Using the new OSApi in Eclipse and Makefiles

Søren Hansen <shan@iha.dk>

April 16, 2012

Contents

Preamble

1.1 Purpose

This is a small tutorial describing how you can use the OSApi in Eclipse and Makefiles. An important precondition is of course that the OSApi has been downloaded and extracted. In the following is it presumed that this has been done and the extracted files placed in `/home/stud/apps/OSApi`.

The tutorial is thus divided into two parts, the first being on how `eclipse` is to be set up and the second describing how a very simple `makefile` using the OSApi could look like.

For the eclipse case it is also assumed that an appropriate project has been made and that it is ready for OSApi inclusion.

NOTE: Be advised compiler versions shown may not match your version, as this is unimportant.

Setting up an Eclipse project for utilizing the OSApi library

A project called TestOSApi has been created for the task of testing the OSApi lib. Note that changing compiler to compile the project for the target is not covered here. It it left for the reader to do.

First thing to do is to ensure that we have included the OSApi include path in the projects include path list. See below for how this is to be done. First though, right click on the soluDon and choose *Properties*.

Do note that you have to choose which configuraDon the changes have to apply to. In general, the changes should apply to *[all configurations]*, however when creating debug or release versions you might want a debug or release version of the OSApi respectively. More on this later.

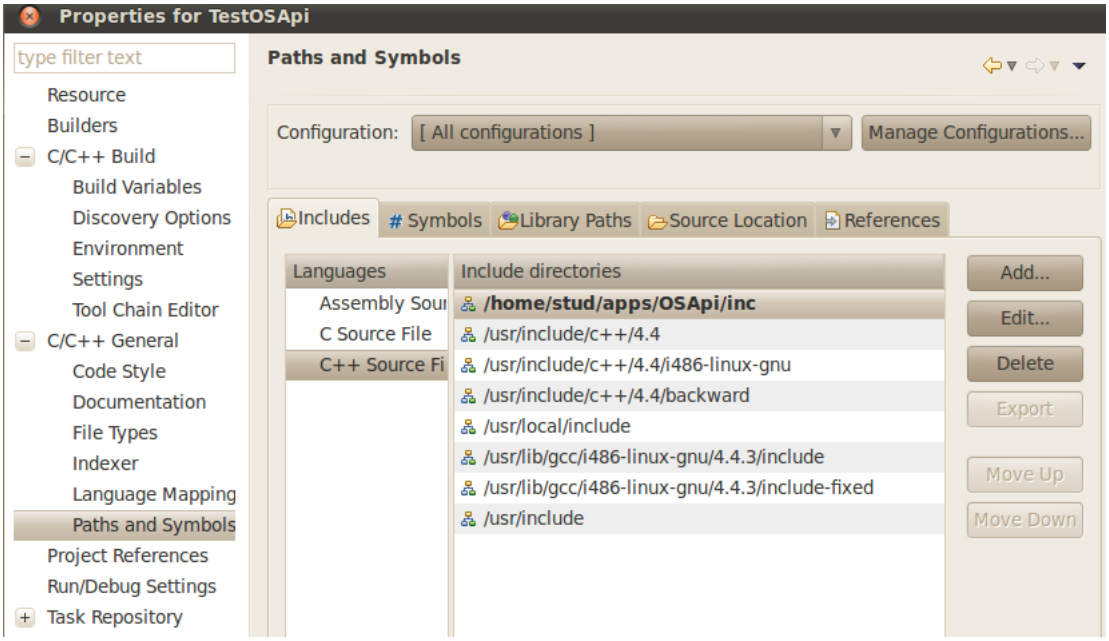

Figure 2.1: Setting up include path

The next thing is to ensure that we get access to the linux implementation within the OSApi. This is done by setting a define called `OS_LINUX`. Shown in the next picture:

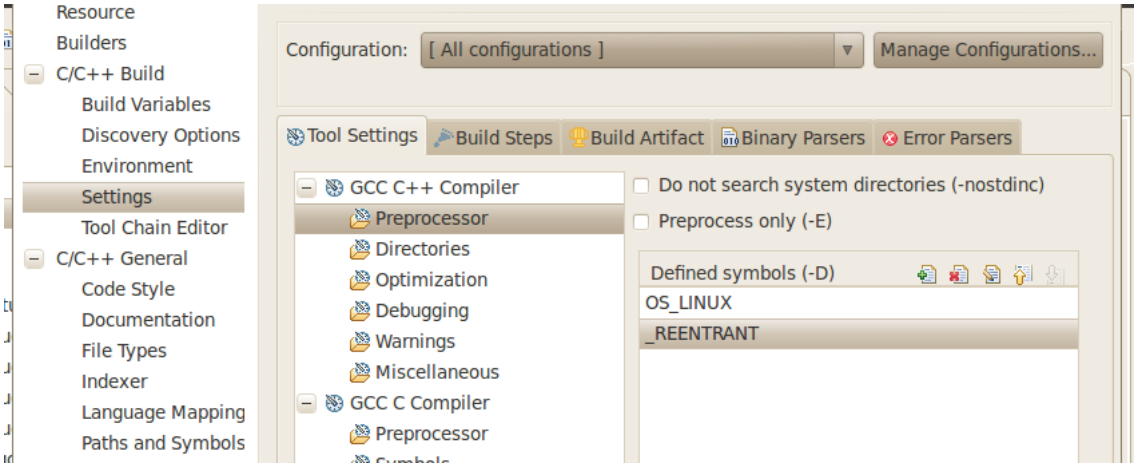

Figure 2.2: Setting up defines

At this point we can start utilizing the different parts of OSApi. Note that we have added `_REENTRANT` here as well. At some later point in time when our program has been created, we need to ensure that the compiler has access to the library file, where most of the OSApi implementation reside.

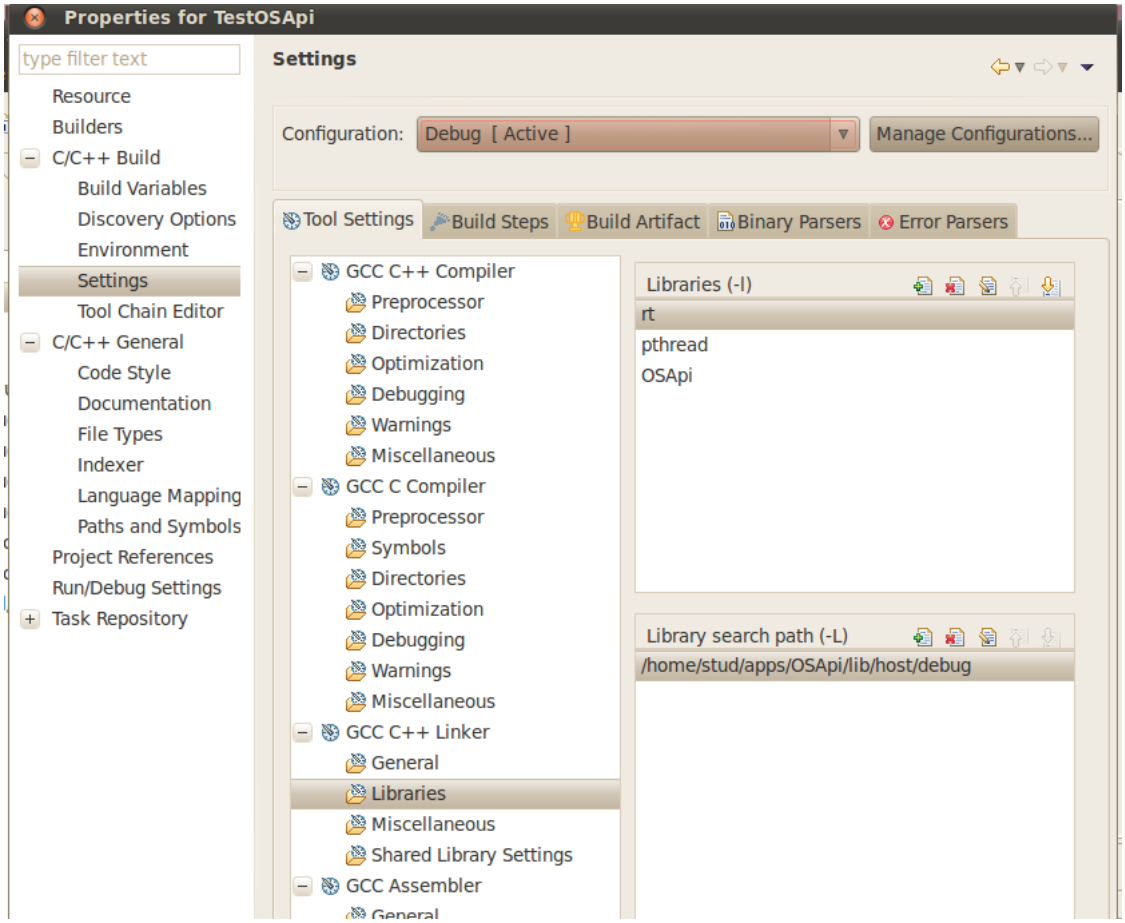

Figure 2.3: Setting up library path

It is at this point that you might want to insert two different paths, one for a debug version and one for a release version. In the above picture you will notice that it is the Debug configuration

that changes are applied to. Thus it comes as no surprise that the OSApi library path inserted, is for the debug version of the said library. Having notified the system of where to find the library, we proceed to specify which libraries should be linked to our application.

Finally we have reached the point where we can compile our program based on the OSApi. Happy programming!

Setting up a Makefile for utilizing the OSApi library

To explain how this can be done, a simple makefile building only a single main.cpp file is listed below:

```
1  # List of files to build
2  SRCS=main.cpp
3  # Determine whether this is a debug build or not
4  ifdef DEBUG
5  CFLAGS=-ggdb -O0
6  LIBPATH=/home/stud/apps/OSApi/lib/host/debug
7  else
8  CFLAGS=-O2
9  LIBPATH=/home/stud/apps/OSApi/lib/host/release
10 endif
11 # Setup the CFLAGS to ensure that the relevant warnings, includes and
       defines.
12 CFLAGS+=-Wall -D_REENTRANT -DOS_LINUX -I/home/stud/apps/OSApi/inc
13 LIBS=rt pthread
14
15 # Note how the flags are part of the compilation of .cpp file(s).
16 main.o: $(SRCS)
17     g++ $(CFLAGS) -c main.cpp
18 # Then again, note how the flags are NOT part of the linking process
19 main: main.o
20     g++ -o main main.o -L$(LIBPATH) -lrt -lpthread -lOSApi
21
22 all: main
23
24 clean:
25     rm -f *.o main
```

Listing 3.1: Simple Makefile

Revision History

Version	Date	Author(s)	Changes
1.0	14/10-2011	SHAN	Iniatial version
1.0	14/10-2011	SHAN	Reviewed and rewritten in Latex