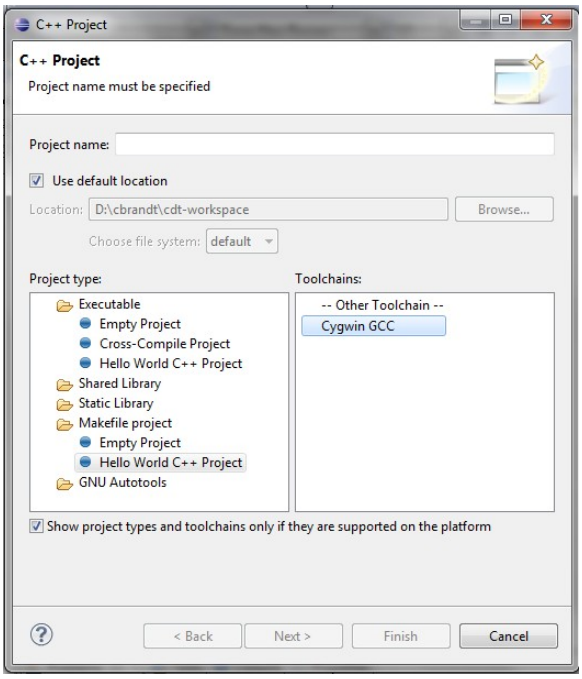
Configuring Eclipse CDT to work with SNAP

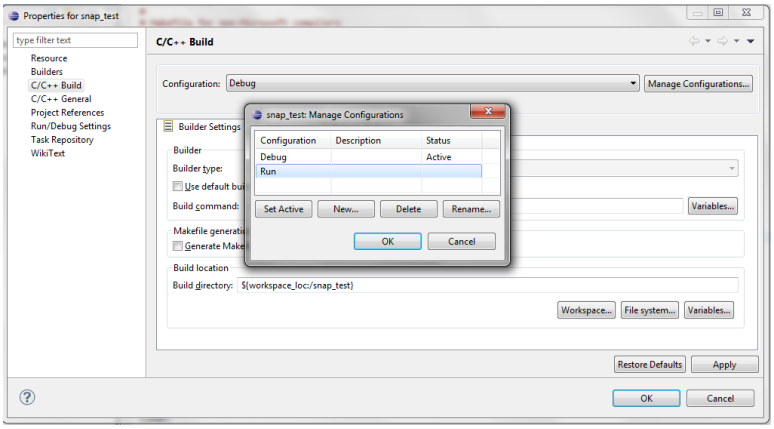
Windows:
First, install cygwin (http://www.cygwin.com/); make sure to get the packages for gdb, gcc, g++, and make.

Installing eclipse CDT: first, as far as I can tell, there is a bug in the 64b-CDT. The output is not always piped to the output. Because of this, it's worth just directly installing the 32b-CDT eclipse.
These instructions work for 32b eclipse indigo cdt.

Create a C++ project (File->new->C++ Project)
Then choose Makefile project, and then either empty or hello world project. Choose the cygwin toolchain and create the project.



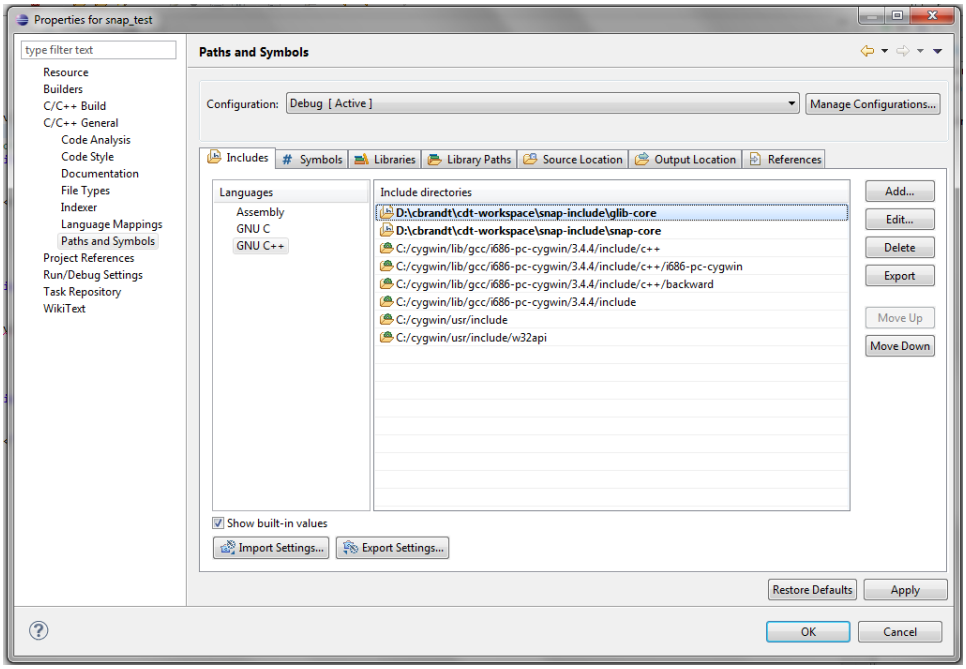
Go to project properties->C/C++ Build and create a new configuration, “Debug.” I also renamed “Default” to “Run”. I also set “Debug” as active. Now change the builder settings for debug. Uncheck “use default build command” and type “make debug” instead.



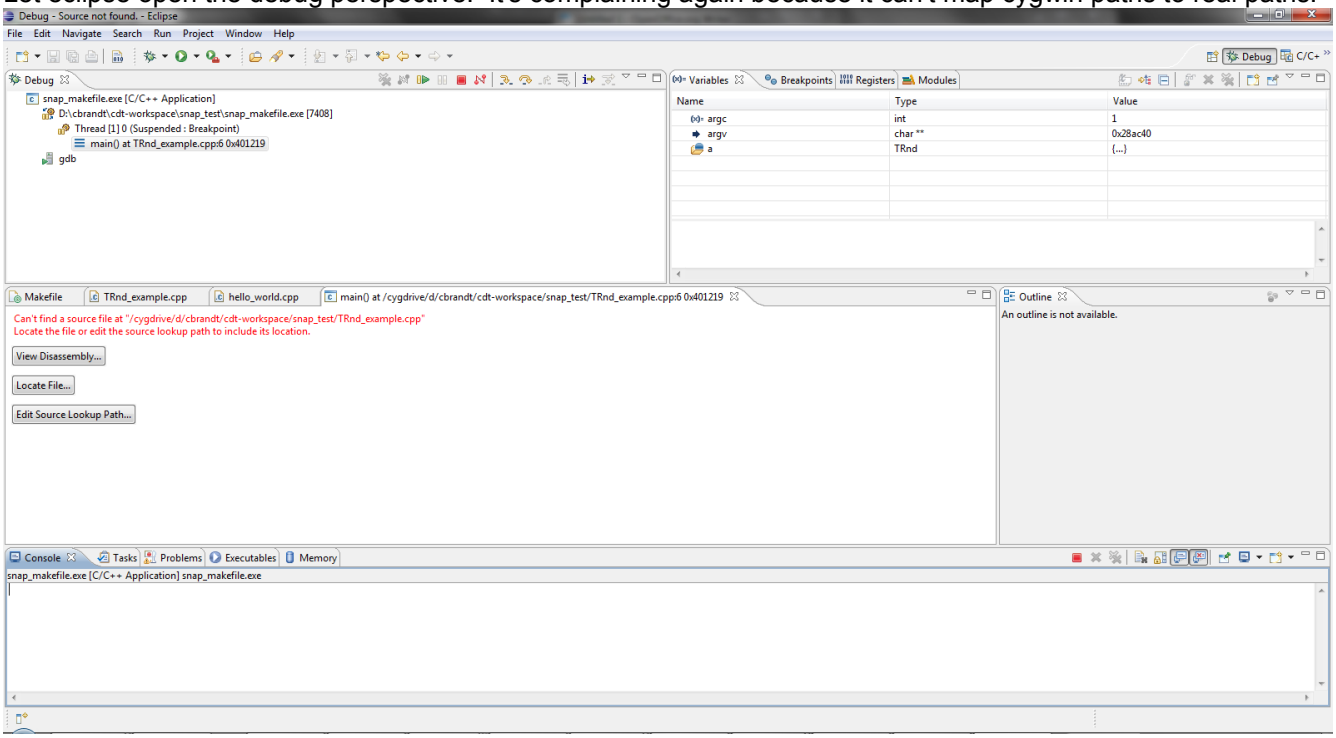
Create your makefile. Mine is a pretty basic modification of the one included in snap; I just added debugging options. I also have two other variables, MAIN and MAIN_OUT. MAIN_OUT is the name of the executable created; for eclipse not to freak out, it must match the name of your project. MAIN is the main file of the project.

```
#
# Makefile for non-Microsoft compilers
# add -rdynamic if you want debugging.
#

## Linux  (uncomment the 2 lines below for compilation on Linux)
#CXXFLAGS += -std=c++98 -Wall
#LDFLAGS += -lrt -fopenmp
SNAPPATH = D:/cbrandt/cdt-workspace/snap-include/
#CBLIBPATH = /u/cbrandt/research/ning/
## CygWin (uncomment the 2 lines below for compilation on CygWin)
CXXFLAGS += -Wall -rdynamic
LDFLAGS += -rdynamic

## Main application file
MAIN = TRnd_example
MAIN_OUT = snap_makefile
##MAIN =  ningbuilder

all: $(MAIN)

opt: CXXFLAGS += -O4
opt: LDFLAGS += -O4
opt: $(MAIN)

debug: CXXFLAGS += -rdynamic -O0 -g -ggdb3 -pg
debug: LDFLAGS += -rdynamic -O0 -g -ggdb3 -pg
debug: $(MAIN)

# COMPILE
$(MAIN): $(MAIN).cpp Snap.o
    g++ $(LDFLAGS) -o $(MAIN_OUT) $(MAIN).cpp Snap.o -I$(SNAPPATH)glib -I$(SNAPPATH)snap

Snap.o:
    g++ -c $(CXXFLAGS) $(SNAPPATH)snap/Snap.cpp -I$(SNAPPATH)glib -I$(SNAPPATH)snap

clean:
    rm -f *.o  $(MAIN)  $(MAIN).exe
    rm -rf Debug Release
```

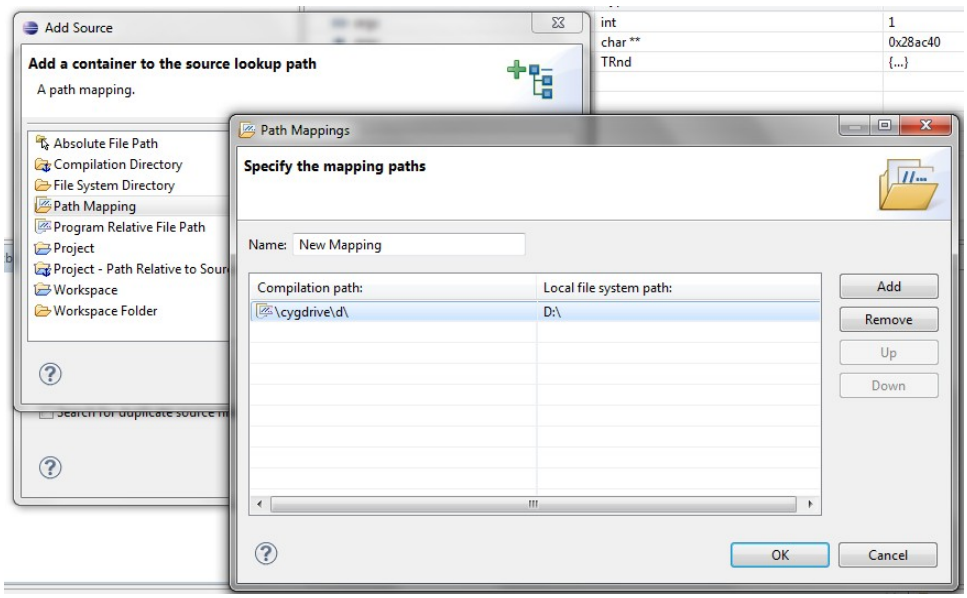
If you take a look at your files, they currently complain that they can't find Snap.
Go to properties->C/C++General->Paths and Symbols.
Under Languages, click on GNU C++ and add the folder locations to snap and glib.
Click “Apply” and let it rebuild the index. All the red squiggles should disappear; if they don't, refresh the project.
If it still doesn't work, try rebuilding the index again.



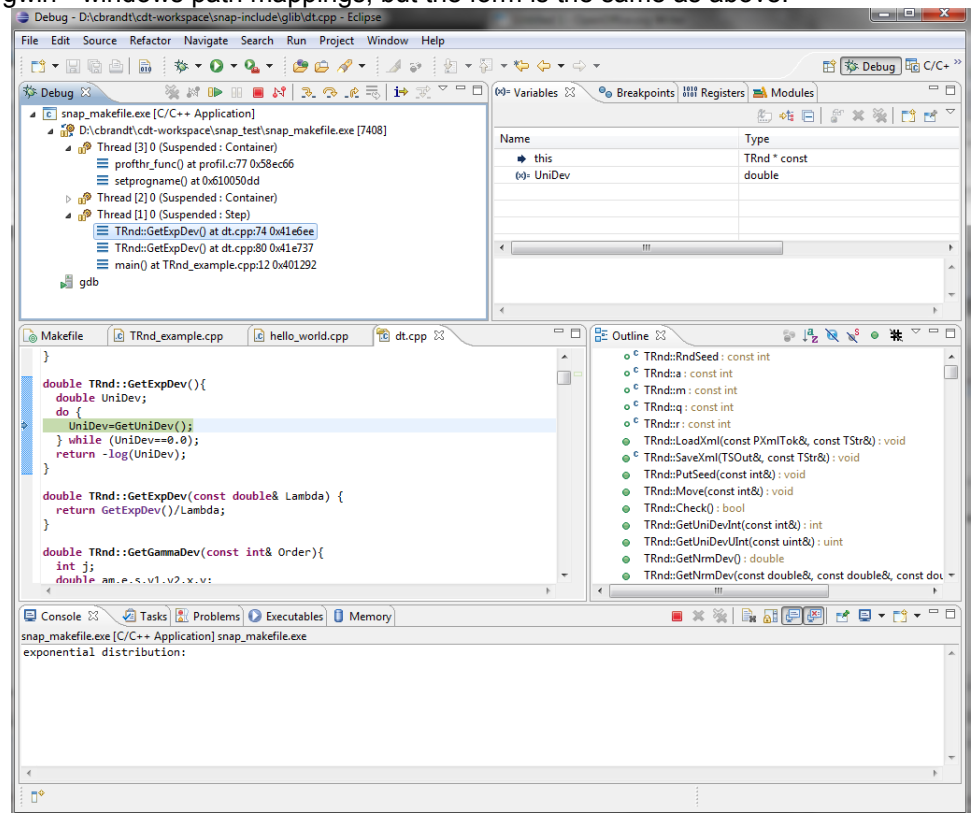
If you try running it at this point, eclipse complains. Instead, first press “CTRL-b” (build). From now on, F11 (debug) will be sufficient to both build and run, but for some reason, you have to build manually once.
Now, put in a breakpoint in your main file (right click on the left margin until a blue dot pops up next to the line) and press F11 (or click “debug”). Choose gdb/mi to launch.
Let eclipse open the debug perspective. It's complaining again because it can't map cygwin paths to real paths.



Click on “edit source lookup path”. Click “add”, and select “Path Mapping.” Add the appropriate mappings from cygdrive to real path names. Mine are shown below.



And now we have the marvelous power of debugging in eclipse! Depending on what you debug, you may need to add more cygwin->windows path mappings, but the form is the same as above.



Linux:
Linux is easier, as one might expect.
Follow the same steps, but choose the appropriate toolchains; there is also no need to map cydrive paths to windows ones.

Note that you need to manage your own makefile; however, the good news is that you then have a makefile to use on the servers.