Getting Started Guide for Cmajor 0.9.1

Seppo Laakko

September 29, 2014

1 Installation in Windows

1.1 Prerequisites

Note: the prerequisites have changed since version 0.8.0. Now POSIX threads should be specified in the mingw_w64 installation. Please remove older mingw_w64's gcc from PATH.

You must uninstall any previous Cmajor version before installing this version (0.9.1).

- Download LLVM tools for mingw_w64:

  http://sourceforge.net/projects/clangonwin/files/MingwBuild/3.5/ClangToolsforMingw-w6
  7z/download

  After downloading extract the package to some directory of your choice, and insert the bin-directory to the **PATH** environment variable. In my system this is `C:\Programming\ClangToolsforMingw-w64-207083-x64\bin` directory.

- 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:

  - Version 4.9.1
  - Architecture: x86_64
  - Threads: **posix**
  - Exception: sjlj
  - Build revision: 1

  Note: Threads setting must be "posix".

  After installation insert the bin-directory to the **PATH** environment variable. In my system this is
  `C:\Program Files\mingw-w64\x86_64-4.9.1-posix-sjlj-rt_v3-rev1\mingw64\bin`
  directory.

1.2 Cmajor Installation

- Download and run **cmajor-0.9.1-win-x86-setup.exe** (for 32-bit Windows) or **cmajor-0.9.1-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. You may want to change the install location under `C:\Program Files` directory in this case.
- The setup adds `C:\Program Files\Cmajor\bin` directory (or whereever you installed it) to your systems **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 System Library from the **Cmajor Development Environment**.

1.3 Building the System Library

- Start **Cmajor Development Environment**.
- Open the `File|Built-in Projects|System Library` project.
- Run `Build|Rebuild Solution` command for the `debug` configuration.
- Select `release` configuration from the configuration combo box and run `Build|Rebuild Solution` command for the `release` configuration.
- Select `debug` configuration from the configuration combo box and `C` backend from the backend combo box and run `Build|Rebuild Solution` command for the `C` backend and the `debug` configuration.
- Select `release` configuration from the configuration combo box and run `Build|Rebuild Solution` command for the `C` backend and for the `release` configuration.
- Now the Cmajor system is ready for building user projects.

Note: If you are updating from previous Cmajor version, it is important to issue the **rebuild** command (not just build command), because System Library directories are not cleared when uninstalling Cmajor.

1.4 Troubleshooting

- cmc I2 error **library reference 'system.cml' not found**.

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

- cmc I2 error **'llc' is not recognized as an internal or external command, operable program or batch file**.

  The bin directory of the LLVM-tools
  (in my machine `C:\Programming\ClangToolsforMingw-w64-207083-x64`) must be in the **PATH** environment variable.

- **Cannot start llc.exe because libgcc_s_sjlj-1.dll is missing** (or something like that).

  The bin directory of mingw_w6
  (in my machine
  `C:\Program Files\mingw-w64\x86_64-4.9.1-posix-sjlj-rt_v3-rev1\mingw64\bin`)
  must be in the PATH environment variable.

- The compiler is still in experimental stage, so surely many bugs exist...

Hope this helps!

2 Installation in Linux

2.1 Prerequisites

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

- Download, build and install Boost (http://www.boost.org/). You will need to build and install the filesystem library.

- Donwload, build and install LLVM tools (http://llvm.org/). Installation instructions can be found in http://llvm.org/docs/GettingStarted.html document.

2.2 Cmajor Installation

- Download and extract **cmajor-0.9.1-src.tar.gz** to some directory here called `<cmajor>`.

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

- 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) and backend (LLVM/C) first.

- sh: 1: llc: not found

  You have probably not installed LLVM tools.

- The compiler is still in experimental stage, so surely many bugs exist...

  Hope this helps!