Eclipse and Embedded Linux Developers: What It Can and Cannot Do for You

Anna Dushistova

anna_dushistova@mentor.com

Comprehensive Solutions for

Android™ ▪ Nucleus® ▪ Linux®

Mobile & Beyond · 2D/3D User Interfaces · Multi-OS · Networking

Why IDE?

- From wikipedia:
  - "...IDEs are designed to maximize programmer productivity by providing ***tightly-knit components*** with ***similar user interfaces***. This should mean that the programmer has ***much less mode switching*** to do than when using discrete development programs..."

Typical Content of an IDE

- a source code editor
- a compiler and/or an interpreter
- build automation tools
- a debugger
- SCM integration

About Eclipse

- What is it?
  - Eclipse is an open source community, whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle.
- History of Eclipse
  - Started in November 2001 by Borland, IBM, MERANT, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft and Webgain
  - On Feb 2, 2004 the Eclipse Board of Stewards announced Eclipse's reorganization into a not-for-profit corporation.The founding Strategic Developers and Strategic Consumers were Ericsson, HP, IBM, Intel, MontaVista Software, QNX, SAP and Serena Software.

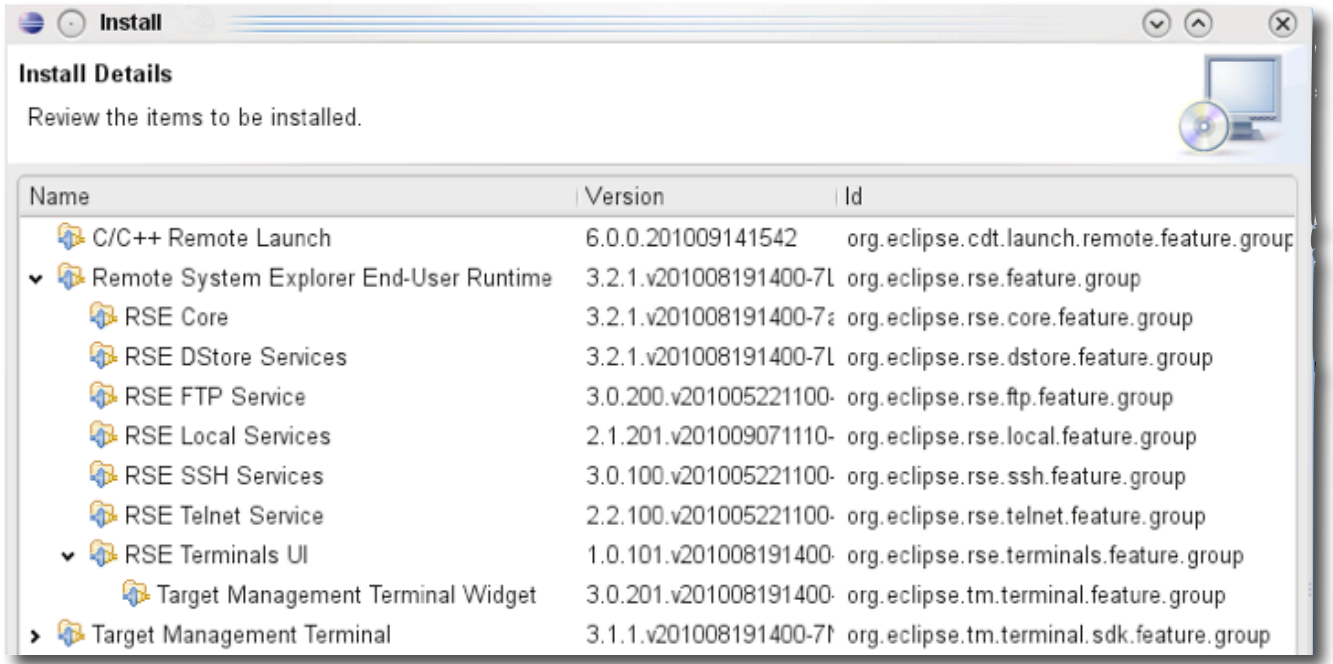
What can Eclipse offer?

- C/C++ Development Tooling(CDT) for edit/compile/debug cycle
- Target Management (TM) for working with remote hosts
- CVS, SVN and Git integration
- Linux Tools for different Linux specific tools (autotools, oprofile, lttng, etc.)

Some companies behind these projects

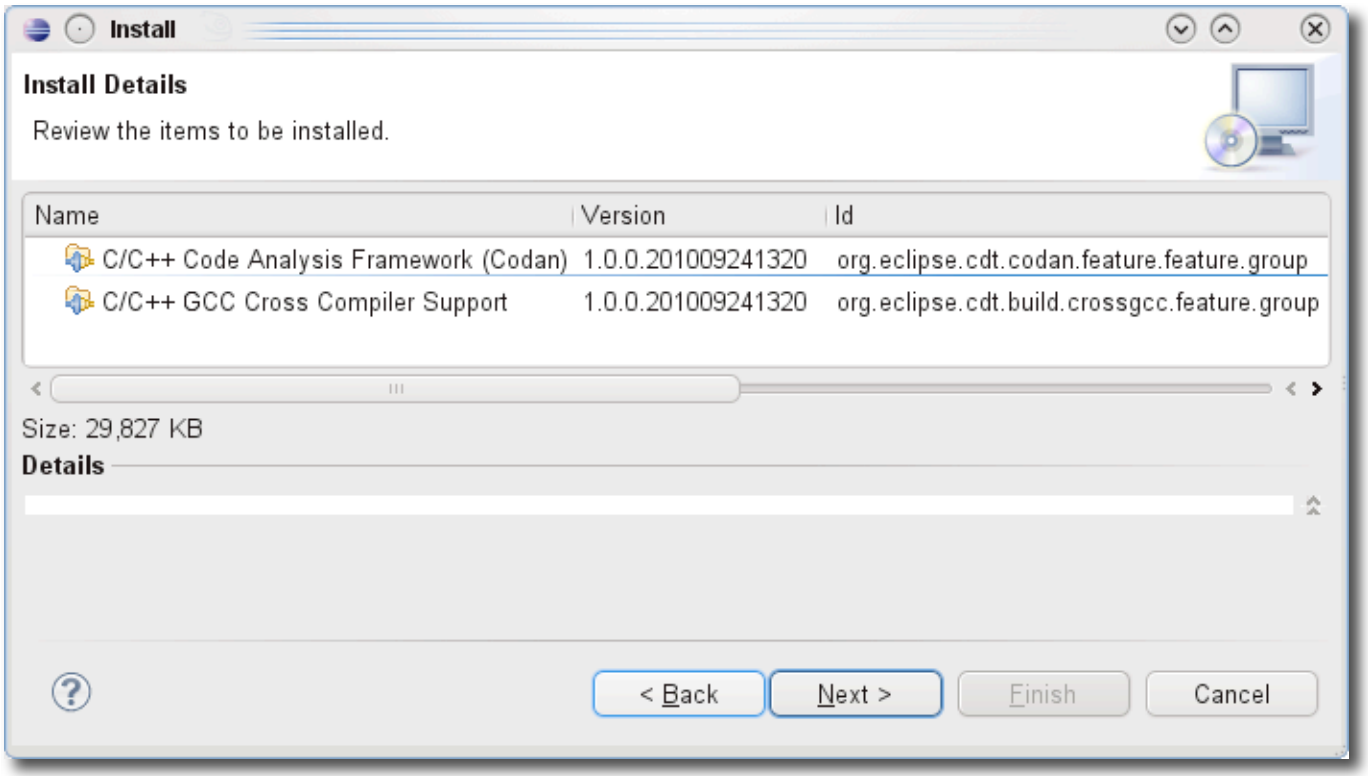
But...

- Is it possible to just use Eclipse with these plugins as is for embedded Linux development?
  - Let's start with Eclipse IDE for C/C++ Linux Developers package (http://www.eclipse.org/downloads/packages/eclipse-ide-cc-linux-developers-includes-incubating-components/heliossr1)
  - Install missing Target Management features



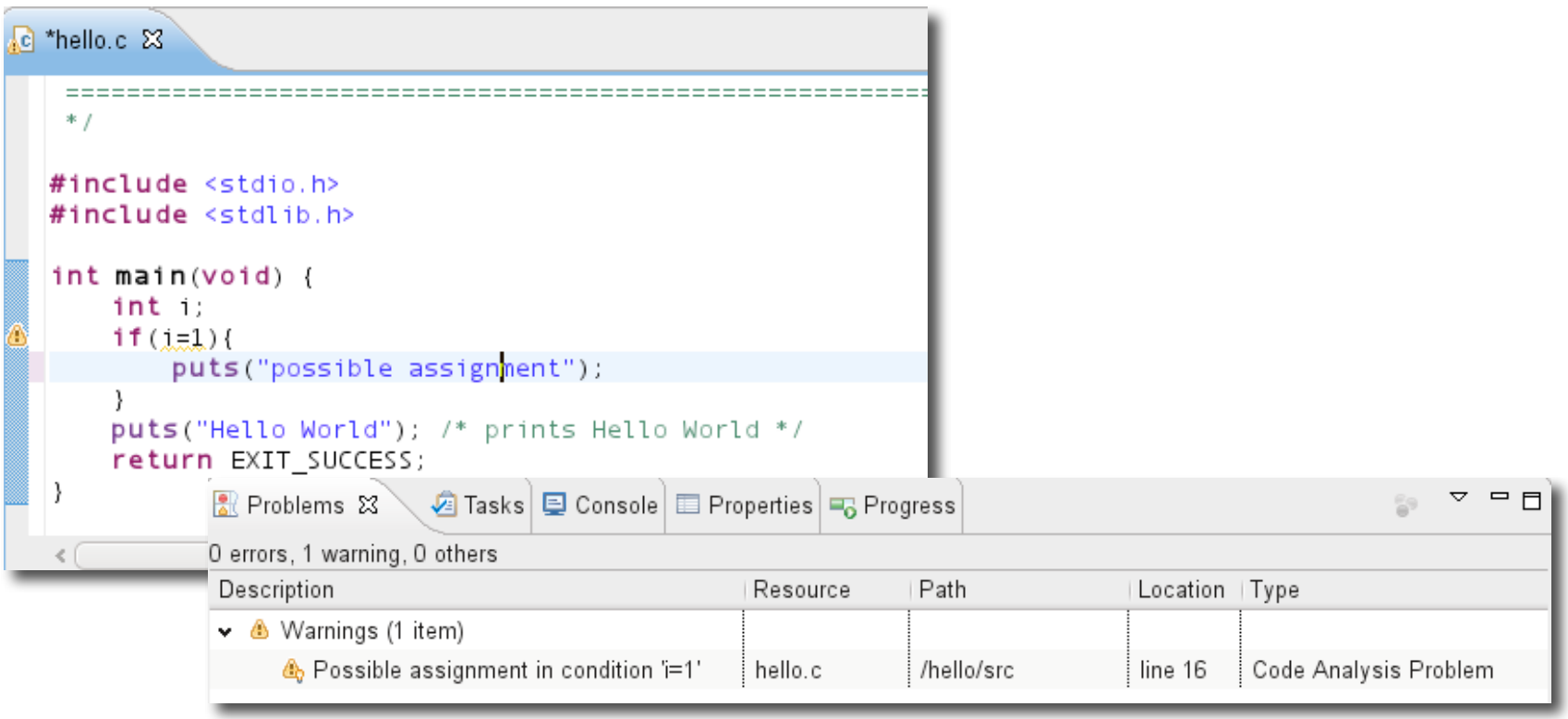
Install additional CDT features

- Not everything we need is in that package
  - Download http://www.eclipse.org/downloads/download.php?file=/tools/cdt/releases/helios/dist/cdt-master-7.0.1-I201009241320.zip
  - Install Codan and cross compiler support



Working with C/C++ code

- New in CDT 7.0:
  - Code Analysis Framework “Codan”
    - For additional information see http://wiki.eclipse.org/CDT/designs/StaticAnalysis



Building your application

- Works only for a local toolchain out of the box!
- How can we integrate CDT with our cross compilation tools?
  - Write a plugin for Eclipse that adds support for it
  - Try using “Cross GCC”
  - Redefine settings for each project manually

Defining your own toolchain

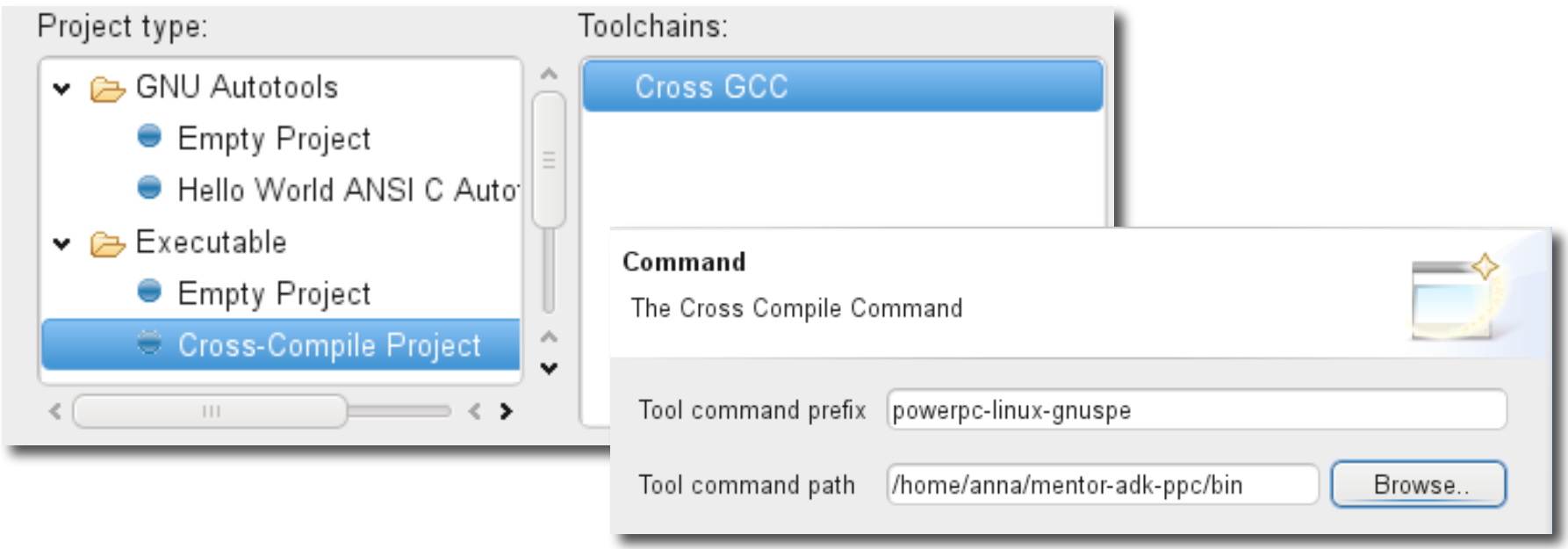
- If you don't need any special settings, just extend existing definitions for "Linux GCC"
- The more special your needs are, the more changes you'll have to make
- Changes are not obvious and require some Eclipse experience

```
name="Anna's GCC" osList="all">
<targetPlatform
          id="cdttest309024.gnu.platform.base"
          name="Debug"
          binaryParser="org.eclipse.cdt.core.ELF"
          osList="linux,hpux,aix,qnx"
          archList="all">
</targetPlatform>
<builder
          superClass="cdt.managedbuild.target.gnu.builder"
          id="cdt.test309024.builder.base">
</builder>

<tool command="$(CXX)"
    id="cdt.test309024.cpp.compiler" isAbstract="false"
    name="Anna's GNU G++ Compiler" superClass="cdt.managedbuild.tool.gnu.cpp.compiler">
</tool>
```

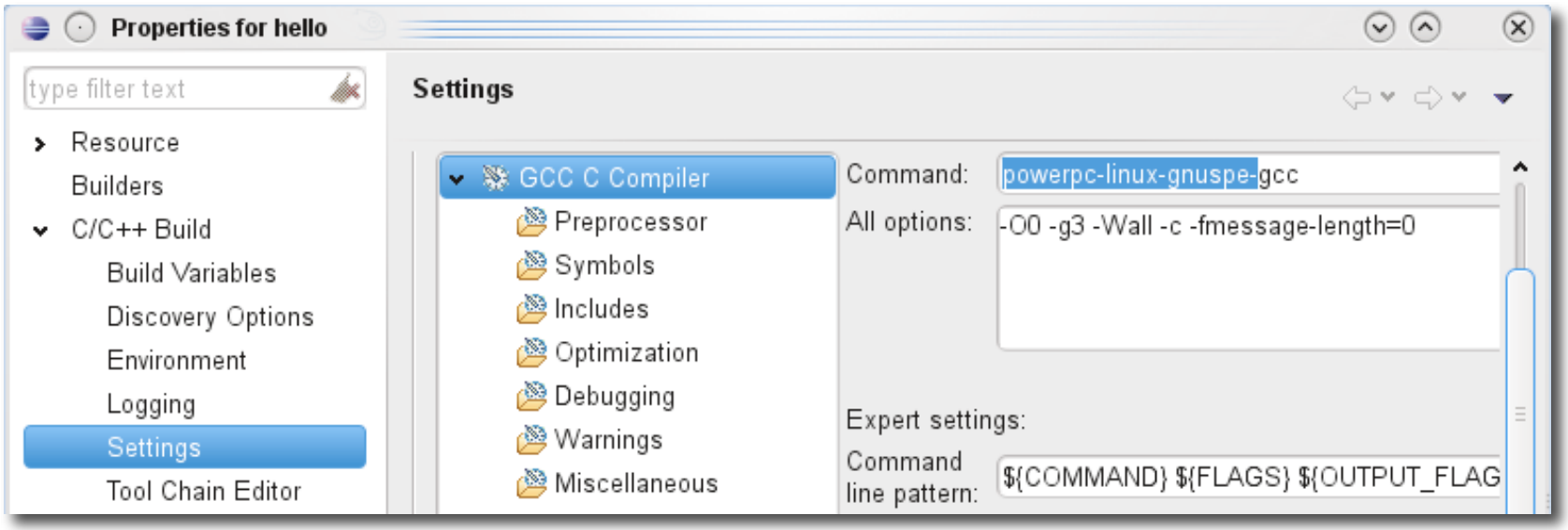
Using “Cross GCC”

- Asks you for just two settings:
  - path
  - prefix(make sure you add dash in the end)
- Not quite there yet



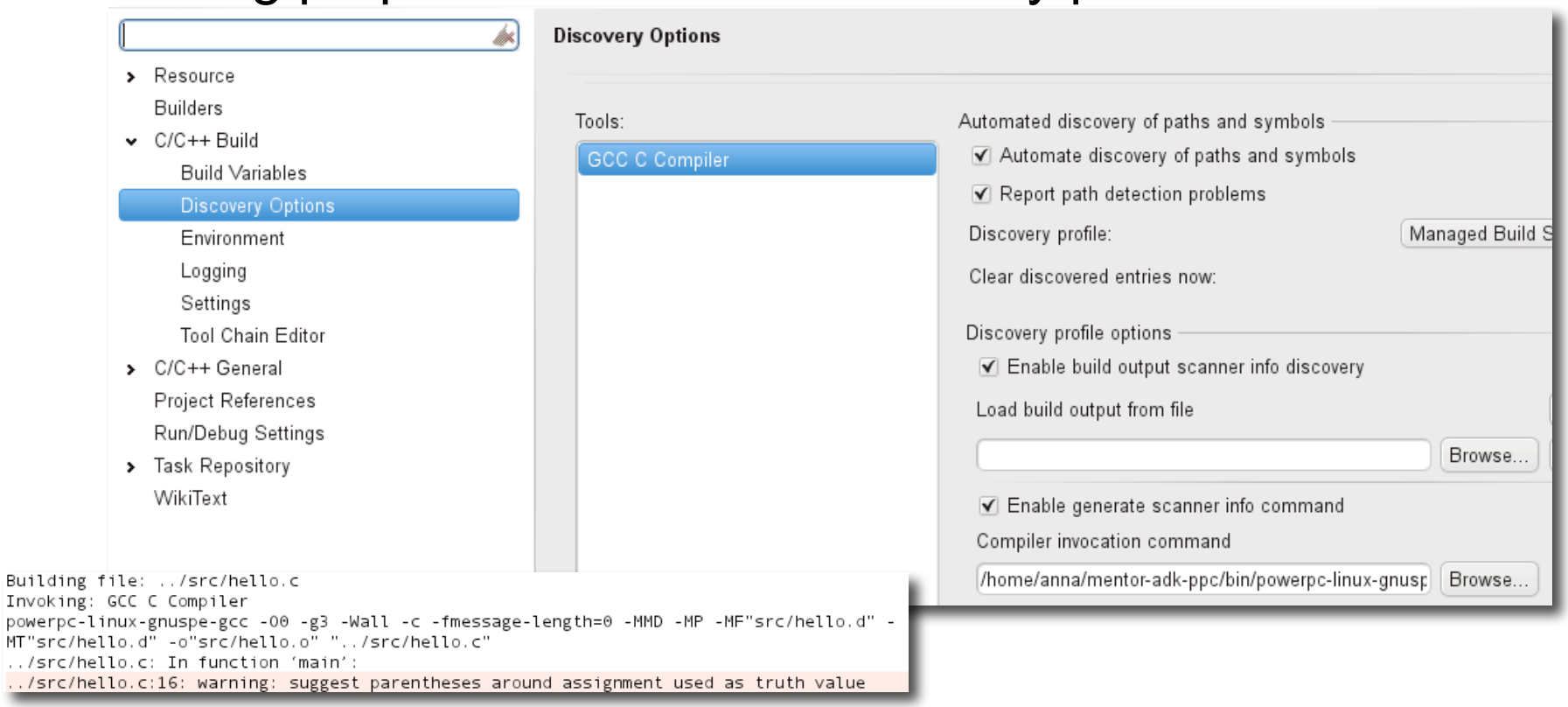
Redefining settings manually

- Has to be done for each configuration in each project in use
- Requires the following modifications of project properties:
  - changing gcc to your <arch>-gcc in Settings for each tool



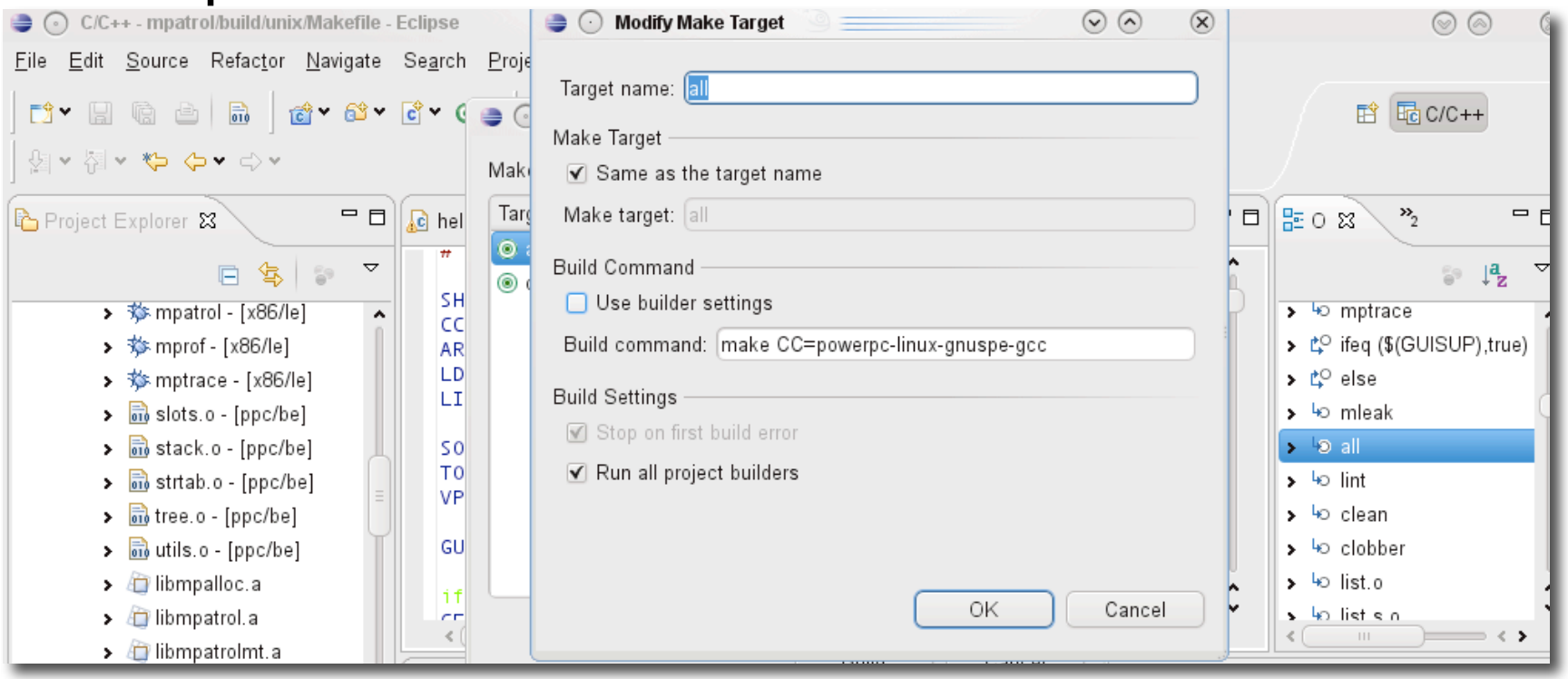
Redefining settings manually - continued

- Modifying PATH variable(adding your cross-toolchain path)
- Setting proper command for discovery profile



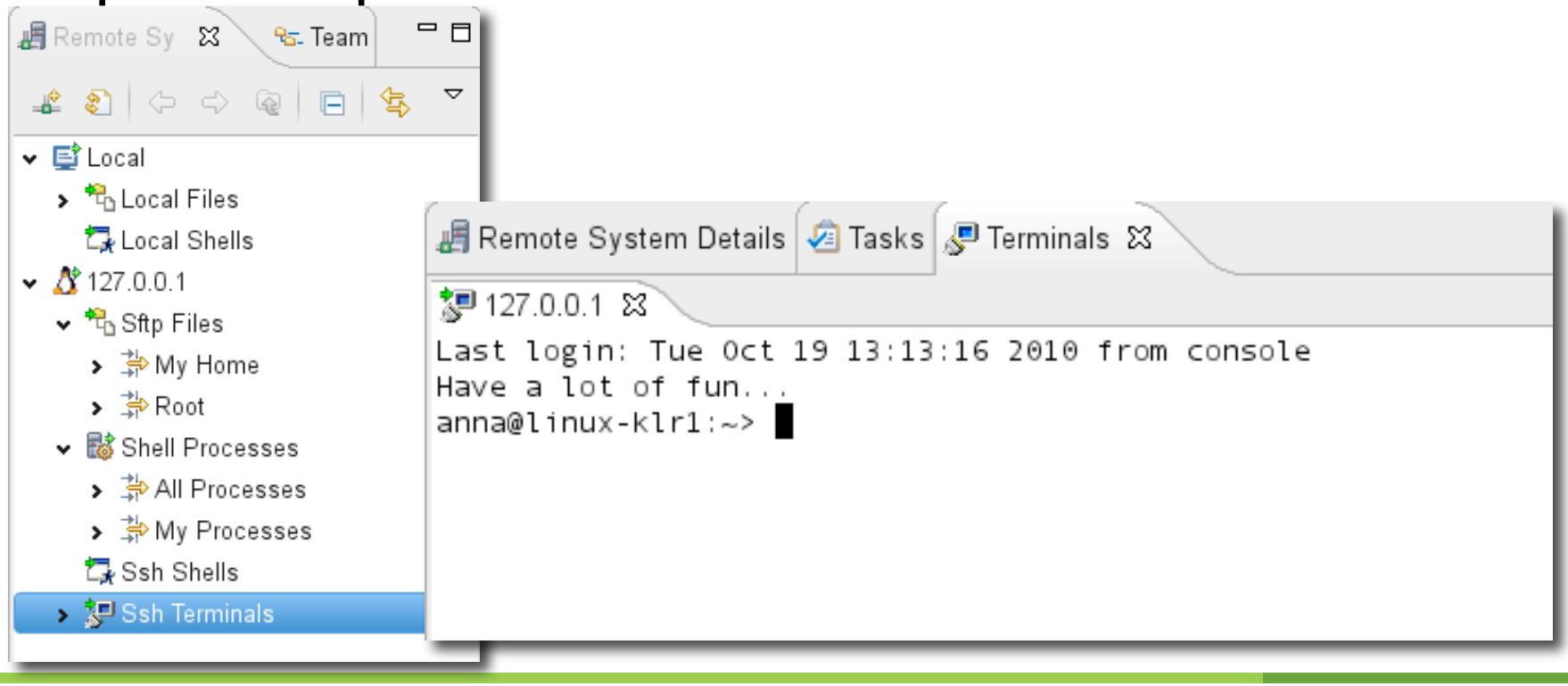
Makefiles

- Generated makefiles are not meant to be used outside CDT
- Better use “Makefile projects” for more control of your build process



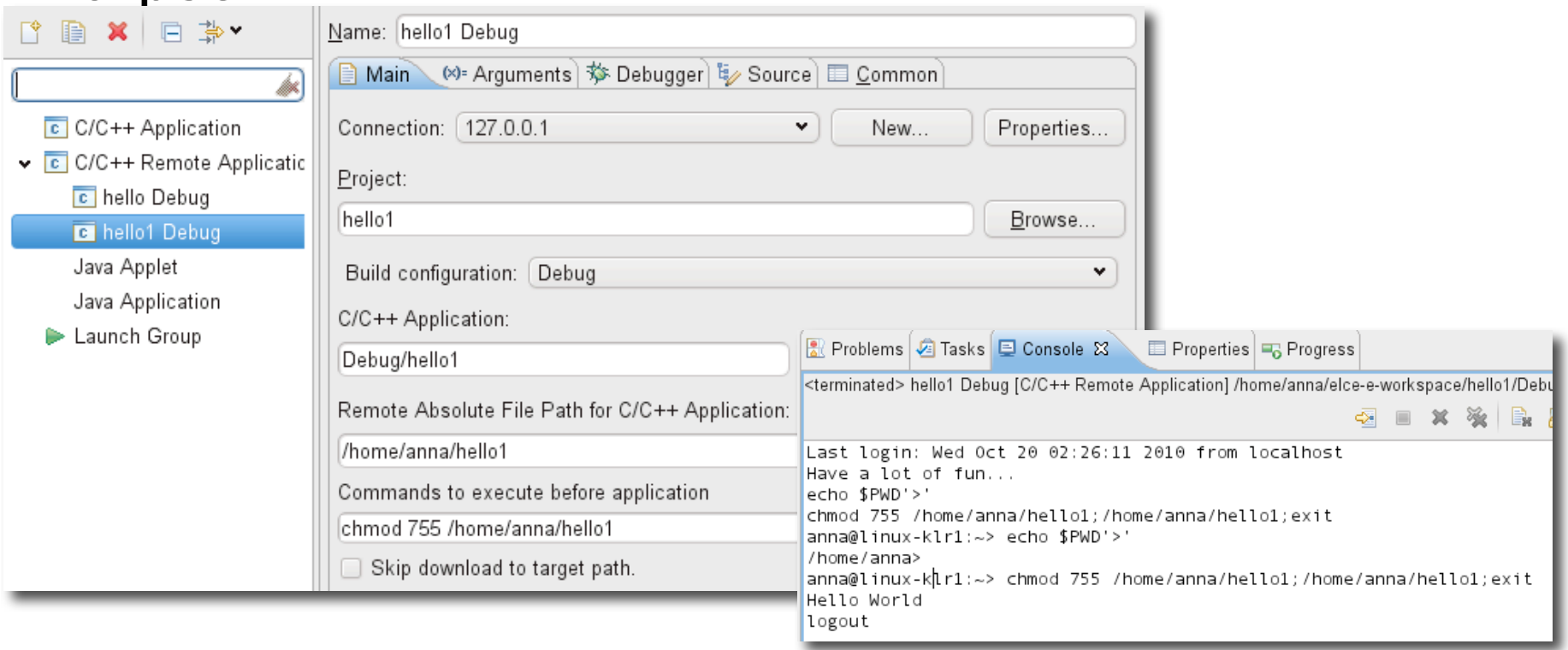
Working with remote systems

- If you have only serial, you can use the Target Management Terminal for board bring-up
- Once you have ethernet, you can use Remote System Explorer capabilities



Running your applications

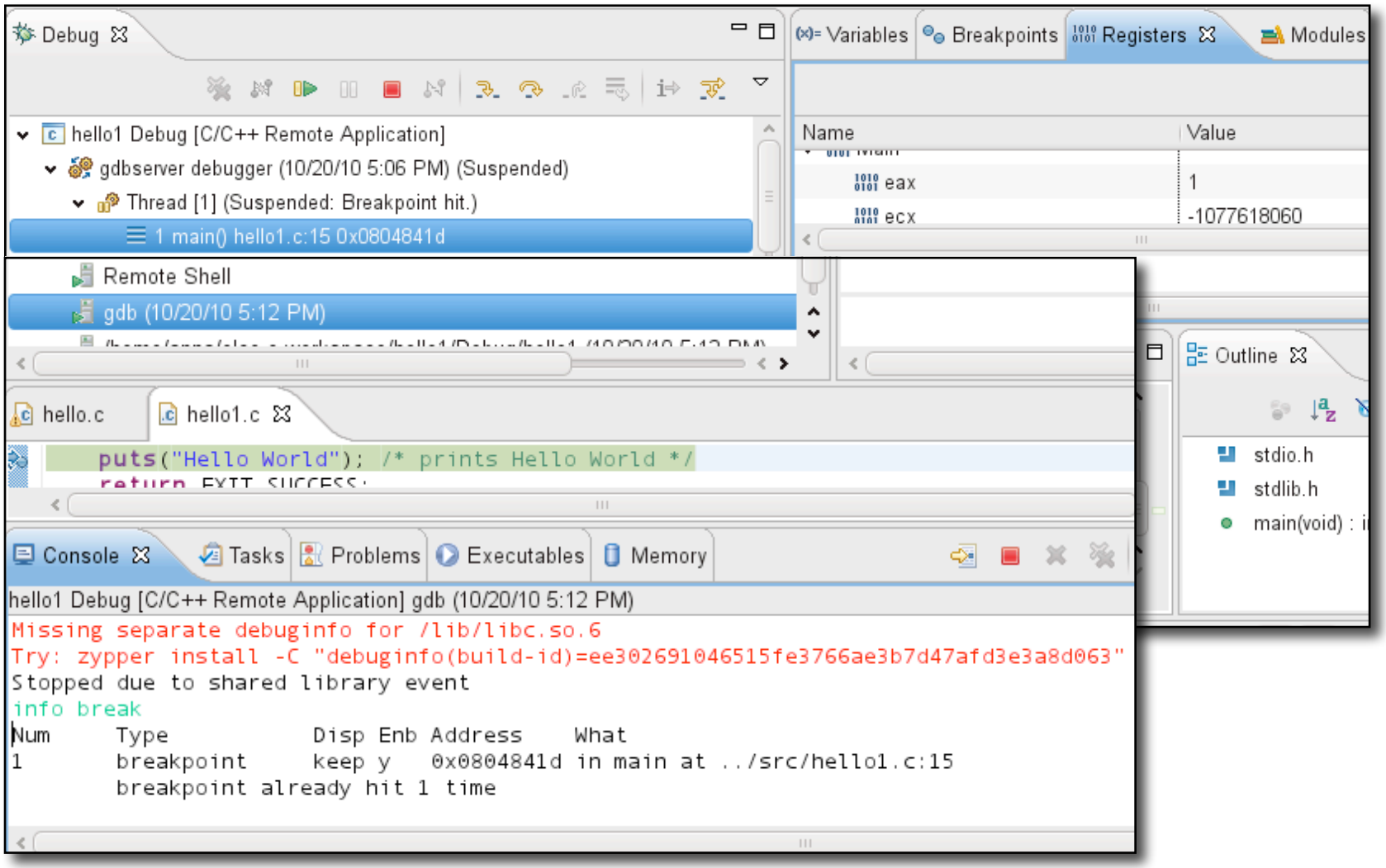
- Use Remote Systems Explorer to copy your application on target and then terminal to launch it
- Use Remote Launch to launch your application from Eclipse



Debugging your applications

- Variety of choices:
  - remote debugger (launches gdbserver on remote side automatically, doesn't support attaching to a running process)
  - "old"(CDI) CDT-GDB integration(you have to launch gdbserver manually, supports attach)
  - "new"(DSF) CDT-GDB integration(you have to launch gdbserver manually, supports attach)
- More coming soon:
  - Eclipse Debugger for C/C++(agent on remote, Java implementation on host based on DSF)
  - Agent based debugger(won't use DSF)

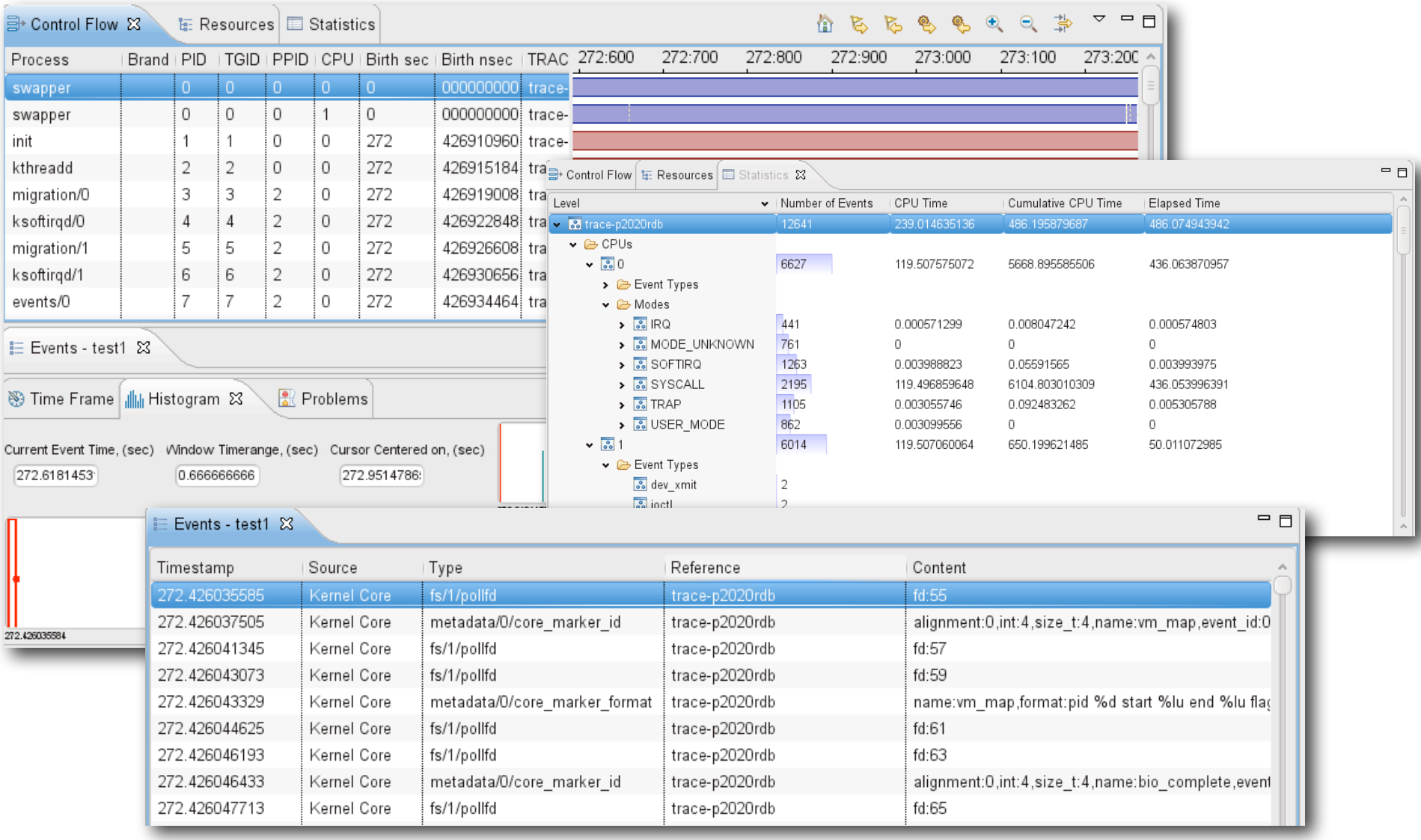
Debugging your applications-continued



Linux measurement and diagnostic tools

- Relatively new in Eclipse
- Profiling
  - oprofile: remote launching is not supported, only local
  - valgrind: same
- Tracing
  - doesn't have remote control
  - requires building and installing lttngtrace library prior to using the tool
  - requires creating an lttng project to import traces

Eclipse LTTng Integration



The result of our experiment

- Technology is there!
- But:
  - almost every piece requires non-trivial actions to make it work the way we need
  - some are just not possible to use for cross development
  - pieces do not integrate together very well
- Overall, the set of plugins and features we installed can hardly be called an IDE.

Questions?

Thank you!