Notes on Integration BIONAPL-PHREEQC

Ivan Marin

February 20, 2013

Chapter 1

Phreeqc Installation and Compilation

This chapter describes the installation procedures and compilation of the library itself and programs that use the Phreeqc and IPhreeqc library.

1.1 Software Sources and Installation

The two main softwares that are going to be used are Phreeqc and IPhreeqc, developed by the USGS.

1.1.1 Phreeqc

From Phreeqc manual, "The geochemical model PHREEQC is capable of simulating a wide range of equilibrium reactions between water and minerals, ion exchangers, surface complexes, solid solutions, and gases. It also has a general kinetic formulation that allows modeling of non-equilibrium mineral dissolution and precipitation, microbial reactions, decomposition of organic compounds, and other kinetic reactions." Phreeqc can be obtained at [?]

`http://wwwbrr.cr.usgs.gov/projects/GWC_coupled/phreeqc/`

more especifically from

`ftp://brrftp.cr.usgs.gov/pub/charlton/phreeqc/phreeqc-3.0.0-7430.tar.gz`

For Linux, the source distribution was used. The makefile in the src directory was changed so the Intel Compiler can be used. The program was compiled directly using

```
make -j 4
```

and for installation,

```
make install BINDIR=/home/ispmarin/src/laval/phreeqc --prefix=/
   home/ispmarin/src/lib/
```

The self-tests on directory tests ran successfully. The manuals for Phreeqc v.3 can be found at

`ftp://brrftp.cr.usgs.gov/pub/charlton/phreeqc/Phreeqc_3_2013_manual.pdf`

1.1.2 IPhreeqc

IPhreeqc are modules in C, C++ and COM to interface with Phreeqc, making it possible to be called in memory, without having to use files to interface on each iteration. The IPhreecq was downloaded from

`ftp://brrftp.cr.usgs.gov/pub/charlton/iphreeqc/iphreeqc-3.0.0-7430.tar.gz`

The library sources were compiled using

```
./configure CC=icc FC=ifort F77=ifort CXX=icpc --prefix=/home/
   ispmarin/src/lib/
```

and compiled with

```
make -j 4
```

The installer will generate the libraries after a

```
make install
```

The libraries will be put on the directory specified at the configure prefix step.

To compile the example given in the manual (found at `ftp://brrftp.cr.usgs.gov/pub/charlton/iphreeqc/IPhreeqc.pdf`), ifort was used:

```
ifort link.f90 -I/home/ispmarin/src/laval/iphreeqclib/include
-L/home/ispmarin/src/laval/iphreeqclib/ -liphreeqc
```

where both the include and the library must be included. As the library is compiled as shared by default, when running the program the $LD_LIBRARY_PATH$ must be adjusted:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:
/home/ispmarin/src/laval/iphreeqclib
```

Then the binary generated using FORTRAN and linking with the library from C should work. The version of the loaded library should match the version that the program was compiled.

1.1.3 PhreeqPy

The Component Object Model (COM) interface that IPhreeqc uses for Python in Microsoft Windows can't be used in other platforms like MacOSX or Linux. the PhreeqPy project provides a platform-independent interface for Phreeqc for Python. It uses IPhreeqc as an interface between the two. It can be found at `http://www.phreeqpy.com/`.

Phreeqpy can be obtained directly from the link
`http://www.phreeqpy.com/download/phreeqpy-0.2.0.tar.gz`

To be installed, the file phreeqc_dll.py has to be changed manually to point to the compiled shared library of IPhreeqc [1]. Then, a simple sudo python setup.py install installs the module to the python distribution:

```
sudo python setup.py install
```

To test to see if the module was correctly installed, open a python console and type

```
import phreeqpy
```

with no errors. The version 0.2 of Phreeqpy now is included in PyPI and can be installed issuing

```
pip install -U phreeqpy
```

32 versus 64 bits architecture

The Iphreeqc library, if compiled in a system with 64 bits architecture, will generate a library for 64 bits. The library that Phreeqpy includes is 32 bits. The architecture of the library can be checked using the Linux command file:

```
ispmarin@gruman:~/src/pywork/fddarcy/src$ file /usr/local/lib/python2
.7/dist-packages/PhreeqPy-0.1.0-py2.7.egg/phreeqpy/iphreeqc/
libiphreeqc.so.0.0.0
```

[1] In the latest linux installation, it seems that the iphreeqc library was already included in the zipped package

```
/usr/local/lib/python2.7/dist-packages/PhreeqPy-0.1.0-py2.7.egg/
    phreeqpy/iphreeqc/libiphreeqc.so.0.0.0: ELF 32-bit LSB shared
    object, Intel 80386, version 1 (SYSV), dynamically linked, not
    stripped
```

for a 32 bits compiled library, and

```
ispmarin@gruman:~/src/pywork/fddarcy/src$ file /home/ispmarin/src/lib
    /iphreeqc-2.18.4-6386/src/.libs/libiphreeqc.so.0.0.0
/home/ispmarin/src/lib/iphreeqc-2.18.4-6386/src/.libs/libiphreeqc.so
    .0.0.0: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV),
    dynamically linked, BuildID[sha1]=0
    x09c8882a8e00ab2dc7a894b26d82878d77623eca, not stripped
```

for a 64 bits. In Python, when loading the library, the architecture of the library must match the architecture of the Python version. In this case, the file iphreeqc_dll.py (located in /usr/local/lib/python2.7/dist-packages/PhreeqPy-0.1.0-py2.7.egg/phreeqpy/iphreeqc) must be changed to match the right architecture. The same instructions are valid for different versions of Phreeqc and IPhreeqc.

1.2 IPhreeqc Interface

The idea of the library IPhreeqc is to interface a user program with Phreeqc. IPhreeqc does it using as input a string formatted as Phreeqc input file. It passes in memory this input file to Phreeqc, runs it, and possibilitates to retrieve the values calculated to the host program.

The IPhreeqc interface, called directly from IPhreeqc library or outside via PhreeqPy, for example, has to be formatted when doing output. Before getting any data from phreeqc.get_selected_output_array(), the method SELECTED_OUTPUT has to be called. RUN_CELLS has also to be called before any action on the phreeqc components, as it sets the environment up and reacts what is in the input file.

The SELECTED_OUTPUT section has also be customized to the process at hand. What is needed is one input file for Phreeqc detailing all the processes in the Calcite geochemistry path, and with that input it on the finite difference scheme. So I need to put it to work a Phreeqc input file!

1.3 Development on Windows Platform

The Windows Platform is incredibly resilient to not permit tools that are not from Microsoft to be used. Fortran is a separate case: a Fortran compiler was never developed by Microsoft, so support is abysmal. The use a combination of the Eclipse CDT/Photran, is presently not possible. Intel support for Visual Studio Fortran is also very strange, and Visual Studio 2008 does not integrate well with Intel Fortran Compiler v 11.03.

After a lot of perusal a Fortran code was compiled using the precompiled versions of Phreeqc and IPhreeqc avaiable on the USGS website (**??**). The compiler used was the Intel Visual Fortran Compiler v. 11.1.054 and Visual Studio 2008 SP1. Several steps had to be changed in the Visual Studio so it can find the libraries.

Phreeqc and IPhreeqc are installed using the Microsoft Windows installers available from `ftp://brrftp.cr.usgs.gov/pub/charlton/phreeqc/phreeqc-3.0.0-7430.msi` and `ftp://brrftp.cr.usgs.gov/pub/charlton/iphreeqc/IPhreeqc-3.0.0-7430-vs2005-x64.zip`

Checking that the Intel Visual Fortran Compiler works with Visual Studio 2008 (they also work for Visual Studio 2005), these steps should be followed:

- Create the project as a Console application.
- Click on Project -¿ Properties -¿ Configuration Management
- Change the platform to "x64", as the compiled libraries from USGS are in this architecture. If the platform x64 does not exist, create it copying the configurations from win32.
- On the left side, click on Fortran, add to "Additional Include Directories" all three IPhreeqc directories: libx64, dllx64 and include.
- On the left side, click on Linker, add to "Additional Include Directories" all three IPhreeqc directories: libx64, dllx64 and include.
- Click on Linker, Input, and add to "Additional Dependencies" IPhreeqc.lib

It should compile now.

1.3.1 DLLs in the Path

Having the DLLS IPhreeqc.dll and IPhreeqcd.dll on the path is not enough for the program to run. The workaround was to copy those libraries to the folder System32 in the Windows folder (as, for example, C:/Windows/System32). Now the application runs.

1.4 Development on Linux Platform

The IDE for development in the Linux Platform is the Eclipse CDT with Photran plugin and the Intel Fortran compiler. The Eclipse IDE must be configured to correctly find both the Intel compiler and the libphreeqc library. It should be noted that the use of Eclipse or any other IDE in Linux is not necessary, and in some cases is not recommended. In any case, the following instructions present how to use the Eclipse IDE with the Intel C/C++ and Fortran compilers. A bash script should be created in the Eclipse installation directory with the contents

```
#!/bin/bash
#

LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/ispmarin/src/lib/iphreeqclib
export LD_LIBRARY_PATH

source /opt/intel/bin/compilervars.sh intel64
source /opt/intel/mkl/bin/mklvars.sh intel64
/home/ispmarin/bin/eclipse/eclipse
```

Noting that the directories are specific to the installation and should be changed to reflect the local installation directories.

A Photran with Makefile should be created using the instructions from the Photran help (`http://help.eclipse.org/juno/index.jsp?topic=%2Forg.eclipse.photran.doc.user%2Fhtml%2Ftoc.html`) that consists of the following steps:

```
1. Click File > New > Fortran Project
2. Call it HelloFortran
3. Expand "Makefile project" in the project type list (it has a
   folder icon), and choose "Empty Project"
4. Select "-- Other Toolchain --" in the toolchain list in the right-
   hand column, and click Next
```

```
5. Click on Advanced Settings
6. Expand C/C++ Build in the list on the left, and click on Settings
7. Click on the Binary Parsers tab.  Check the appropriate parsers
   for your platform. If you are using Windows, check PE Windows
8. Parser and/or Cygwin PE Parser; if you are using Linux, check Elf
   Parser; if you are using Mac, check Mach-O parser.
9. Click on the Error Parsers tab. Check the error parser(s) for the
   Fortran compiler(s) you will use.
10. Click OK
11. Click Finish
12. Click File > New > Source File
13. Call it hello.f90
14. Click Finish
15. Type the standard "Hello, World" program shown below.
16. Click File > New > File
17. Call it Makefile
18. Click Finish
19. Create a Makefile similar to the one shown below. Again, we
   assume you are familiar with the structure of a Makefile. You
   cannot simply copy-and-paste this example because the gfortran and
    rm lines must start with a tab, not spaces. The -g switch
   instructs gfortran to include debugging symbols in the generated
   executable so that it can be debugged later. The -o switch tells
   it what to name the generated executable.
20. Click Project > Clean, then click OK
21. Open the Console view, and make sure "make" ran OK and compiled
   your program
22. In the Fortran Projects view, expand the Binaries entry, and
   click on your executable (e.g., "hello.exe - x86le")
23. Click Run > Run As > Local Fortran Application
24. Choose GDB Debugger (Cygwin GDB Debugger if you're under Windows)
25. Check the Console view, and make sure "Hello World" appeared.
```

After these steps the Eclipse IDE should be able to compile correctly using the Makefile. Note that the project compiles using the provided Makefile directly from the console, without setting up the Eclipse Photran project, differently from the Visual Studio solution.

An example of the Makefile presently used to compile the phreeqc-fortran code with the bio-geochem module is as follows:

```
#
   ============================================================================

# Name        : Makefile
# Author      : Ivan Marin
# Version     :
# Copyright   : Proprietary
# Description : Makefile for phreeqc-fortran
#
   ============================================================================
```

```
.PHONY: all clean

# Change this line if you are using a different Fortran compiler
IFORT = ifort
GFORTRAN = gfortran
#FORTRAN_COMPILER = gfortran

COMPILER_OPTIONS_GFORTRAN = -g -fbounds-check -g -Wall -
   Wuninitialized -O -Wtabs -fbacktrace
COMPILER_OPTIONS_IFORT = -g #-check all -fpe0 -warn -traceback -debug
    extended


LIBRARY_PATH_2=/home/ispmarin/src/lib/iphreeqc-2.18.4-6386/lib/
INCLUDE_PATH_2=/home/ispmarin/src/lib/iphreeqc-2.18.4-6386/include/

LIBRARY_PATH_3=/home/ispmarin/src/lib/iphreeqc-3.0.0-7430/lib
INCLUDE_PATH_3=/home/ispmarin/src/lib/iphreeqc-3.0.0-7430/include/

COMP_DIR = ../Debug
SOURCE = bionapl-fpp.f
SOURCE_MOD = biogeochem.f90
BINARY = bionapl-fpp
OBJECTS = biogeochem.o bionapl-fpp.o

ifeq ($(FC),ifort)
       COMPILER=$(IFORT)
           COMPILER_OPTIONS=$(COMPILER_OPTIONS_IFORT)
     else
           COMPILER=$(GFORTRAN)
       COMPILER_OPTIONS=$(COMPILER_OPTIONS_GFORTRAN)
     endif


all: $(OBJECTS)
        $(COMPILER) $(COMPILER_OPTIONS_IFORT) $(OBJECTS) -o $(
           COMP_DIR)/$(BINARY) -I$(INCLUDE_PATH_3) -L$(LIBRARY_PATH_3
           ) -liphreeqc

biogeochem.o biogeochem.mod: $(SOURCE_MOD)
        $(COMPILER) -c $(COMPILER_OPTIONS) $(SOURCE_MOD) -I$(
           INCLUDE_PATH_3) -L$(LIBRARY_PATH_3) -liphreeqc

bionapl-fpp.o: $(SOURCE)
        $(COMPILER) -c $(COMPILER_OPTIONS) $(SOURCE) -I$(
           INCLUDE_PATH_3) -L$(LIBRARY_PATH_3) -liphreeqc


clean:
        rm -f *.mod *.o
```