Getting Started Guide for Cmajor 1.5.0

Seppo Laakko

August 23, 2016

1 Installation in Windows

1.1 Prerequisites

Note: You must uninstall any previous Cmajor version before installing this version (1.5.0).

It is also recommended that you delete the `%APPDATA%\Cmajor` directory before installing this version.

- Download and install MinGW-w64 GCC:
  http://sourceforge.net/projects/mingw-w64/files/Toolchains%20targetting%20Win32/Personal%20Builds/mingw-builds/installer/mingw-w64-install.exe/download

  Installation settings for my system (64-bit Windows):

  - Version 5.4.0
  - Architecture: x86_64
  - Threads: **posix**
  - Exception: sjlj
  - Build revision: 0

  Installation settings for 32-bit Windows:

  - Version 5.4.0
  - Architecture: i686
  - Threads: **posix**
  - Exception: sjlj
  - Build revision: 0

  Note: Threads setting must be "posix".

  Note: As of this writing GCC version 6.1.0 is not able to compile all Cmajor example programs. GCC version 5.4.0 is the latest tested version.

  After installation insert the bin-directory to the **PATH** environment variable. In my system this is
  `C:\mingw-w64\mingw64\bin` directory.

- Download and install Visual C++ Redistributable for Visual Studio 2015:
  64-bit: http://sourceforge.net/projects/cmajor/files/1.5.0/vcredist_x64.exe/download
  32-bit: http://sourceforge.net/projects/cmajor/files/1.5.0/vcredist_x86.exe/download

1.2 Cmajor Installation

- Download and run **cmajor-1.5.0-win-x86-setup.exe** (for 32-bit Windows) or **cmajor-1.5.0-win-x64-setup.exe** (for 64-bit Windows).
- Cmajor is installed by default to `C:\Program Files\Cmajor` directory (under 32-bit Windows) or to `C:\Program Files (x86)\Cmajor` directory (32-bit and 64-bit versions under 64-bit Windows).

  Note: The x64 version is also installed by default under `C:\Program Files (x86)` directory although the programs are genuingly 64-bit versions). This is due to restrictions of InstallShield Limited Edition.
- The setup adds `C:\Program Files\Cmajor\bin` directory or `C:\Program Files (x86)\Cmajor\bin` directory to your system's **PATH** environment variable, so the Cmajor programs can be executed from any directory from the command prompt without specifying full paths.
- The setup also adds a **CM_LIBRARY_PATH** environment variable and sets it to contain a path to the Cmajor System Library directory that is `%APPDATA%\Cmajor\system`. If you need to modify the **CM_LIBRARY_PATH** environment variable, you can find it from the Advanced System Settings pane in the System Control Panel.

  In my computer the `%APPDATA%` points actually to the `C:\Users\Seppo\AppData\Roaming\` directory. The **AppData** folder is hidden by default. To see it you will have to modify the settings in the *Folder Options* Control Panel.
- The setup adds an icon to **Cmajor Development Environment** to the desktop.
- After installation you have to build the Cmajor System Library.

1.3 Building the Cmajor System Library

- Option 1: using IDE:

  Start **Cmajor Development Environment**.

  - Open the `File|Built-in Projects|System Library` project.
  - Choose `Build|Batch build...` command
  - Click the `Select All` button or select the configurations you plan to use.
  - Click the Rebuild... button.
  - Now the Cmajor system is ready for building user projects.

  Option 2: using batch file:

- Open command prompt and change to Cmajor system directory by issuing command `cd %APPDATA%\Cmajor\system`.
- Run **build.bat**. This builds the Cmajor System Libary for each backend (LLVM/C) and configuration (debug/release/profile/full).
- Now the Cmajor system is ready for building user projects.

1.4 Troubleshooting

- **could not obtain LLVM compiler version (llc –version)...**

  Setup seems not to properly notify change of environment variables and paths. Often restarting the Cmajor Development Environment helps. If that doesn't help, logging out and back in seems to solve the problem.

- **library reference 'system.cml' not found.**

  You have to build the System Library for the used configuration (debug/release/profile/full) and backend (LLVM/C) first.

- **gcc is not recognized as an internal or external command, operable program or batch file.**

  or **'ar' is not recognized as an internal or external command, operable program or batch file.**

  or **Cannot start llc.exe because libgcc_s_sjlj-1.dll is missing**.

  The bin directory of mingw-w64
  (in my machine
  `C:\mingw-w64\mingw64\bin`) must be in the PATH environment variable.

- **undefined reference to 'WinMain' collect2.exe: error: ld returned 1 exit status**

  You have probably 32-bit Cmajor and 64-bit MinGW-w64's gcc. Both must be either 32-bit or 64-bit.

- Build seems to succeed but program does not work, or other mysterious error.

  Try **rebuild** command (-R option) or **clean** and then **build**.

- Sometimes IDE messes up things or error messages from command line compiler are not propagated to IDE properly.

  Try using the command line compiler (cmc.exe) so you can get more information.

  Programs compiled for C backend both in debug and release configuration have debug symbols in them, so that they can be debugged also using gdb.

- How to generate 32-bit executables in a 64-bit system.

  Use 32-bit MinGW-w64 (i686) and 32-bit Cmajor.

2 Installation in Linux

2.1 Prerequisites

- GCC with g++ compiler.
  Must be recent enough to compile C++11 code.

- Download, build and install Boost C++ libraries (http://www.boost.org/). At minimum you will need to build and install the **filesystem** and **iostreams** libraries:
  (`./b2 --with-filesystem --with-iostreams install`)

  The **iostreams** library uses **zlib** (http://www.zlib.net/) and **libbz2** (http://www.bzip.org/) compressions libraries. The compression filters are not needed by Cmajor so you can disable them by using the -s option in the build command:
  `./b2 --with-filesystem --with-iostreams -sNO_ZLIB -sNO_BZIP2=1 install`.

  However if you want to use compression filters with the **iostreams** library, see instructions in http://www.boost.org/doc/libs/1_60_0/libs/iostreams/doc/index.html how to use them.

- Obtain LLVM tools (http://llvm.org/). You can use LLVM tools that come with your Linux distribution or compile them from sources. Easest is to install the tools that come with your Linux distribution although they are probably a bit dated. For example in Ubuntu this is done by executing command `sudo apt-get install llvm`. If you try to execute the LLVM compiler `llc --version` without it being installed, the system will probably tell you in which package the LLVM tools can be found and how to install them.

  If you want to compile the tools from sources, the LLVM releases can be found in http://llvm.org/releases/. Build instructions are in http://llvm.org/docs/CMake.html.

  If you want to use absolutely latest version of LLVM tools, http://llvm.org/docs/GettingStarted.html contains instructions how to obtain and build them.

2.2 Cmajor Installation

- Download and extract **cmajor-1.5.0.tar.gz** or **cmajor-1.5.0.tar.bz2** to some directory here called `<cmajor>`.

- Change to `<cmajor>` directory and run `make` and then `[sudo] make install`.

  By default Cmajor tools are installed to /usr/bin directory (*prefix=/usr*). If you want to install to a different path you can use the prefix setting:
  `make prefix=/where/to/install install`.

  By default release versions of the tools are built and installed. To build and install debug versions use following commands:
  `make config=debug` and then `make config=debug install`. The debug versions of the tools have 'd' appended to the name of the executable, so for example debug version of the Cmajor compiler is named **cmcd**.

- Set an environment variable `CM_LIBRARY_PATH` to contain path to `<cmajor>/system` directory. You may want to insert a statement like:

`export CM_LIBRARY_PATH=/path/to/cmajor/system`
into you .bashrc script.

2.3 Building the System Library

- Run command `make sys` from `<cmajor>` directory.
- Now the Cmajor system is ready for building user projects.

2.4 Troubleshooting

- library reference 'system.cml' not found

  You have to build the System Library for the used configuration (debug/release/profile/full) and backend (LLVM/C) first.

- sh: 1: llc: not found

  You have probably not installed LLVM tools.

- Build seems to succeed but program does not work, or other mysterious error.

  Try **rebuild** command (-R option) or **clean** and then **build**.

3 Contact

seppo.laakko@pp.inet.fi