IBM XL Fortran and IBM XL C/C++ for AIX

Interoperability of IBM XL Fortran, IBM XL C/C++ and Java

By: David Forster

Level: Introductory

May 2012

Contents

Before you start

About this Tutorial

This tutorial demonstrates how to use the interoperability features of the IBM XL Fortran and the IBM XL C/C++ compiler to integrate Fortran, C and Java source code into an application. The demonstration shows how to call Java source routines from Fortran and vice versa.

Interoperability is useful for integrating parts written in different languages into an application. This enables developers to take advantage of language best situated for the job or easily integrate legacy assets into a new application.

Objectives

- Use Fortran, C/C++ and Java source code parts to build a program, to demonstrate interoperability features available with IBM XL Fortran and IBM XL C/C++ compilers.
- Total time: 30 minutes

Prerequisites

- Basic Unix skills
- Basic Source code compile and build experience

System Requirements

IBM XL Fortran Compiler
http://www.ibm.com/software/awdtools/fortran/xlfortran/aix/sysreq

IBM XL C/C++ Compiler
http://www.ibm.com/software/awdtools/xlcpp/aix/sysreq/

Glossary

IBM XL Fortran Compiler: The IBM® XL Fortran compiler offers advanced compiler and optimization technologies and is built on a common code base for easier porting of your applications between platforms. It complies with the latest Fortran international standards and industry specifications and supports a large array of common language features.

IBM XL C/C++ Compiler: IBM® XL C and C++ compilers offer advanced compiler and optimization technologies and are built on a common code base for easier porting of your applications between platforms. They comply with the latest C/C++ international standards and industry specifications and support a large array of common language features.

Getting Started

Start the Terminal Emulator to AIX System

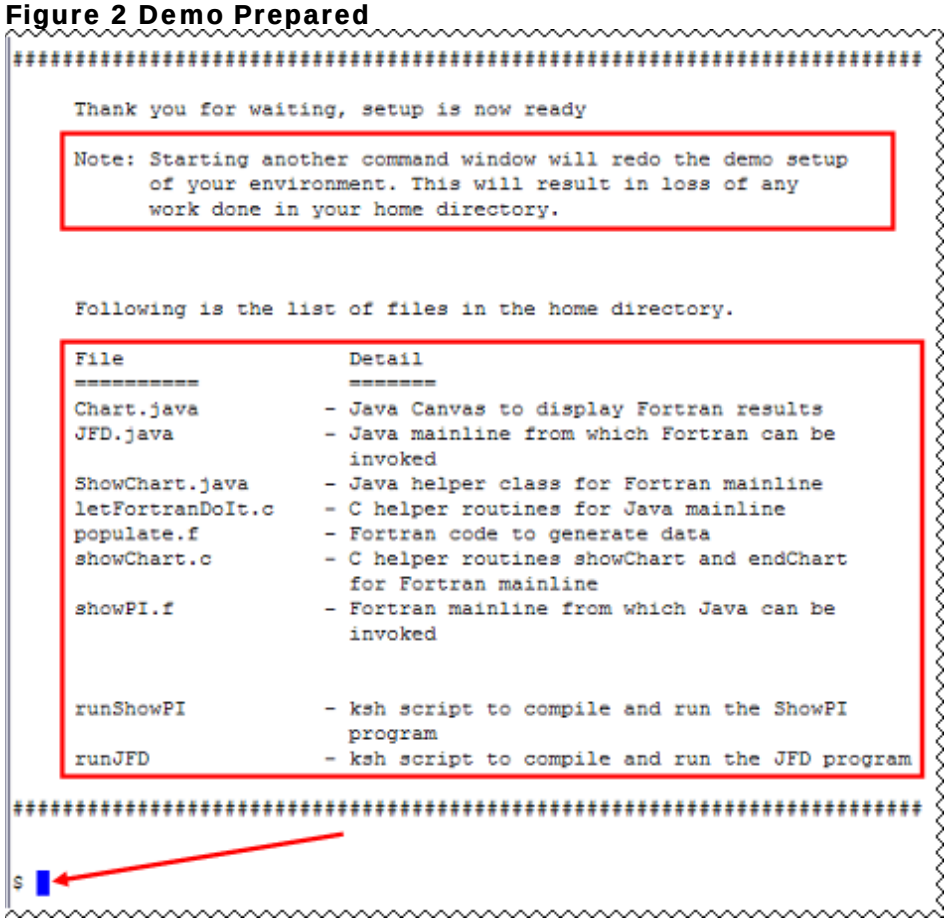
Figure 1 Get Started

Double click the “Launch AIX” icon on the desktop (See Figure 1) to start the character terminal to AIX system.

Get Started with IBM XL Fortran and IBM XL C/C++ Compiler

A successful login will result with the user presented with a menu of the demos hosted on the server. Type 4 and press Enter to select the “XL Fortran and XL C/C++ with Java” demo.

Figure 2 Demo Prepared



Note: Starting another command window will start the demonstration setup of your environment. This will result in loss of any work done in your home directory. This will impact any progress you have made on demo steps going forward.

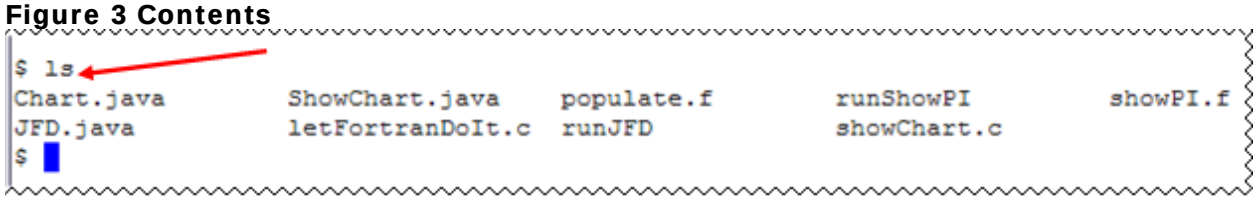
This demo does not require more than one terminal window. However, if you prefer more than one terminal window then you may open them before going forward.

On the terminal window you will see important information and the list of files available in your home directory. The terminal window is now ready for commands (See Figure 2 Demo Prepared). Your home directory contains the necessary source code and your environment has been set for this demonstration. Type the ls command to see the directory content (See Figure 3 Contents).

Command:

```
ls
```

Figure 3 Contents



Suite of programs, library routines and classes to allow Fortran to communicate with C and with Java, and vice-versa.

The list of files in your home directory:

Chart.java	Java Canvas to display Fortran results
JFD.java	Java mainline from which Fortran can be invoked
ShowChart.java	Java helper class for Fortran mainline
letFortranDoIt.c	C helper routines for Java mainline
populate.f	Fortran code to generate data
showChart.c	C helper routines showChart and endChart for Fortran mainline
showPI.f	Fortran mainline from which Java can be invoked

Helper scripts:

We recommend that you use the detail commands given in the steps in each section to compile and run the programs. However, if you are finding it difficult to type long commands then you may use the following scripts to compile and execute the programs directly and skip over to highlight topic of the section.

runShowPI	ksh script to compile and run the ShowPI program in Invoking Java from Fortran section
runJFD	ksh script to compile and run the JFD program in Invoking Fortran from Java section

Invoking Java from Fortran

Steps:

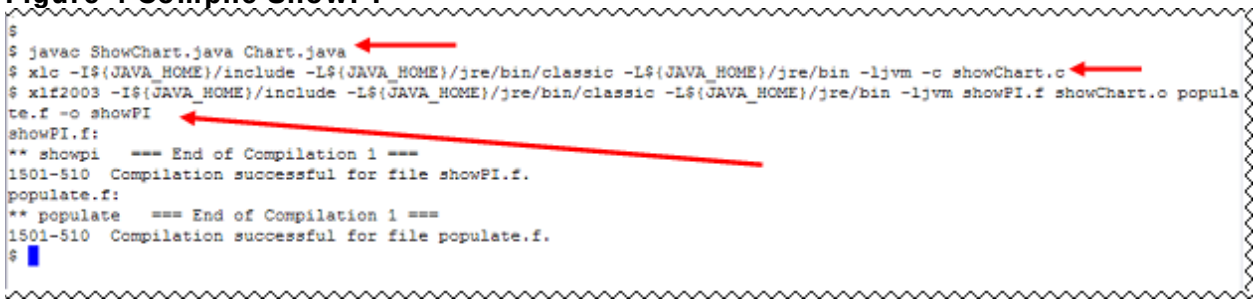
1. Compile the source files. See figure 4 Compile ShowPI

   Note: JAVA_HOME is a environment variable that is already set in your environment. You may run the command echo $JAVA_HOME to see its contents.

   Command:

```
javac ShowChart.java Chart.java

xlc -I${JAVA_HOME}/include -L${JAVA_HOME}/jre/bin/classic \
-L${JAVA_HOME}/jre/bin -ljvm -c showChart.c

xlf2003 -I${JAVA_HOME}/include -L${JAVA_HOME}/jre/bin/classic \
-L${JAVA_HOME}/jre/bin -ljvm showPI.f showChart.o populate.f -o showPI
```

Figure 4 Compile ShowPI



   What you did with the commands above:

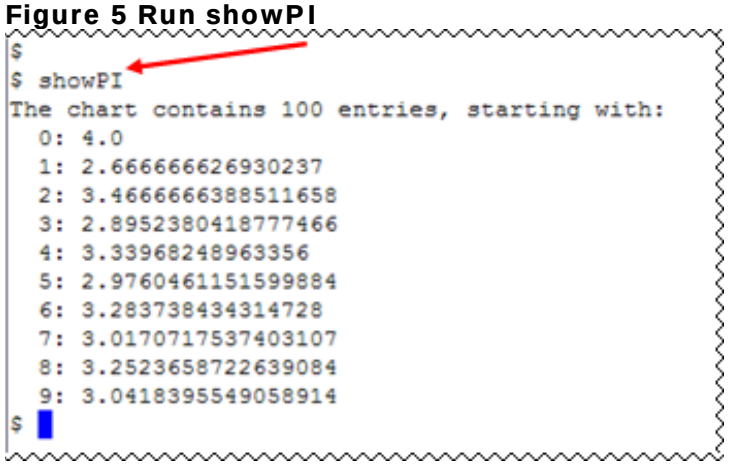
   - Compiled two Java source files ShowChart.java and Chart.java into class files
   - Compiled one C source file, showChart.c, with Java interface built in, to an object file, showChart.o
   - Compiled two Fortran source files, showPI.f and populate.f, linked in object file showChart.o (produced in the C compile step), and created the showPI executable program

2. Run showPI program (See Figure 5 Run showPI).

   Note: The Java accessed from showPI executable relies on the environment variables LIBPATH and LDR_CNTRL. You may see the contents of the environment variable using echo $LIBPATH and echo $LDR_CNTRL command respectively on the command prompt.

Command:
    showPI

Figure 5 Run showPI



Highlights:

The program files required are showPI.f, showChart.c, populate.f, ShowChart.java and Chart.java: The "showPI" Fortran program invokes the "populate" Fortran subroutine to generate data, and then invokes the C functions "showChart" and "endChart".

“showChart" arranges to load the Java VM, the "ShowChart" class, and run a static method of that class ("show"), while "endChart" arranges to terminate the Java VM. If a windowing system (say, X11) is available from the AIX terminal window, you will see a chart displayed (using the "Chart" class); failing that, the program prints out the values from within Java.

The call in showPI (Fortran program) to showChart (C function) looks like a normal Fortran subroutine call:

```
…
…
call showChart(NPOINTS, p, 2.0d0, 4.0d0)
…
…
```

See line 92 of showPI.f Fortran program
    Hint Command:
        more –p 92j showPI.f

The difference is in the interface declaration, which the compiler uses to determine how it should pass the data to the C routine:

```
…
…
     subroutine showChart(n, rarr, ymin, ymax) bind(C,name="showChart")
       use, intrinsic :: ISO_C_BINDING
       integer(C_INT), value :: n
       real(C_DOUBLE) :: rarr(0:n-1)
       real(C_DOUBLE), value :: ymin, ymax
     end subroutine showChart
…
…
```

See line 73 of showPI.f Fortran program
    Hint Command:
          more –p 73j showPI.f

Note that all arguments rely on KIND parameters from the Fortran ISO_C_BINDING module, and that all but one argument is passed by value.

The thing to take away here is that Fortran calls to C (and vice-versa) are made simple by the use of Fortran and C interoperability.

In the C program showChart, the call to Java is more involved, as it requires the starting of the Java Virtual Machine, the loading of the ShowChart class and the conversion of C arguments to Java types before invoking ShowChart's static method "show".

See line 147 of showChart.c for details of invoking Java.
    Hint Command:
          more –p 147j showChart.c

You may also use more command to look at programming details of ShowChart.java and Chart.java programs

Invoking Fortran from Java

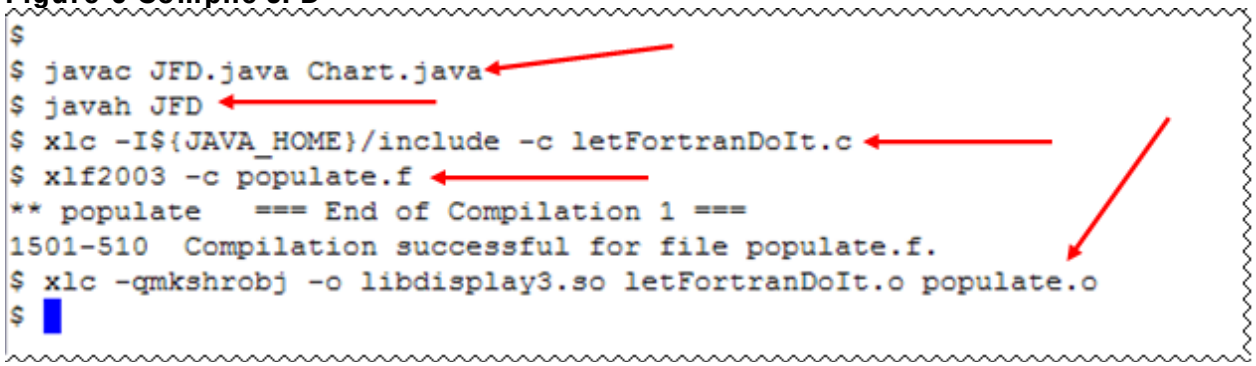
Steps:

1. Compile the source files. See figure 6 Compile JFD

   Command:

```
javac JFD.java Chart.java

javah JFD

xlc -I${JAVA_HOME}/include -c letFortranDoIt.c

xlf2003 -c populate.f

xlc -qmkshrobj -o libdisplay3.so letFortranDoIt.o populate.o
```

Figure 6 Compile JFD



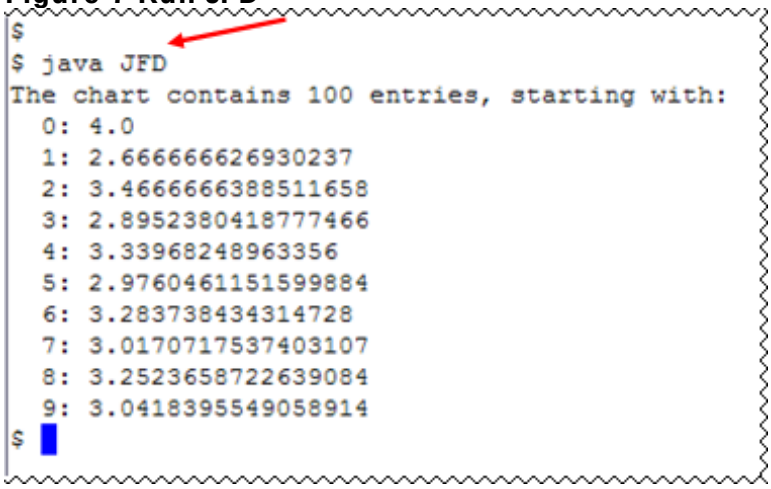
What you did with the commands above:

- Compiled two Java source files ShowChart.java and Chart.java into class files using the javac command
- Generated a C header file for the JFD class using the javah command. This C header file is used in letFortranDoIt.c
- Compiled letFortranDoIt.c to an object file letFortranDoIt.o using the xlc C compiler command
- Compiled the Fortran source file populate.f into an object file populate.o using the xlf2003 Fortran compiler command
- Produced a shared library, libdisplay3.so, containing letFortranDoIt.o and populate.o object files using xlc C compiler command

2. Run the Java interpreter on the JFD class. See Figure 7 Run JFD

Command:
```
java JFD
```

Figure 7 Run JFD



Note: The program relies on the environment variable LIBPATH to find libdisplay3.so. You may see the contents of the environment variable using echo `$LIBPATH` on the command prompt.

Highlights:

This demo demonstrates the invocation of Fortran procedures from Java.

The class JFD contains a main method which calls the Fortran routine "populate" via the C function "letFortranDoIt". In the JFD class, this is declared as a native method. This native method or C routine shows some of the same techniques as we saw with "showChart", i.e., copying and transforming Java data into a form useable by C. The actual invocation of the Fortran routine is a single line:

```
…
…
populate(len,rarr);
…
…
```

See line 59 of letFortranDoIt.c
Hint Command:
```
more –p 59j letForranDoIt.c
```

This simple call from C to Fortran is made possible by declarations used in the Fortran subroutine.

```
…
…
   subroutine populate(n, rarr) bind(C)
     use, intrinsic :: ISO_C_BINDING   ! The module containing
                                       ! definitions for C_INT, etc.
     integer(C_INT), value :: n        ! int call-by-value argument
     real(C_DOUBLE) :: rarr(0:n-1)     ! double call-by-reference array
                                       ! argument
     real(C_DOUBLE) :: pi              ! work variable, double (same KIND
                                       ! as "rarr")
…
…
```

See line 50 of populate.f
    Hint Command:
        Man –p 50j populate.f

Other than these declarations, a programmer need not make any accommodations in Fortran to be used by a C or a Java program.

The thing to take away here is that there is one small difference in the running of this program. It relies on the compiled Fortran and C code being stored in a library. After the compilers generate the object code for the Fortran and C routines, they are copied to the library, which is ultimately loaded by the JFD class.

What you have learned

In this exercise you learned how to:

- How to use the flexibility provide by IBM XL Fortran and IBM XL C/C++ compiler to mix languages.
- How to call Java from Fortran using C.
- How to call C and Fortran routines from Java.
- How to generate shared libraries with mixed routines from Fortran and C.

Conclusion

This concludes the tutorial on interoperability using IBM XL Fortran, IBM XL C/C++ and Java. This tutorial has shown how to compile programs with the IBM XL Fortran and IBM XL C/C++ compilers on AIX and use the call the routines to and from Java.

Trademarks

Resources

XL C/C++ for AIX, Library :

http://www.ibm.com/software/awdtools/xlcpp/aix/library/

XL Fortran for AIX, Library :

http://www.ibm.com/software/awdtools/fortran/xlfortran/aix/library/

Community Cafe :

http://www.ibm.com/software/rational/cafe/community/ccpp