mex-moos: A Matlab Interface for MOOS:V10 Communications

Paul Newman, University of Oxford

....ten years on

10.0.2

Contents

1 Getting Started - Acquiring and Building MOOS

If you already have MOOS::V10 installed and built you can skip this section.

1.1 Before you start you will need...

- a working compiler like `gcc` or `clang`
- `CMake` installed
- `git` installed (well actually this is optional as you can download the source code as .zip file and we won't make much use of git in this tutorial)
- if you are using a mac on OSX 10.9 and an R2013 matlab release, please look at Section 7 which will explain why for you an extra step will be invovled.

1.2 Downloading and Building MOOS V10

We shall begin where we should and check out a version of MOOS-V10 from a git repos. We will follow good practice and do an out of place build - the source code will go in "src" and we will build in "build". We will also, after fetching the source switch to the "devel" branch because here we are living on the edge [1].

```
pmn@mac:~$ mkdir core-moos-v10
pmn@mac:~$ cd core-moos-v10
pmn@mac:~$ git clone https://github.com/themoos/core-moos.git src
pmn@mac:~$ mkdir build
pmn@mac:~$ cd build
pmn@mac:~$ ccmake ../src
```

At this point you should, after hitting 'c' a couple of times be presented with a CMake screen that looks like that shown in Figure 1.1 (note some of the entries are platform dependent so don't worry if what you see is not identical to this).

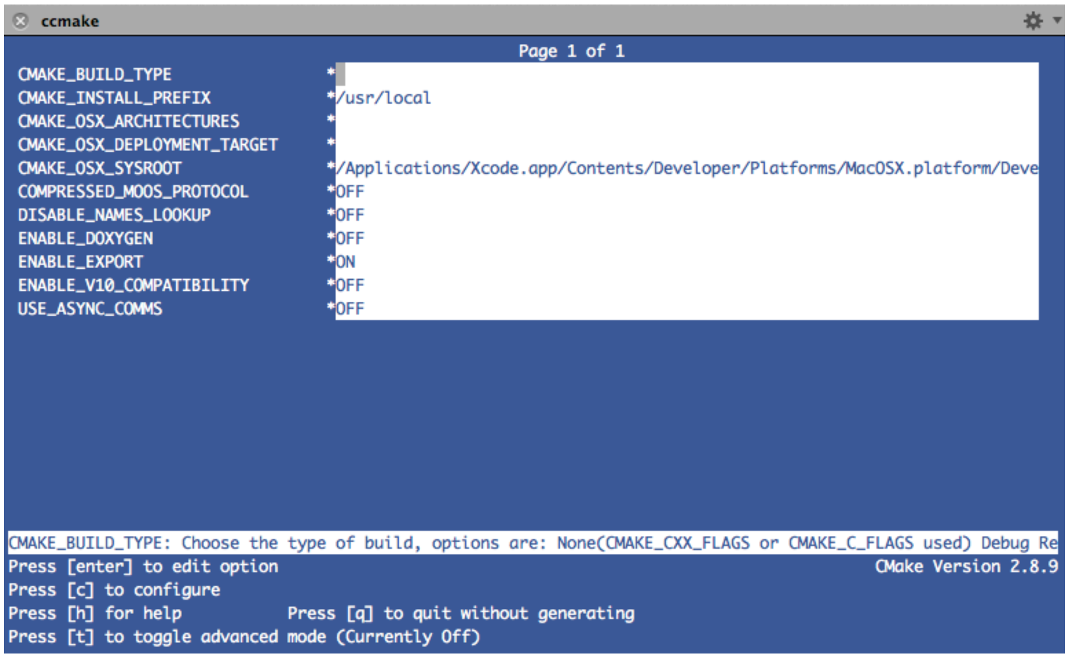
You are are now in a position to build the MOOS. So press 'c' until 'g' appears, then press 'g' and you are good to go. Then at the terminal prompt type 'make' to build the project. Two directories should have been created **bin** and **lib.** In lib you will see `libMOOS.a` and in `bin` you will find the newly created `MOOSDB` and some other fabulous tools like UMM, MTM and MQOS. Nice job.

2 Acquiring mex-moos

MEX-MOOS is hosted on github (at time of writing) but where ever you get it from the build path looks somethign like this.

```
pmn@mac:~$ mkdir mex-moos
pmn@mac:~$ cd mex-moos
pmn@mac:~$ git clone https://github.com/themoos/mex-moos.git src
pmn@mac:~$ mkdir build
pmn@mac:~$ cd build
pmn@mac:~$ ccmake ../src
```

[1] if you want to know what branches are available type `git branch`



```
ccmake
                                         Page 1 of 1
 CMAKE_BUILD_TYPE                *
 CMAKE_INSTALL_PREFIX            */usr/local
 CMAKE_OSX_ARCHITECTURES         *
 CMAKE_OSX_DEPLOYMENT_TARGET     *
 CMAKE_OSX_SYSROOT               */Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Deve
 COMPRESSED_MOOS_PROTOCOL        *OFF
 DISABLE_NAMES_LOOKUP            *OFF
 ENABLE_DOXYGEN                  *OFF
 ENABLE_EXPORT                   *ON
 ENABLE_V10_COMPATIBILITY        *OFF
 USE_ASYNC_COMMS                 *OFF

CMAKE_BUILD_TYPE: Choose the type of build, options are: None(CMAKE_CXX_FLAGS or CMAKE_C_FLAGS used) Debug Re
Press [enter] to edit option                                                          CMake Version 2.8.9
Press [c] to configure
Press [h] for help           Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```

Fig. 1.1: The default build screen for MOOS V10.

At the end of this process you should see in the build directory a mex file. It will be called `mexmoos.XYZ` where XYZ depends on you operating system. For example on a 64 bit Apple machine it is called `mexmoos.mexmaci64`.

2.1 A Quick Health Check

Now open matlab and add the path to mexmoos to your session. Alternatively just “cd” to that directory. Then to check everything is working nicely

```
>> mexmoos('help')
```

You should see help text being printed out.

3 Initialisation

Initialisation is executed with a a single string parameter

```
>> mexmoos('init')
```

this starts up a MOOS connection with default settings - looking for a MOOSDB serving on port 9000 on localhost with a default name of “mexmoos”. You can overload any of these parameters by using a parameter value pair for MOOSNAME, SERVERPORT, and SERVERHOST.

```
>> mexmoos('help', 'MOOSNAME', 'darkness', 'SERVERPORT', '9001')
```

3.1 Registration

Registration for MOOS data is done with the the “REGISTER” command. You also get to specify the name of the variable you are interested in and of course the maximum rate at which you want to receive notifications.

```
>> mexmoos('REGISTER','vader_count',0.2) % 5Hz max
>> mexmoos('REGISTER','cookie_count',0.0) % receive every change
```

4 Notification

Notification (the sending of data) is executed using the “NOTIFY” key word followed by a string which is the name of the message (variable) you are sending. Then comes the data - it can be a single double, a string or a byte array.

```
>> mexmoos('NOTIFY','varA',pi)
>> mexmoos('NOTIFY','varB','a remarkable story in a string')
>> mexmoos('NOTIFY','varA',zeros(1,3,'uint8'))
```

5 Receiving

If you have registed for a variable, according to the period you set in the registration step, messages will be collected for you behind the scenes. You retrieve this “mail” with the “FETCH” command. This returns to you a struct array of messages. You can see the message field listed below

```
>> msgs = mexmoos('FETCH');
>> msgs
1x3 struct array with fields:
    KEY
    TYPE
    TIME
    STR
    DBL
    BIN
    SRC
    ORIGINATING_COMMUNITY
```

6 Closing

You can explicitly close a connection with “CLOSE”

```
>> mexmoos(CLOSE,')
```

or you can simple call “`clear mexmoos`” which will unload the mex-moos library from Matlab.

7 Other Notes

7.1 The libstdc++ issue

On Mac platforms certain combinations of OSX and Matlab caused a world of pain because MATLAB itlsef linked against libstdc++ while libc++ was the system standard. This would lead to a wretched world if you built MOOS and then mex-moos linked against that but was called from inside matlab expecting libstdc++. The solution is that for these known combinations mex-moos will bring down its own provate version of MOOS and build that using libstdc++ explicitly. But for this to work you need the release called release-0.9

7.2 Capitalisation

Commands are not case sensirive so for example you can use “`fetch`” instead of “FETCH”.