Integrated Java™ and C Language Debugging using the Eclipse platform

Matthew B White

Staff Software Engineer
IBM UK
http://www.ibm.com

TS-1011

Goal of this Talk

Learn how to debug hybrid C and Java™ technology-based applications using the Eclipse Platform at the same time!

Agenda

Introduction

Building Native Code in Eclipse

How to Do Integrated Debugging With the Demo

Alternatives to Java Native Interface (JNI)

JNI—Some Best Practices

Agenda

Introduction

Building Native Code in Eclipse

How to Do Integrated Debugging With the Demo

Alternatives to Java Native Interface (JNI)

JNI—Some Best Practices

Java Technology from a C Perspective

- How does the operating system see the Java VM?
- A Java VM is just another process
- This process can be controlled by a debugger
- Java programming environment source level debugging
  - “Application level” operation from the OS position
- This assumption is critical to this hybrid approach

The Problem…

- Java technology is good
- … but sometimes we need to use native code
- When a problem occurs how do we use the normal debugging techniques to solve it?
- Normal C and Java based debuggers don't know about the "other world"
- Some proprietary debuggers have solved this
- How do we do this with other tools?

State of the Art

- Using GDB and JDB it was possible to do this
- Eclipse with the JDT and CDT have arrived
- It is possible to move the techniques to this space
- A similar approach using other GUI debuggers is possible

Agenda

Introduction

Building Native Code in Eclipse

How to Do Integrated Debugging With the Demo

Alternatives to Java Native Interface (JNI)

JNI—Some Best Practices

Setup—and Building JNI Code in Eclipse

- Tools used
  - Eclipse 3.1 and CDT
  - Cygwin
  - MinGW
  - JDK™ 1.4.2
- Notable points
  - Use correct linking options in makefile
  - Set the binary parser to the correct value
  - Set the makefile command

Sample Compile Command for a DLL

```
gcc  -shared -g -o JavaImp.dll jnitest.c
     -Wl,--kill-at -Id:\mingaw\include
     -Id:\_jdks\ibm_jdk1.4.2\include
     -Id:\_jdks\ibm_jdk1.4.2\include\win32
```

- -kill-at is important to ensure that the __stdcall naming convention is used

DEMO

Building JNI Code in Eclipse

Agenda

Introduction

Building Native Code in Eclipse

How to Do Integrated Debugging With the Demo

Alternatives to Java Native Interface (JNI)

JNI—Some Best Practices

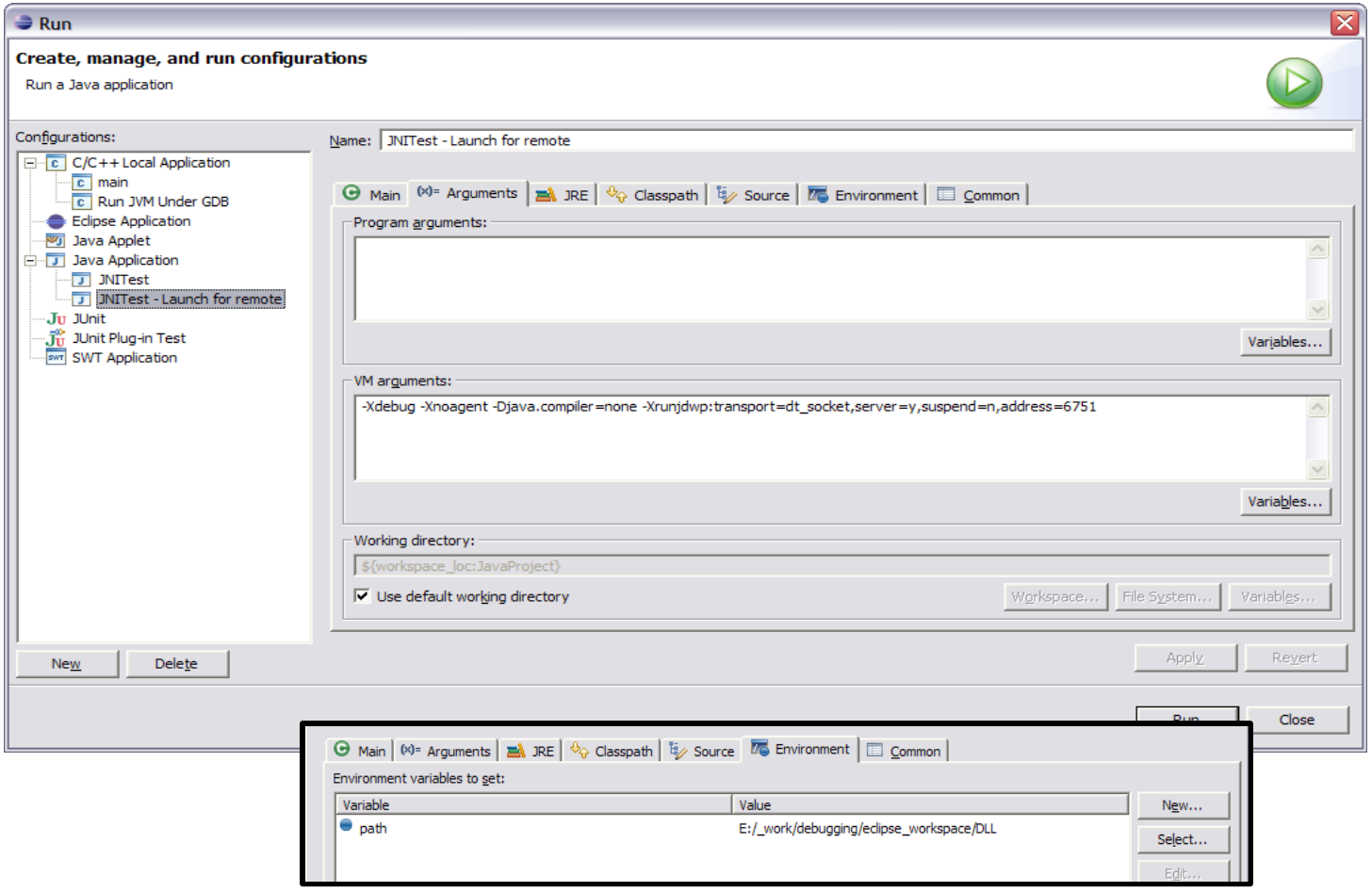
Integrated Debugging

- The general approach is to
  - Launch the debugee Java VM process
  - Attach to this debugee Java VM process from GDB
  - Attach to the Java VM using the Java based debugger
- This can all be done from within the Eclipse debug perspective
- Not a totally integrated solution
  - 3 process are being launched
  - Eclipse perspective helps by showing output in same place

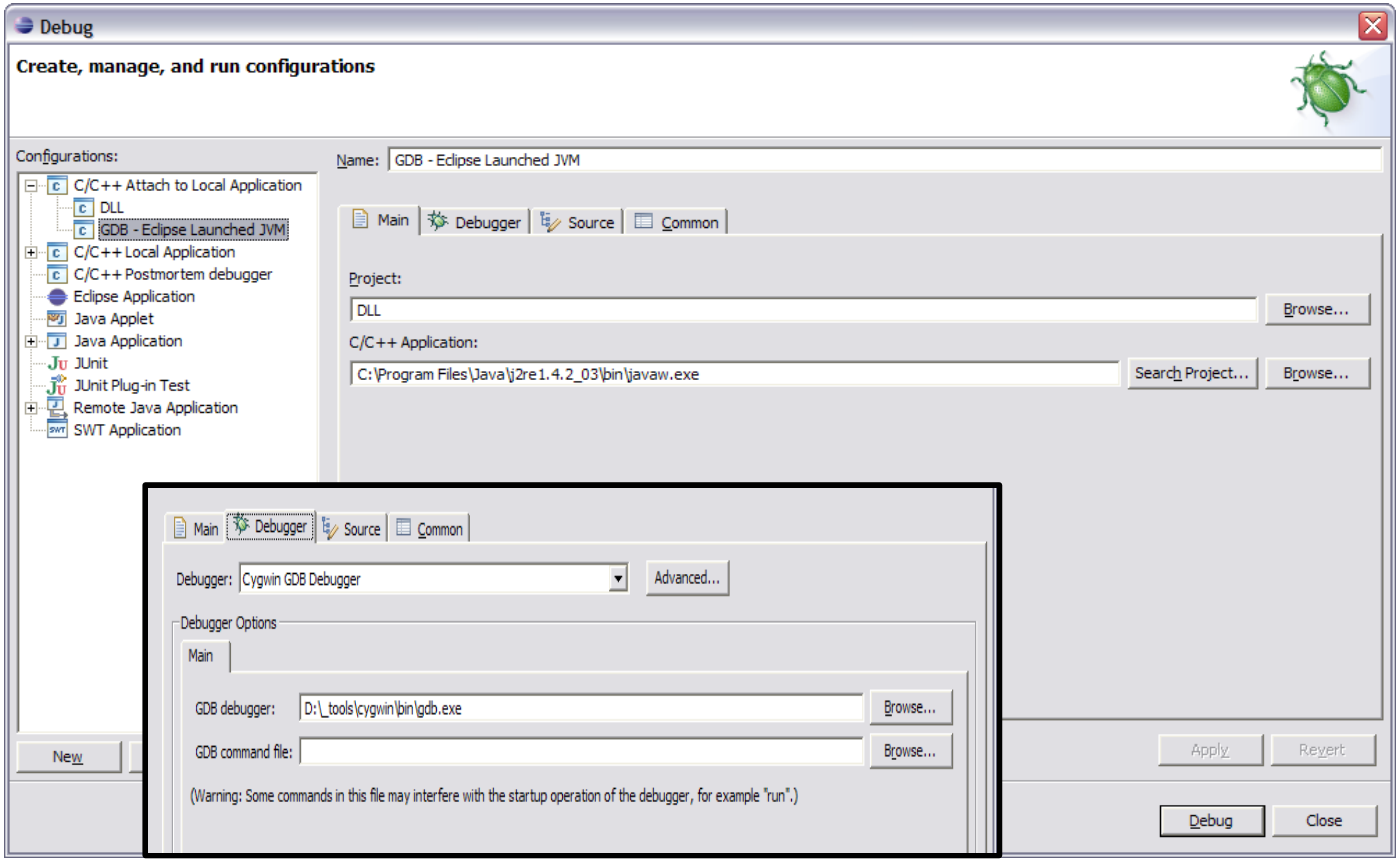
Limitations

- Timing is important in launching processes
- Need to be able identify the Java based process
- Simple JNI based utility DLL can help here
  - It contains simple OS API call to find process id
- Native JNI libraries need to be loaded early

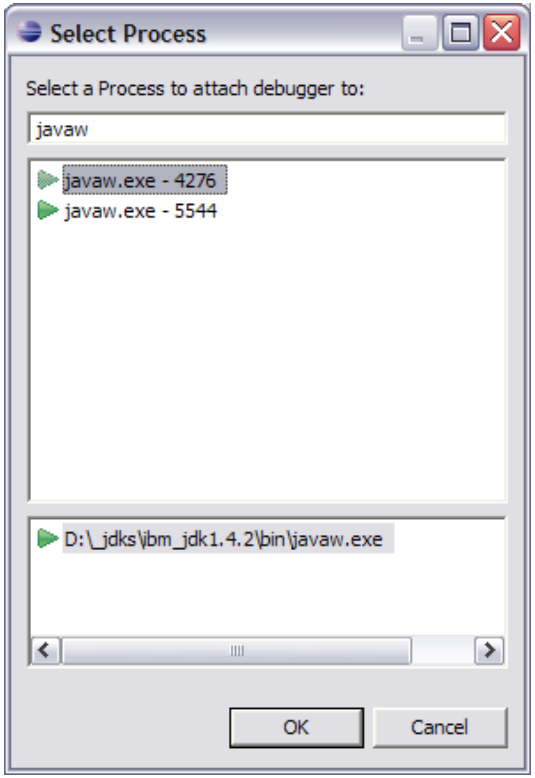
Step 1—Launch App for Remote Java Code Debugging

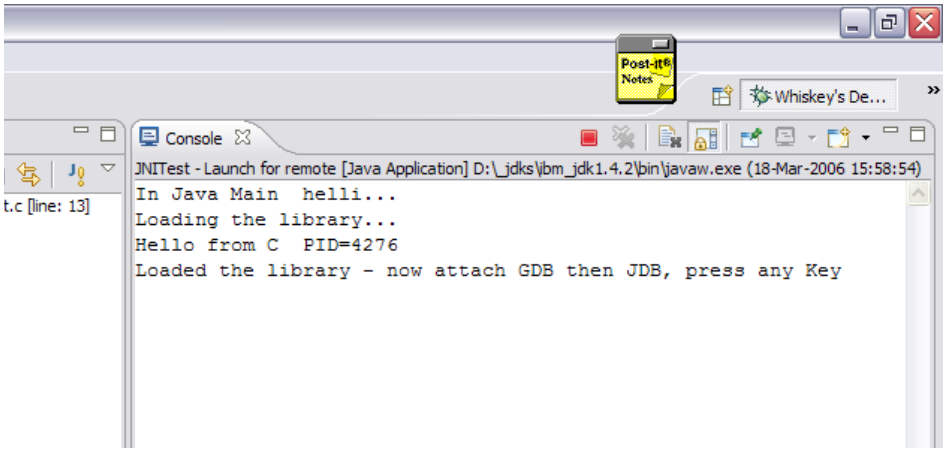


Step 2—Attach GDB to the Application's Java VM



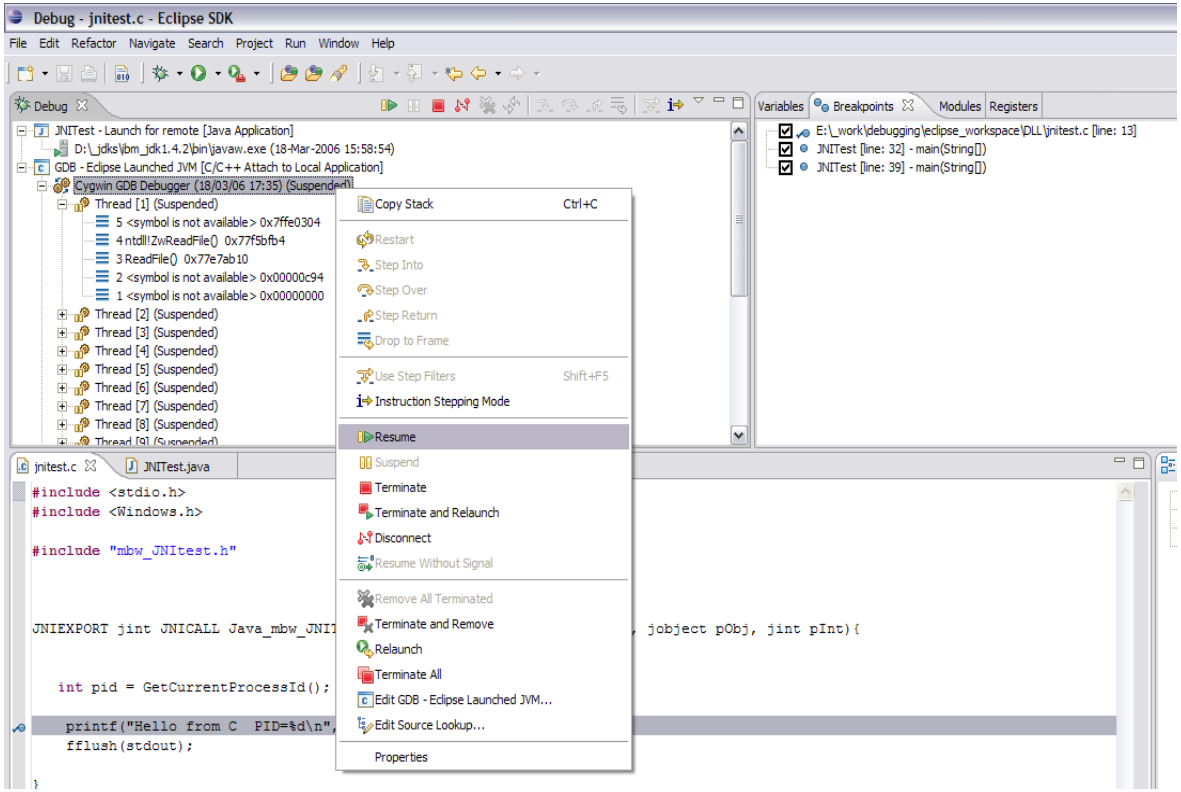
Step 3—Select Process ID



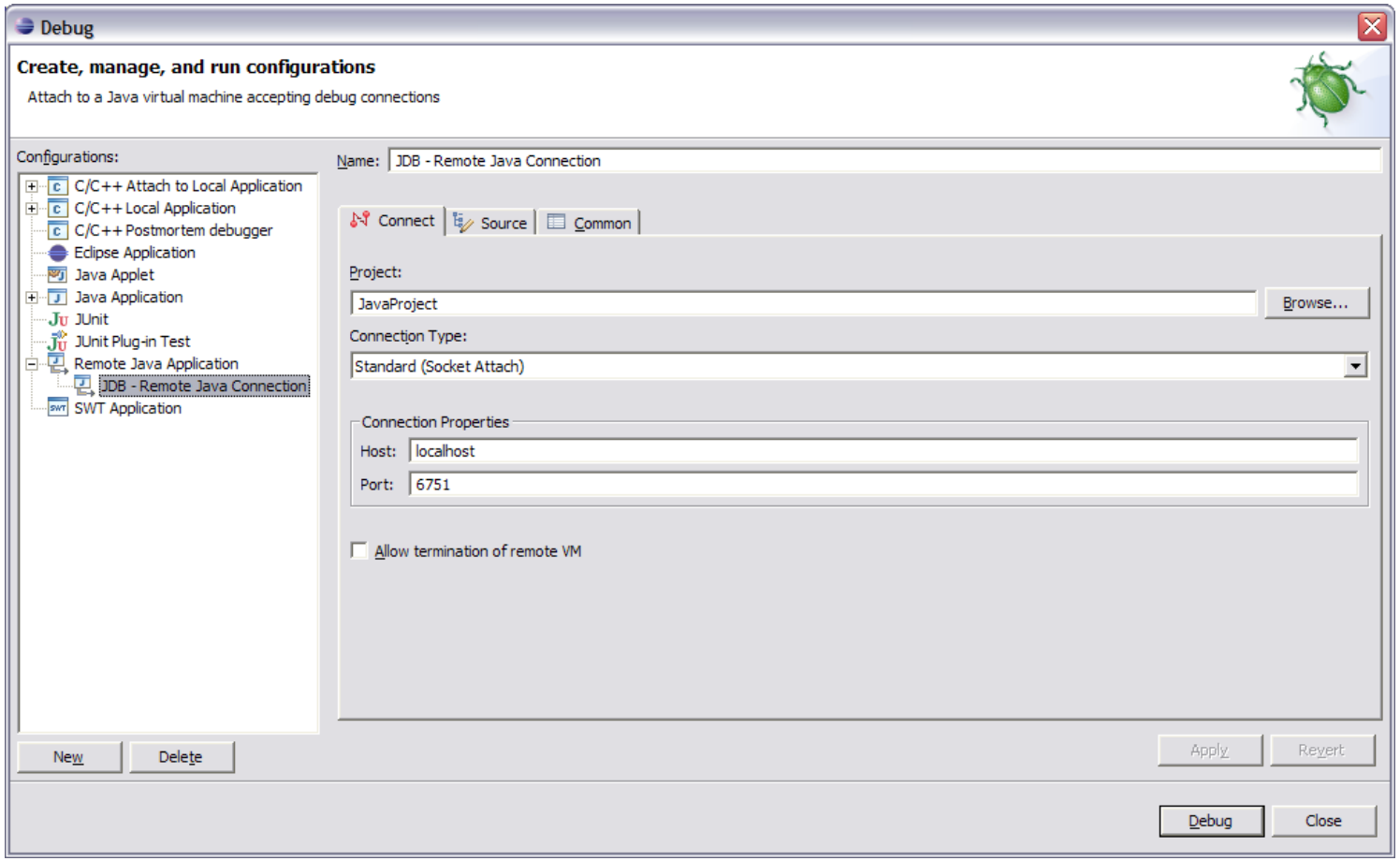


Step 4—Launch GDB

- GDB will attach to the Java VM
- It will suspend the Java VM
- Click resume

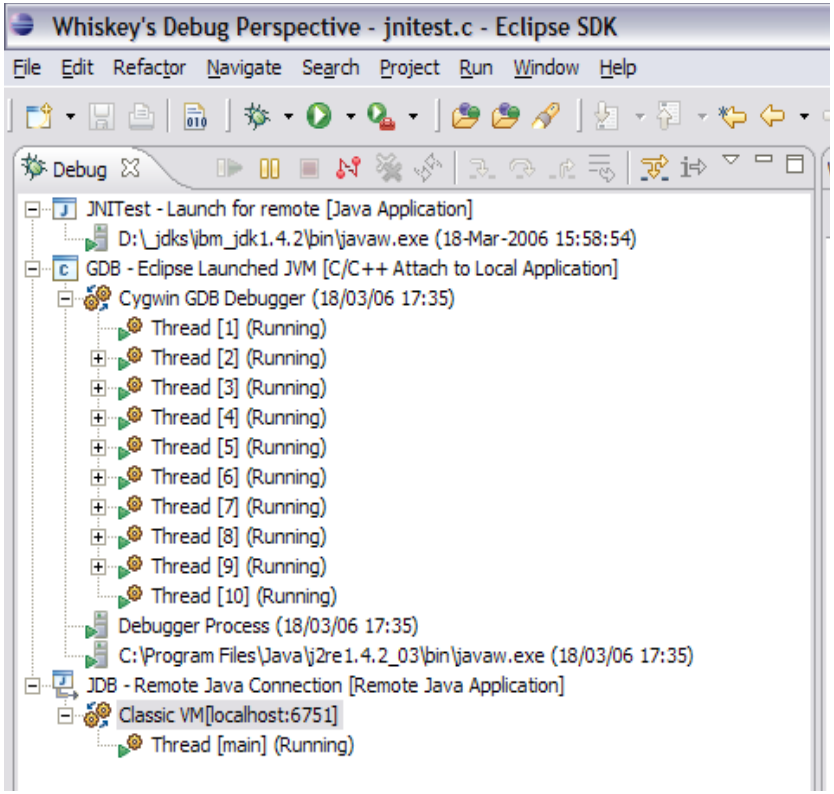


Step 5—Attach the Debugger



Step 6—Ready to Go…

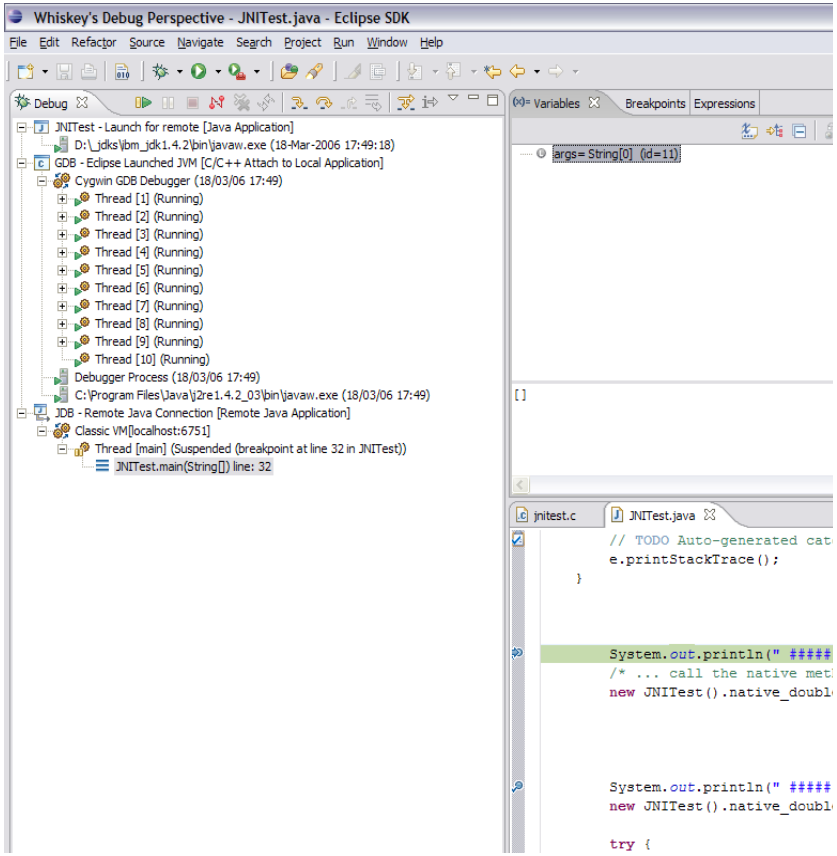
- Three Process
- Main Application
- GDB
- JDB



All are running, application is blocked waiting for input; hit key in Console for main application

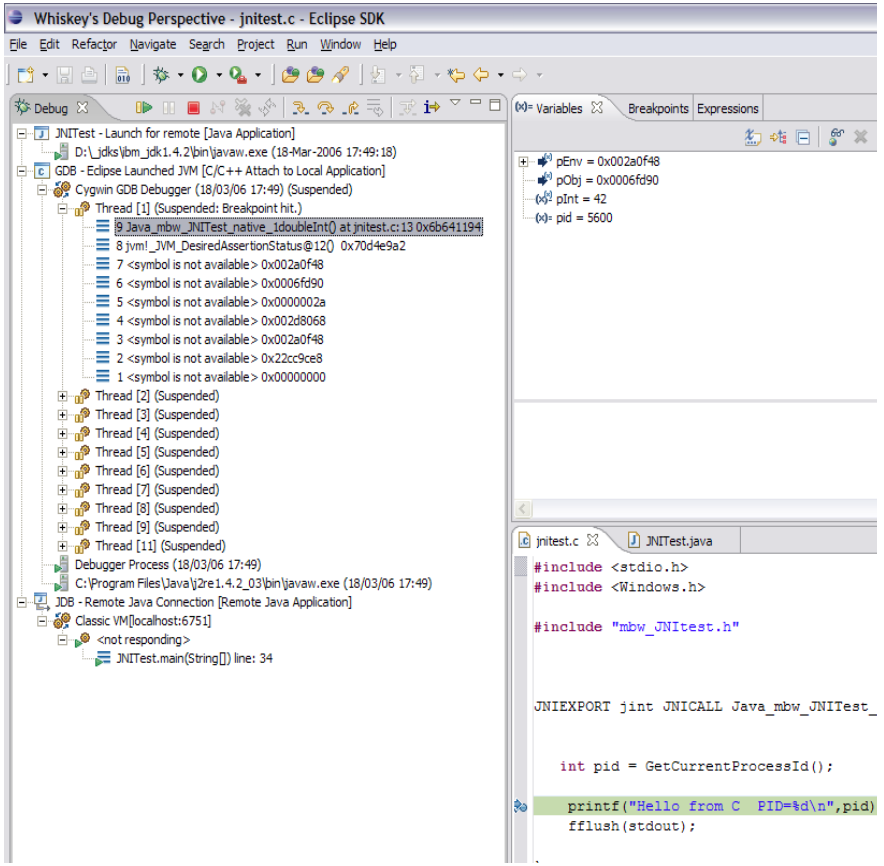
Java Breakpoint is Hit First

- First Breakpoint is in Java
- GDB notes program as running
- JDB notes as suspended
- Step over the native call



C Breakpoint is Hit Next

- GDB has now stopped at the break point
- Can step over native code
- Note JDB reports as not responding
- Step Return—so don't debug in Java VM



DEMO

Debugging in Eclipse

Agenda

Introduction

Building Native Code in Eclipse

How to Do Integrated Debugging With the Demo

Alternatives to Java Native Interface (JNI)

JNI—Some Best Practices

Alternatives to JNI

- JNI isn't hard—but can be daunting
  - Should be used in its place
- Two types of alternatives
  - Tools to simplify usage
  - Totally bypassing

Bypassing Completely

- Typically some form of remoting
  - Could be done with custom network marshalling code
- WebServices infrastructures have improved this
  - http://www-128.ibm.com/developerworks/library/ws-nativecode/index.html?ca=drs-ws1704

Tool Support

- JACE
  - http://reyelts.dyndns.org:8080/jace/release/docs/index.html
- JNI++
  - http://jnipp.sourceforge.net/
- XFunction
  - http://www.excelsior-usa.com/xfunction.html
- Complete List
  - http://weblog.janek.org/Archive/2005/07/28/AlternativestoJavaNativeI.html

Agenda

Introduction

Building Native Code in Eclipse

How to Do Integrated Debugging With the Demo

Alternatives to Java Native Interface (JNI)

JNI—Some Best Practices

JNI Best Practices

- Be careful with the distinction between Local and Global References
  - Local should remain local
  - Watch out for Invocation API
  - Global should always be freed
- Always check the Exception AND return code
- Cache Method, Field, Class IDs
- Avoiding marshalling data across JNI
  - Use new java.nio.ByteBuffer class

JNI Performance

- Often cited to be slow
- Performance can be affected
- Results though are very variable depending on situation
- Avoid marshalling the data across the boundary
  - Use the java.nio.ByteBuffer
- Do local performance tests
  - Test your architecture
  - Test your Java VM

Summary

- Integrated Debugging is possible
- JNI is okay to use if treated with respect
- Do local performance tests

For More Information

- Tools used
  - Eclipse http://www.eclipse.org/
  - CDT http://www.eclipse.org/cdtMinGW
  - Cygwin http://www.cygwin.com
  - MinGW http://www.mingw.org

- Books
  - “The Java Native Interface” Sheng Liang
  - “JNI” Rob Gordon

For More Information

Articles

- Mixed Java and C Debugging
- GBD and JBD Command Line
  - Matthew White—IBM
  - http://www-128.ibm.com/developerworks/java/library/j-jnidebug/index.html?dwzone=java
- GUI Version using Eclipse and Insight
  - Nick Jancewicz—Kinetek Systems, Inc
  - http://www.kineteksystems.com/white-papers/mixedjavaandc.html

For More Information

JNI References

- General Introduction
  - http://en.wikipedia.org/wiki/Java_Native_Interface
- Sun's JNI Reference
  - http://java.sun.com/j2se/1.5.0/docs/guide/jni/index.html
- Performance
  - http://java.sun.com/developer/TechTips/2000/tt0801.html#tip2

Q&A

Integrated Java™ and C Language Debugging using the Eclipse platform

Matthew B White

Staff Software Engineer
IBM UK
http://www.ibm.com

TS-1011