MathCode C++ installation instructions for Linux and Mac OS X machines and license administration

Version 1.4.4, 16 March 2011.

Chapter 1 Installation step by step

Please follow these steps for successful *MathCode* C++ installation.

1.1 Check your Mathematica and GCC version

•**Linux***: MathCode* will work on any Linux distribution with the proper tools installed. *Mathematica* 5.2, 6.0, 7.0, 8.0 are supported on computers with Linux (32-bit and 64-bit).

•**MacOSX**: *Mathematica* 6.0, 7.0 and 8.0 are supported only, for Intel processors only. Versions 5.2.2 and earlier are not supported. You need the GCC compiler which is included in free XCode tool (http://developer.apple.com/tools/download/). These releses were tested:

- Mac OS X version 10.4.* ("Tiger") with XCode 2.4.1,
- Mac OS X version 10.5.* ("Leopard") with XCode 3.1.1,
- Mac OS X version 10.6.* ("SLeopard") with XCode 3.2.1

MathCode relies on compatibility between GCC versions. GCC versions between 3.3 and 4.2 were tested. If you have a different GCC version please read the Section 1.2.

1.2 Using different GCC version

If you have a different 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. 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. Typical commands can be:

```
su
cd /usr/local/Wolfram/Mathematica/5.1/Executables
ln -s /usr/local/gcc/3.3.4/bin/g++ .
```

1.3 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
```

1.4 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!

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.

1.5 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.

1.6 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

1.7 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.

On **Mac OS X** machines you must install MathCode under your own home directory. The user name should not be "**root**".

1.8 Installation procedure

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

You obtain file `mathcode-`*`system`*`-`*`version`*`.tar` (here *system* can be **linux** or **macosx**).

Use command `tar -xvf mathcode-`*`system`*`-`*`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.

1.9 Parallel installations

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

1.10 Uninstall

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

Chapter 2 License management

2.1 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.

2.2 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.

2.3 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 3 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->"g++"];
```