NetRexx Compilation: Using alternatives to javac

22nd International Rexx Language Symposium

René Vincent Jansen 2011-12-06

Aruba, Dutch West Indies.

Agenda

- NetRexx interpretation versus compilation
- Finding the Java compiler
- Using alternatives to javac
- Designing new translator options and behaviour

NetRexx

- You can choose between interpreted execution
- Without class declaration, 'double scripting mode'
- Compiler is fast enough for most development cycles
- (Very seldom) the interpreted execution behaves differently
- Two parts: NetRexx translated to Java
- Java translated to bytecode

Java Source

- Use options format for readability
- Use options noformat for exact error location (corresponding debug info)
- New option in 3.01: -keepasjava - no need to rename the generated sourcefile
- Use option comments to leave comments in generated java, e.g. for usage in Javadoc documentation generator

Compiling generated source

- Default action is to automatically call the Java compiler
- Option nocompile skips this step

- The Java compiler is a class in the tools package
- The javac executable is just a wrapper - a native program, javac on Unix and javac.exe, that starts a VM and calls the compiler class

Finding the compiler class

- Packaging the class libraries is up to the vendor or producer of the specific Java implementation on the platform
- Sun, for example, changed this packaging and put the compiler after Java 1.x into a special tools package
- This package should be on the classpath
- To successfully get NetRexx to work, you should understand classpath
- You can get lucky and have it work without understanding, but then you still will have difficulty putting together larger Java applications

Excursion: how does javac find the compiler class?

- In RxTranslator:
  - javacok=sun.tools.javac.Main(System.out, 'javac').compile(args)
- You could replace this with
  - javacok=org.eclipse.jdt.core.compiler.batch.BatchCompiler.compile(args, PrintWriter(System.out), PrintWriter(System.err), null)
- This class needs to be on your classpath
- Best way to get it is download a current Eclipse and find the jar

Current translator behaviour

- NetRexx reads the classes (including compressed archive) on the classpath on startup
- It makes an index of these
- There is help built in for only a very small set of platforms (of which recurring problems were reported)
- Sometimes the tools package is well hidden, or the compiler is not in an archive that contains the word 'tools' at all

Well known locations

- Most known locations of OS/JVM release combinations are, or will be soon, documented in the NetRexx User's Guide
- More action is required:
  - Sometimes you just can't find it
  - That might be because it isn't there (at some installations you might find a usable JRE but no JDK)

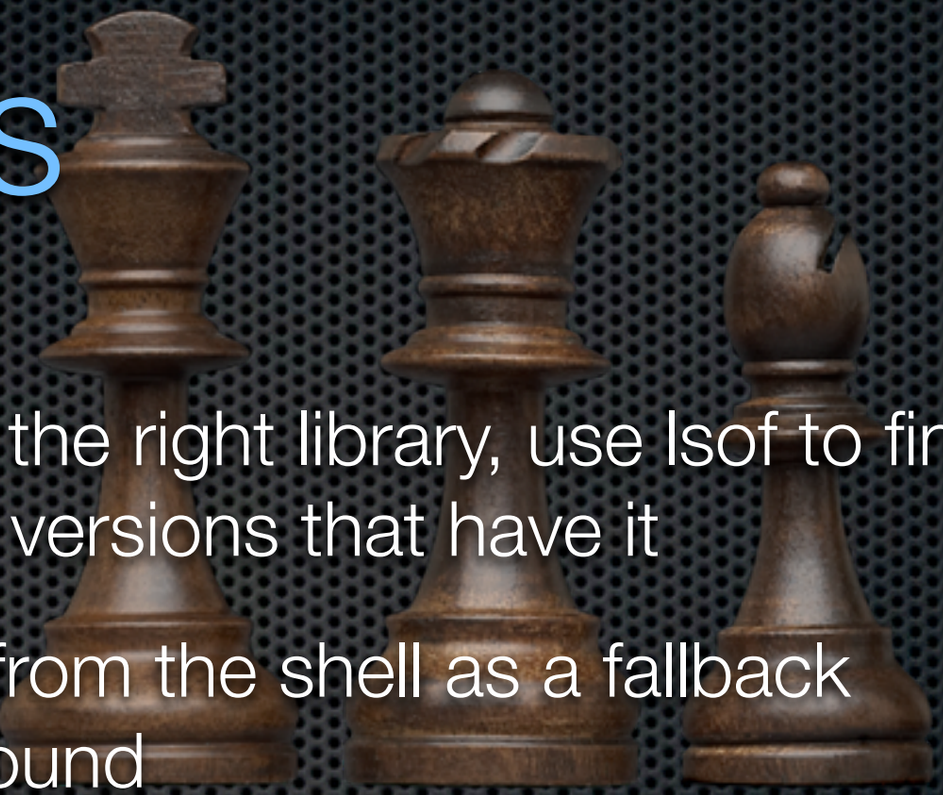
What to do?

- When you encounter such a situation, you can BYOC
- Bring your own compiler
- I was there recently and that is how I found out about the Eclipse batch compiler
- There is a separate download of it available if you do not have the time or opportunity to install the whole IDE
- Remember that a jar file, to the OS, is just data.As long as you can put is somewhere reachable, you're OK

Does it work?

- Yes, I am not using Javac anymore, even if I know where to find it
- Warnings are better, and have options to switch on and off, from very useful to very pedantic
  - For example, non-static use of static methods
- It offers a single compiler over platforms, so I never have doubt if there is a Mac, Windows or Linux specific issue with the compiler implementation

Other strategies



- When the javac program can find the right library, use lsof to find the files that it has open - on unix versions that have it
- We should look into calling javac from the shell as a fallback option when tools.jar cannot be found
- There are other alternative compilers around - this is up to the individual
- We could have maven download a compiler when a dependency is not resolved
- Like message: java compiler not found. Download and install one?

Translator Modifications

- Wanting option compile, but with an alternative compiler seems to be a valid wish
- Maybe should introduce an environment variable that specifies an alternative compiler implementation, including complete path to its classfiles
- Combinable with the 'download a compiler?' approach
- (MacOSX Lion asked me 'download java for you?' when I started my first NetRexx program after installation)

How to design

- NetRexx already has a lot of options
- The defaults should be usable for new users rightaway
- Error situations should have severall fallback options
- If there are drawbacks, these need to be clearly explained