Providing a Development Environment for Linux

Presented by:
Rick Moseley
Senior Software Developer
Red Hat, Inc., Huntsville, Alabama
Committer for CDT-Contrib
rmoseley@redhat.com
Red Hat Eclipse Development Team

- John Healy, Manager
- Andrew Overholt
- Keith Seitz
- Chris Moller
- Phil Muldoon
- Rick Moseley
- Jeremy Handcock
- Aaron Luchko
- Benjamin Konrath
- Jeffrey Pound
Red Hat History with Eclipse

**Spring 2002**
- Allocated resources to explore Eclipse as an addition to our desktop offerings

**Summer 2002**
- Setup a team to actively contribute to the development of Eclipse and to aggressively QA the Eclipse releases on Linux

**Spring 2003**
- Intense testing began on the Linux platform
- Submitted several dozen bugs and feature requests
- Started development on several plugins (Oprofile, Changelog, RPM, Hoverhelp, etc.)
Red Hat History with Eclipse (continued)

Summer 2003
- Resources assigned to work with the Visual Editor team

Summer 2004
- Incorporated Eclipse into our “Red Hat Developer Suite” product

Currently
- Currently working on a “gcj” version of Eclipse which requires no JVM
- Planning to have a “gcj” version of Eclipse in Fedora Core 4
- Improving current plugins
Changelog Plugin

Purpose

- To allow developers an easy way of keeping changelogs up-to-date

Where to get it

- http://people.redhat.com/pmuldoon/update
- Use this URL as an update site

How to configure it

- From Windows->Preferences->Changelog
- Allows user to set “Author Name”, “Author Email” and “Formatters”
  - “Author Name” and “Author E-mail” are used to create a new entry in
    the changelog file when the changelog plugin is activated.

How to activate it

- Place cursor in desired location then:
  - CTRL-ALT-C or Edit->Changelog Entry
Changelog plugin (continued)

How it Works

- When activated:
  - Searches directory structure for a “Changelog” file
  - If a “Changelog” file is found:
    - It opens it in an editor
    - Places the beginning line for the entry at the top of the file
    - Gets the name of the method/function the cursor is in and includes it in the entry followed by a colon
  - If a “Changelog” file is not found:
    - A “Changelog” file is created in the directory of the file where the cursor is located
    - The same thing happens as when a “Changelog” file is found

Languages Currently Supported

- Java, C/C++
Changelog Plugin
(continued)

Preferences Page
Changelog
Plugin
(continued)
Changelog Entry
Example
Changelog plugin (continued)

Points of Information

- The Changelog plugin has two extension points, one that allows the addition of language parsers and another that allows the addition of other formatters.
- Currently, the GNU Formatter is the only supplied formatter and Java/C/C++ are the only languages supported.
Changelog Plugin (continued)

Possible Future Enhancements

- Separating the C/C++ and Java parsers
- RPM Changelog formatter
OProfile Plugin

Purpose

- OProfile is a powerful, low-overhead system profiler that leverages processor hardware performance counters to enable system-wide profiling of a wide variety of events.

Where to get it

- CVS repository at dev.eclipse.org:/home/tools/org.eclipse.cdt-contrib

Requirements

- SMP kernel must be running
- OProfile kernel module must be turned on and loaded
- Project developed with Eclipse CDT
- Root password as the OProfile module necessarily runs as root

Other resources

OProfile Plugin (continued)

Examples of Items that can be profiled

- Clock cycles that a CPU is halted
- Number of instruction fetch misses
- Number of L2 data loads or stores

Help

- Available under the Eclipse help as “OProfile Basic Tutorial”

How to launch the OProfile Daemon

- Select Run->Run to open Run dialog
- Select “Profile C/C++ Application”
- Fill in the appropriate information for all of the tabs
OProfile Plugin (continued)

OProfile daemon setup screen – page 1 of 2

Create, manage, and run configurations

- C/C++ Local
  - Pi
  - Java Applet
  - Java Application
  - JUnit
  - JUnit Plug-in Test
- Profile C/C++ Application
  - New_configuration

Application: Pi

Profiling Setup

Configuration

Common

Kernl image file:

/boot/vmlinux-2.4.21-15.ELsmp

- Verbose
- Per-application shared libs samples files
- Per-application kernel samples files

New | Delete

Apply | Revert

Run | Close
OProfile Plugin
(continued)

OProfile daemon setup screen – page 2 of 2
How to view OProfile samples

- From the perspective toolbar, select the OProfile perspective or alternatively Window->Open Perspective->Other->OProfile and choose a session to display
OProfile Plugin (continued)

- Now select the desired executable to show the profiled information and the function within the executable.
OProfile Plugin (continued)

Issue – GPL versus EPL

- One module in OProfile plugin is GPL'ed - opxml
- The rest of the plugin is EPL'ed, the OProfile kernel module itself is GPL'ed
- The opxml module uses the OProfile kernel module's libraries to read sample files output by the kernel module. It then generates XML output that can be parsed by the OProfile plugin.

Possible Future Enhancements

- Integrating with Hyades project
RPM Plugin

Purpose

- Allow developers to import source RPMs into a C/C++ project, modify them and re-export them as source and/or binary RPMs
- Allow developers to export simple C/C++ projects as source/binary RPMs

Where to get it

- dev.eclipse.org:/home/tools/cdt-contrib

Requirements

- Eclipse core
- Eclipse CDT
RPM Plugin (continued)

Setup

- Windows->Preferences->RPM Plugin
- Configurable items
  - Spec file changelog items
    - Author name
    - Author e-mail
  - RPM plugin work area
  - Shell commands
RPM Plugin
(continued)

RPM Plugin Preferences Page
RPM Plugin (continued)

Functions

- Import source RPMs into a C/C++ project
- Export previously imported RPMs as binary and/or source RPMs
- Export simple C/C++ projects that have never been in RPM format previously
- Create patches with user-entered patch tag for inclusion in the source rpm export
- Spec file modification
  - Modifies version/release numbers to user-specified values
  - Adds “Patch:”/“%patch” statements
  - Adds user-specified changelog comments
RPM Plugin (continued)

Importing a source RPM

- File->Import->Source RPM
- Select source rpm to import
- User-selectable Options
  - Create a project to import into
  - Apply all patches
  - Run AutoConf (configure) if applicable
RPM Plugin (continued)

Source RPM import screens

Select
Imports a Source RPM to a project.

Select an import source:
- Existing Project into Workspace
- External Features
- External Plug-ins and Fragments
- File system
- Source RPM
- Team Project Set
- Zip file

Select a project to import an SRPM into
Select a project to import an SRPM into

SRPM Name
/home/rmoseley/helloworld-2.2.src.rpm

Import SRPM into…
- Select a project...
- pump
- slocate
- test

Check out as a project configured using the New Project Wizard

Build Options
- Apply all patches to pristine source
- Run AutoConf (configure) if applicable

< Back  Next >  Finish  Cancel
< Back  Next >  Finish  Cancel
RPM Plugin (continued)

Exporting a Binary/Source RPM

- File->Export->Source/Binary RPM
- Select a C/C++ project to export
- Select a spec file to use or use the default
- Enter a version and/or release number for the new RPMs
- If the RPM plugin detects no changes to a previously-imported RPM or this is a project that has never been exported, just click “Finish”
- If the RPM plugin detects changes made to the project since it was imported, click “Next” and:
  - Enter Patch Tag name
  - Enter Changelog Comment
RPM Plugin Export of new project or no changes to previously-imported project
RPM Plugin (continued)

RPM Plugin export of a previously-imported RPM with changes

- Export an SRPM from a project
- Select a project to export:
  - copyDirTree
  - helloworld
  - pump

- SPEC file:
  - eclipse_pump.spec

- Patch Options:
  - Patch Tag:
  - Changelog Date Stamp: Wed Dec 03 2003 -- rmoseley <rmoseley@mail.com>
  - Changelog Comment:

- Composite Export Type:
  - Export a Binary RPM
  - Export a Source RPM

- Patch Needed:
  - The project has changed since last imported as an SRPM. To continue export as an SRPM, you will have to generate a patch (on the next page)
Source and Binary RPMs currently end up in the project's directory
RPM Plugin (continued)

Possible Future Enhancements

- Refactor org.eclipse.cdt.rpm.core and create an org.eclipse.cdt.rpm.utils class to further break out the shell commands
- Automatically run the CDT build when a source RPM is imported
- Allow user to select where to place exported RPM(s)
- Move more features into Eclipse framework as it becomes practical and move away from shell commands
- Automated builds
- Import/Export of Java projects
- JUnit tests
- Spec file editor
RPM Plugin (continued)

Where to file bugs/enhancement requests

- http://bugs.eclipse.org
  - CDT then CDT-Contrib

Mailing list

- On eclipse.org -> cdt-contrib-dev
CDT Libhover Plugin

Purpose

 To provide a mechanism whereby descriptive material may be popped up on the user's screen when the mouse pointer is allowed to pause for a period of time ("hovers") over a text string. The information provided by the Libhover plugin is that relevant to C programming libraries such as libc.

Description

 Libhover extends the Eclipse CDT through extension point CCompletionContributor. This extension is invoked under a variety of circumstances, one of which is the “hover” situation described above. When this occurs, the plugin is initialized (necessary only on the first invocation) and then an appropriate XML file is examined for an element corresponding to the supplied text string.
Libhover CDT Plugin (continued)

Where to get it

- ?

Requirements

- Eclipse platform
- Eclipse CDT

Libraries currently supported

- FSF/GNU standard C library: libc
Libhover CDT Plugin (continued)

Challenges

- Each library has a different format for their documentation. Each library supported by Libhover must be parsed and placed in properly structured XML files and that in turn requires that the original library documentation be presented in a form that can be efficiently and accurately abstracted into a Libhover XML file. Fortunately, libc is presented in the form of texinfo files which contain what may be interpreted as semantic tags for much (but not all) of the data needed to construct a Libhover XML file.
Libhover CDT plugin (continued)

Hoverhelp screenshot

```c
#include <stdio.h>

int ll = strlen("iiiiiii");

gtk_label
gtk_display

Name: strlen
Prototype: size_t strlen (const char *s)

Description:
The strlen function returns the length of the null-terminated string s in bytes. (In
other words, it returns the offset of the terminating null character within the array.)
For example, @smallexample strlen ("hello, world") 12 @end smallexample
When applied to a character array, the strlen function returns the length of the
string stored there, not its allocated size. You can get the allocated size of the
character array that holds a string using the sizeof operator:
@smallexample char string[32] = "hello, world"; sizeof (string) 32 strlen (string) 12
```
Libhover CDT Plugin (continued)

Possible Future Enhancements

- **Syntactical significance**
  - For example, in the C Fragment `struct foo { int foo; int bar; };` the string “foo” could refer to either the structure, resulting in the display of a description of the structure or of the inner member within the structure, resulting in a display appearing to the member.

- **Templates**
  - An extension of the Libhover XML mechanism could be devised that would support function call template generation.

- **Other libraries**
  - As alluded to above, Libhover itself can be easily be extended to include libraries other than libc (or, for that matter, large applications such as, gcc, X, etc.) “simply” by providing XML documentation.
Libhover CDT Plugin (continued)

Possible Future enhancements (continued)

- Other syntactic elements
  - The CDT implicitly assumes that data returned to it from its CCompletionContributor extensions refer only to function descriptions and formats the popups accordingly. Libhover, on the other hand, presently supports the descriptions of a large variety of other syntactic elements such as structs, unions, enums, etc., and could be easily enhanced to support others. For this use, however, the CDT itself would have to be modified slightly.
JPackage Project

What are the primary goals of the Jpackage Project?

- To provide a coherent set of Java software packages for Linux, satisfying all quality requirements of other applications
- To establish an efficient and robust policy for Java software installation

Secondary goals

- Focus is on using free and open source software and avoid proprietary software whenever possible. For convenience, nosrc packages of non-free source packages are provided when no other solution is possible.
- Provide generic RPMs that work in any RPM-based Linux distributions

Where is it?

- http://jpackage.org
JPackage Project (continued)

Why do I mention it?

- Eclipse RPMs reside there (usually the latest versions or pretty close)
- Red Hat Java stack is based on its work

Supports multiple update interfaces

- yum
- apt-rpm
- up2date

Number of packages available as of January 2005

- 1381
gcj Version of Eclipse

Status

- Test Eclipse RPM's pushed into Fedora Core 4 January 2005

Purpose

- JVM-less version of Eclipse
- Architecture coverage
- Performance
- Reproducibility
- Ability to fix bugs in compiler and not rely on proprietary JVM to build
- Fully F/OSS stack
Current deficiencies
  - No support for JWDP

Future Enhancements
  - JWDP support
  - Optimization for natively-compiled code
  - Native debugging (perhaps)
Eclipse Projects Currently Under Consideration

Bugzilla Plugin

- Red Hat is continuing development of the Bugzilla plugin that was started by the Eclipse.org team. We feel this plugin is very important to our customers and to the Open Source community.

Gnome Integration

- Gnome-wide preferences, General desktop and menu interactions, Session awareness, Native widgets, Recently used files, Yelp integration

Extend RPM plugin to create plugin RPMs

- Allow a user to export an entire plugin as an rpm that can be installed on any Linux system
Commitment to Eclipse

- Red Hat is working to make Eclipse the premier developer suite available to not only open source developers, but to all developers.

- The Eclipse development team at Red Hat invites any and all suggestions for enhancements to be filed at the appropriate location.

- For the OProfile and RPM plugins submit bugs and/or enhancement requests to: [http://bugs.eclipse.org](http://bugs.eclipse.org) -> CDT -> CDT_Contrib

- For everything else submit to: [http://bugzilla.redhat.com](http://bugzilla.redhat.com) -> Fedora Core -> eclipse