Using the DCMTK in Eclipse

A guide for using the Offis-Dicom toolkit with Eclipse.

Frank Blaauw & Spencer Shaw
9-3-2009

Using the DCMTK in Eclipse

By Spencer Shaw and Frank Blaauw.

This guide is written for development on the Playstation 3 and describes how to install the Cell Broadband Engine FS-Simulaotr on a PC (with FC9). This document was written for the UMCG department Radiology. For this guide the README and INSTALL file included in the DCMTK package were used. We also got help from the people at the DICOM@OFFICE forums, thanks a lot guys!

In no event will the UMCG nor the writers of this document be liable for any damage arising directly or indirectly from any use of the information obtained from this document.

This guide and many other guides can be downloaded from:
http://code.google.com/p/fedora-cell-project/

If you encounter any errors on using this document, please read the inform us via the google-code page or google-group.

Table of Contents

1. Requirements for installation.

The things you might need for this guide is described below.

1.1 Minimal Requirements.

- ✓ A PC with Linux (preferably Fedora Core 9).
- ✓ An internet connection.
- ✓ Eclipse 3.4
- ✓ This guide

For this guide, we used Fedora Core 9 (Linux). We assume you are using a version of Fedora Linux or another Linux distribution as well. We are not sure if using this guide on another distribution gives the same results, some things might not work.

2. The DCMTK installation

This chapter will describe the basics for installing the DCMTK on your system. We start off by downloading the DCMTK from the DICOM@OFFICE website. When the downloading has finished we will continue by compiling and installing the toolkit.

1. Download the DCMTK from the dcmtk.org website

   http://dcmtk.org/dcmtk.php.en

   for this guide we used dcmtk-3.5.4.tar.gz

2. Un - tar the downloaded file using the tar command.

   cd <Download location, default: /user/<username>/Download/>

   tar -x dcmtk-3.5.4.tar.gz

3. Navigate to the dcmtk-3.5.4 directory

   cd dcmtk-3.5.4

4. Run the configure file to make the Makefiles.

   ./configure

5. When the configure has finished, run the make all command to compile all the files. For the eclipse installation, its useful to write the output to a file because we will need this later on.

   make all > output_make.txt

   This should take a while. If you get errors / compiling does not start / your output_make.txt contains errors, please read the readme file or install file included in the DCMTK directory.

6. When making has finished, install the files .

   su -

<root password>

make install

7. Your DCMTK has now been installed. To test if everything was done correct, you could run one of the programs included in the package, for example dcmdump.

   dcmdata/apps/./dcmdump <dicom file>

   If the installation went correct, you should now get the header info of your Dicom file.

For removing your DCMTK or if you want to do specific things with the DCMTK, we'd like to refer to the INSALL file included in the DCMTK package.

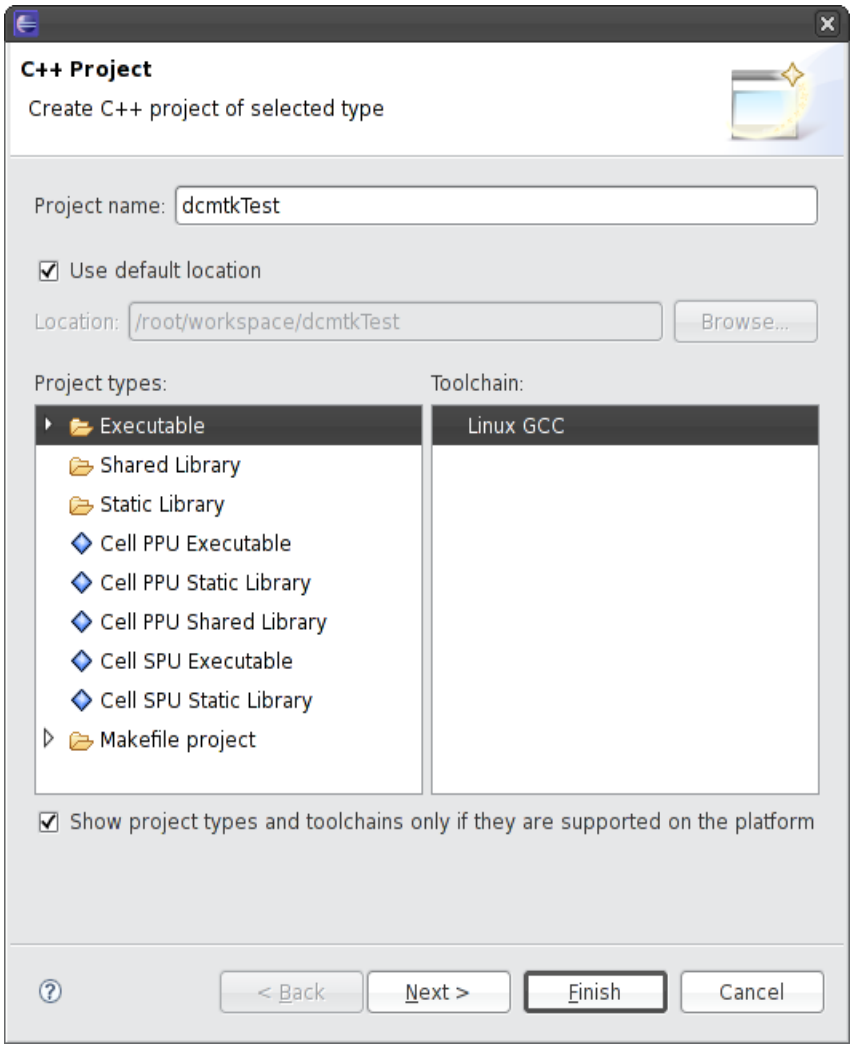
3. Integration with Eclipse.

We will now try to compile the dcmdump program using eclipse. This doesn't seem to be hard, but if you don't know where to look, it sure is a herculean task!

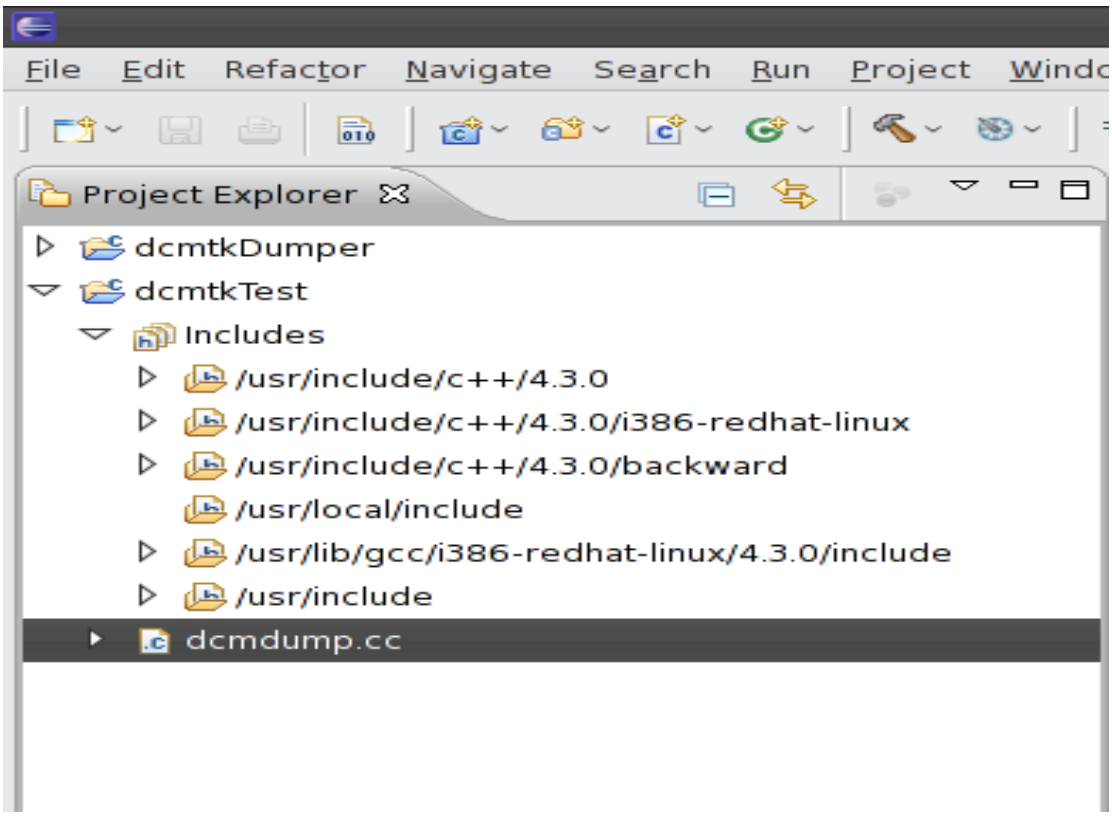
If you don't have eclipse (with CDT) yet, please download the latest version of it from http://eclipse.org.

For this tutorial we used Eclipse 3.4 running on the IBM Java VM (*ibm-java2-i386-jre*).

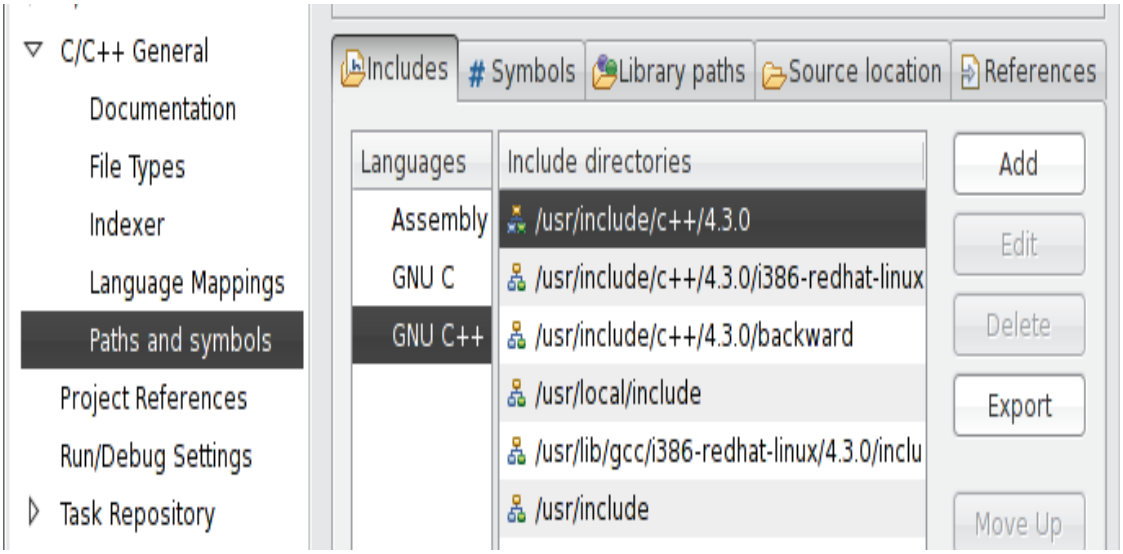
1. Create a new C/C++ Project with the following settings and click finish:



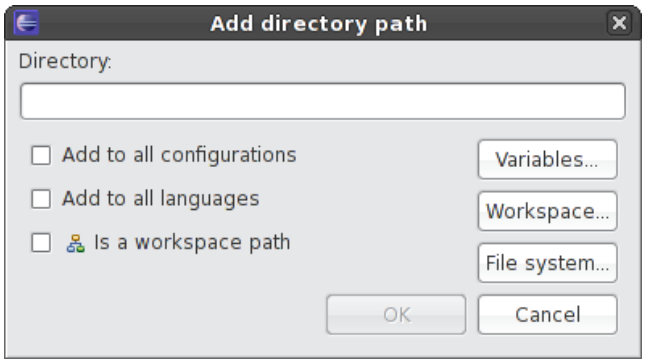
2. Drag the dcmdump.cc in the project browser. This file can be found in the directory: *dcmtk-3.5.4/dcmdata/apps/dcmdump.cc*



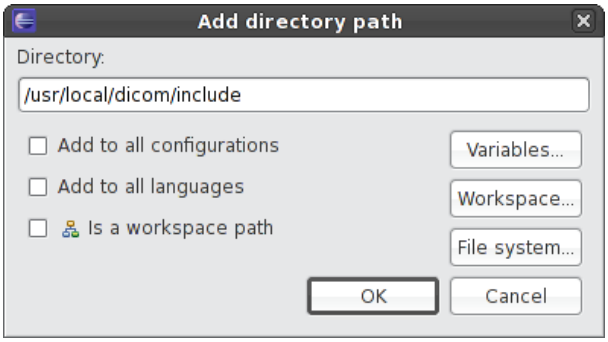
3. Go to the properties of your C++ project (hotkey: *Alt + Enter* or Right mouse button on *dcmtkTest > Properties*) and navigate to C/C++ General > Paths and Symbols and click your C++ compiler (GNU C++ in our example).



5. Click the 'Add' button. A screen should popup.



6. In this screen select the place you installed your DCMTK to (standard */usr/local/dicom/include*) and click OK.



If you don't have an include directory in the */usr/local/dicom* directory, download the get_includes.sh file from the same site you've got this file from as well (http://code.google.com/p/dcmtk-and-eclipse/) this script will gather all includes from your dcmtk-3.5.4 directory and merge them in the directory */usr/local/dicom/include*.

7. Now the tricky part. Open your *output_make.txt* and search for dcmdump this file is located in your *dcmtk-3.5.4/* directory.

8. This search should result in two lines similar to these:

*c++ -DHAVE_CONFIG_H -DNDEBUG -c -I. -I. -I../include -I../../config/include -I../../ofstd/include *

-O -D_REENTRANT -D_XOPEN_SOURCE_EXTENDED -D_XOPEN_SOURCE=500 -D_BSD_SOURCE -D_BSD_COMPAT -D_OSF_SOURCE -D_POSIX_C_SOURCE=199506L -Wall dcmdump.cc

c++ -O -D_REENTRANT -D_XOPEN_SOURCE_EXTENDED -D_XOPEN_SOURCE=500 -D_BSD_SOURCE -D_BSD_COMPAT -D_OSF_SOURCE -D_POSIX_C_SOURCE=199506L -Wall -L../libsrc -L../../ofstd/libsrc -o dcmdump dcmdump.o -ldcmdata -lofstd -lz -lm -lrt -lpthread -lnsl

The first line is the line used to compile the dcmdump.cc file, the second line is to link with the libraries. We will first try to compile the dcmdump.cc and if that works, we will link all libraries to it.

9. Go back to your Eclipse-project's properties and navigate to *Properties > C/C++ Build > Settings > GCC C++ Compiler > Preprocessor*. Add the parameters starting with -D from your the first string from output_make.txt (WITHOUT -D, so -D_Test will be _Test). In our example I had to add the following lines:

   HAVE_CONFIG_H

   NDEBUG

   _REENTRANT

   _XOPEN_SOURCE_EXTENDED

   _XOPEN_SOURCE=500

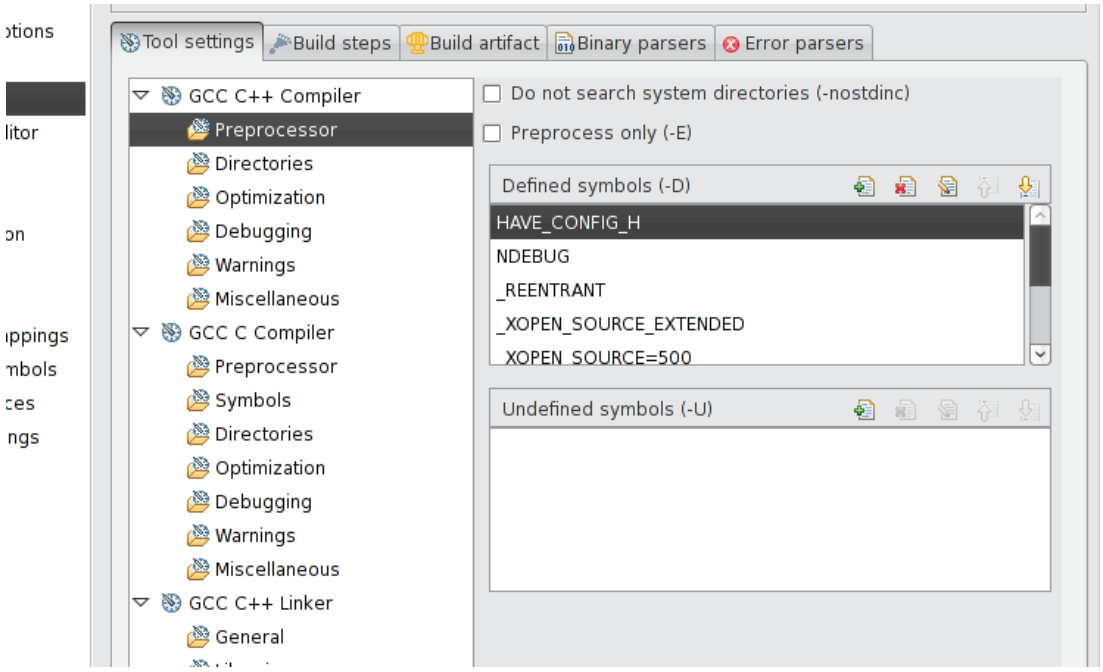
   _BSD_SOURCE

   _BSD_COMPAT

   _OSF_SOURCE

   _POSIX_C_SOURCE=199506L

   Your settings should now look like this:



10. The next thing is to define your include paths. Navigate to *Properties > C/C++ build > Settings > GCC C++ Compiler > Directories*. Click the add button and add the arguments from your output_make.txt file starting with -I (that's a capital i). In our example these are:

.

.

../include

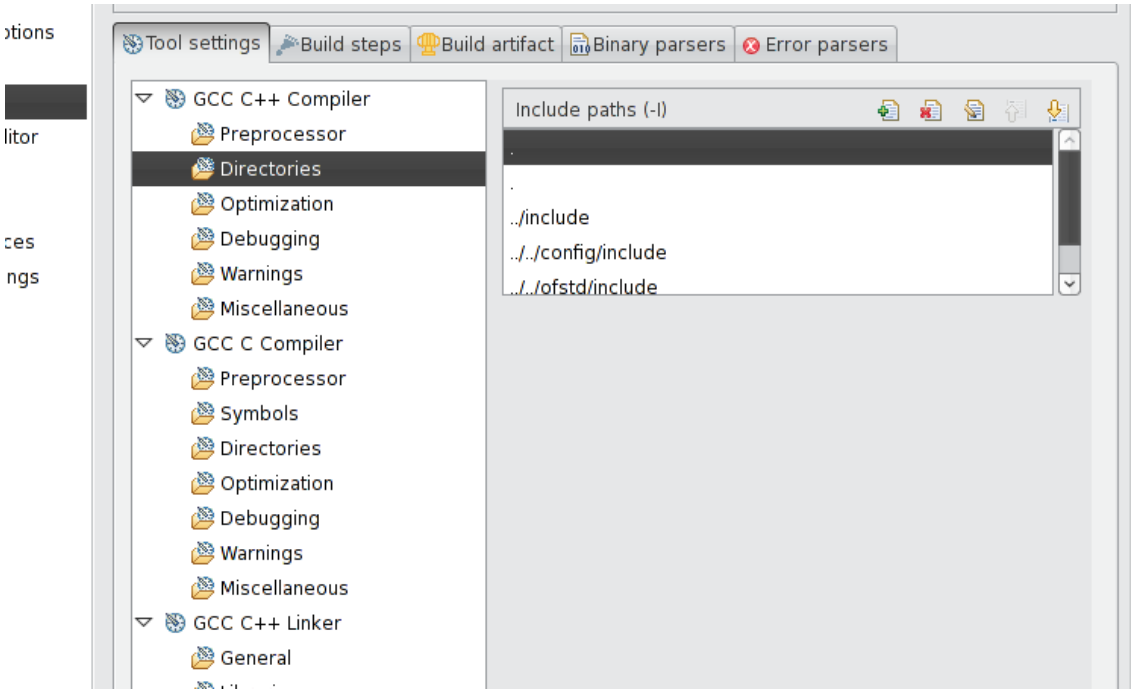
../../config/include

../../ofstd/include

I also needed to include the /usr/local/dicom/include directory, so I added this line as well:

/usr/local/dicom/include

This should look like this:



Why we have to add the dot twice, we don't know, these settings have just been copied from the Makefile of the DCMTK itself.

11. Next is to define your include paths. Navigate to *Properties > C/C++ build > Settings > GCC C++ Compiler > Warnings*. Check the *"All warnings (-Wall)"* check box.

12. Navigate to the *Properties > C/C++ build > Settings > GCC C++ Compiler > Miscellaneous*. And type in the "*Other flags*" box:

    -c

    Your compiler is now ready to use. The only remaining thing to configure before you can start is the linker. For compiling this we need the second line of the *output_make.txt* file.

13. Navigate to the *Properties > C/C++ build > Settings > GCC C++ Linker > Libraries* section. You should add a few libraries in the "*Libraries (-l)*" box, and a few to the "*Libraries search path (-L)*" section. You can find the libraries you should include in the two lines you copied from the output of your make file. In our example:

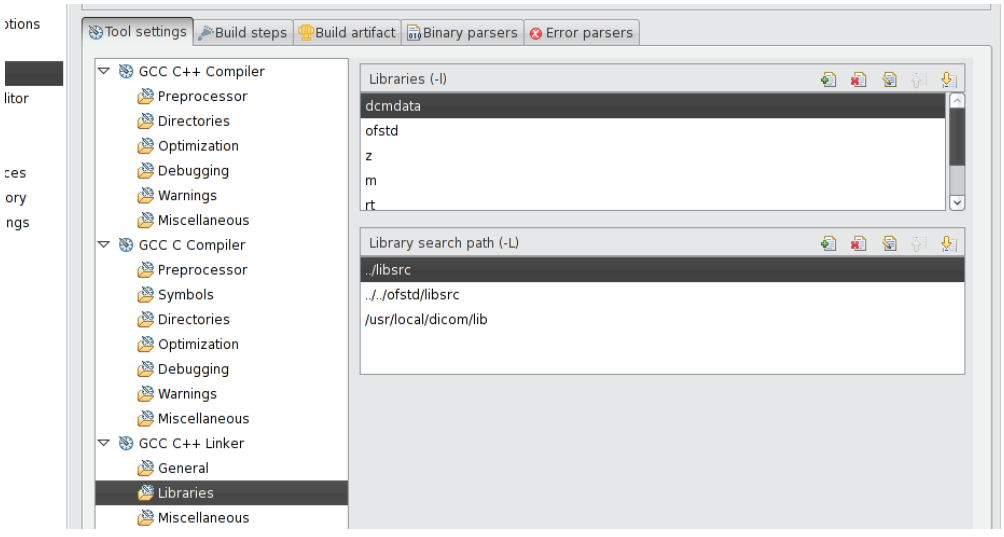
    Libraries (-l)

    dcmdata
    ofstd
    z
    m
    rt
    pthread
    nsl

    Libraries search path (-L)

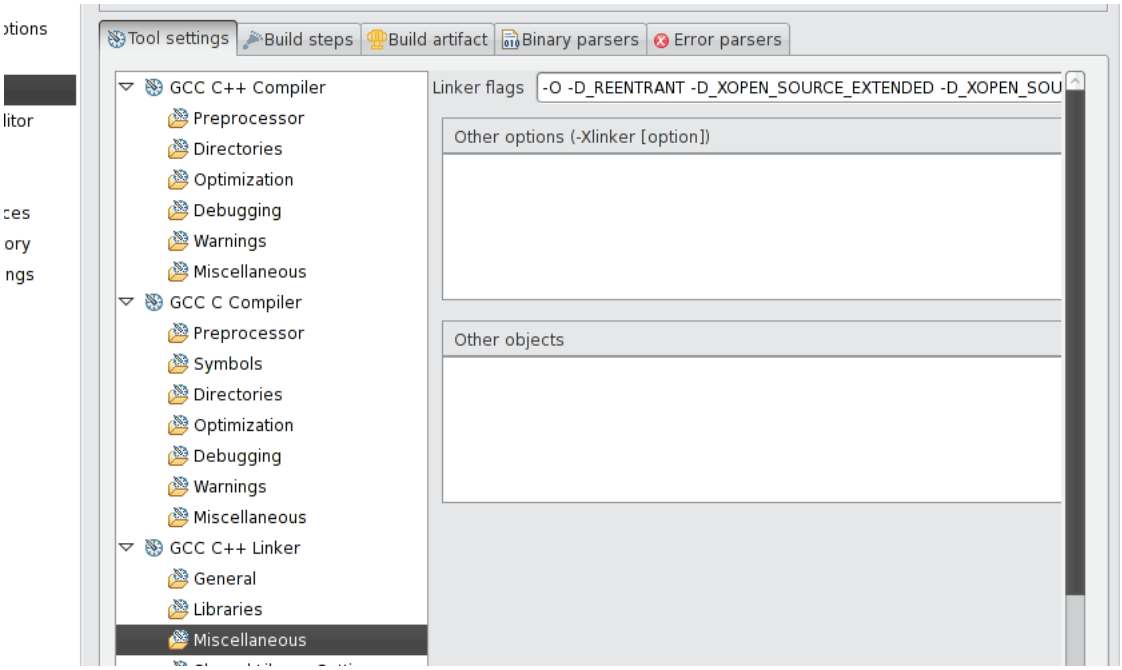
    ../libsrc
    ../../ofstd/libsrc

    And we also needed to include */usr/local/dicom/lib*.
    In our example it looked like this:



14. Next up is the Miscellaneous section. Because Eclipse linker section doesn't support any preprocessing, I added my additional -D parameters and other arguments from the second line of *output_make.txt* to my *GCC C++ Linker > Miscellaneous > Linker flags* section. So my Linker flags box looks like this:

    -O -D_REENTRANT -D_XOPEN_SOURCE_EXTENDED -D_XOPEN_SOURCE=500 -D_BSD_SOURCE -D_BSD_COMPAT -D_OSF_SOURCE -D_POSIX_C_SOURCE=199506L



15. All configuring has now been done, and if it has been done correct, Eclipse and the DCMTK (and dcmdump) should now work like a charm!. The console should output the following:

**** Build of configuration Debug for project dcmtkTest ****

make all
Building file: ../dcmdump.cc
Invoking: GCC C++ Compiler
g++ -DHAVE_CONFIG_H -DNDEBUG -D_REENTRANT -D_XOPEN_SOURCE_EXTENDED
- D_XOPEN_SOURCE=500 -D_BSD_SOURCE -D_BSD_COMPAT -D_OSF_SOURCE
- D_POSIX_C_SOURCE=199506L -I. -I. -I../include -I../../config/include -
I../../ofstd/include -I/usr/local/dicom/include -O0 -g3 -Wall -c -MMD -MP
-MF"dcmdump.d" -MT"dcmdump.d" -o"dcmdump.o" "../dcmdump.cc"
Finished building: ../dcmdump.cc

Building target: dcmtkTest
Invoking: GCC C++ Linker
g++ -L../libsrc -L../../ofstd/libsrc -L/usr/local/dicom/lib -O -D_REENTRANT -
D_XOPEN_SOURCE_EXTENDED -D_XOPEN_SOURCE=500 -D_BSD_SOURCE -
D_BSD_COMPAT -D_OSF_SOURCE -D_POSIX_C_SOURCE=199506L -o"dcmtkTest"
./dcmdump.o - ldcmdata -lofstd -lz -lm -lrt -lpthread -lnsl
Finished building target: dcmtkTest