* license, you will receive a file that should be put in the `Licensing` subdirectory of your *MathCode* installation.

3.3 The license index file

MathCode will use an index file `index.m` in the `Licensing` directory to speed up license lookups. If a new license is added, `index.m` is rebuilt automatically as needed.

If you experience problems with the licensing, you can remove the `index.m` file, forcing *MathCode* to rebuild it on the next license check.

For a site installation, users might not have write permissions to the `Licensing` subdirectory. In this case, the system administrator should rebuild the index file by evaluating the following in *Mathematica*:

```
Needs["MathCode`"];
RebuildIndex[ToFileName[{$MCRoot,"Licensing"}]];
```

If `index.m` didn't exist, you will se an error message about opening it. This error message can safely be ignored.

Chapter 4 More on compiler definitions

The file `MathCodeConfig.m` in the main *MathCode* directory controls the *MathCode* runtime configuration. This file is really a *Mathematica* package that contains some configuration directives; currently `DefineCompiler[]` and `DefaultCompiler[]`.

`DefineCompiler[]` is used to associate a symbolic compiler name (a string) with a make file, a command template, and a build command. You don't normally need to bother with these details.

`DefaultCompiler[]` is used to select the default compiler definition for a language. Currently the only language supported for code generation is C++. In `MathCodeConfig.m` you might find a line

```
DefaultCompiler["C++"->"mingw32"];
```

This tells *MathCode* to use the included `"mingw32"` compiler definition when generating C++ code. If you wish to use Visual C++ instead (assuming you are on the Windows platform), you should change this to read:

```
DefaultCompiler["C++"->"vc60"];
```

If there are several `DefaultCompiler` definitions, the last one is taken into account.

Using a different compiler can be easier than that, with the new options to `CompilePackage[]`, `MakeBinary[]` and `BuildCode[]`.

`CompilePackage[]` takes a `Language` option (currently only C++ is supported). MathCode will then use the default compiler for the specified language. Example:

```
CompilePackage[Language->"C++"];
```

`MakeBinary[]` takes a Compiler option; the option value should be one of the symbolic names (strings) defined using `DefineCompiler[]`. The Compiler option to `MakeBinary[]` overrides the default compiler specified for the selected language. Example:

```
MakeBinary[Compiler->"g++"];
```

As usual, `BuildCode[]` can be given both `CompilePackage[]` and `MakeBinary[]` options. The following example will generate C++ code and use the `"CC"` compiler to compile it, overriding any default specification:

```
BuildCode[Language->"C++", Compiler->"CC"];
```

The above example assumes that you are using *MathCode* on Solaris.